Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation

by

Shirin Akther Khanam

M.Sc. (Computer Science and Engineering), Sungkyunkwan University, 2015
B.Sc. (Computer Science and Engineering), Chittagong University of Engineering & Technology, 2007

A Thesis Submitted in Total Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Department of Computer Science and Information Technology
School of Engineering and Mathematical Science
College of Science, Health and Engineering
La Trobe University
Victoria, Australia

JUNE 2021

Declaration of Authorship

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Shirin Akther Khanam

25th June 2021

Acknowledgements

First and foremost, praise and thanks to the Allah, the Almighty, for giving me strength and blessings throughout my PhD candidature to finalize my research work.

I am extremely grateful to my supervisors, Dr. Fei Liu and Prof. Yi-Ping Phoebe Chen for their invaluable advice, continuous support, and patience during my PhD study. Their immense knowledge and plentiful experience encouraged me at all times throughout my academic research and daily life.

This work was supported by a La Trobe University Postgraduate Research Scholarship (LTUPRS). This support during my PhD journey is sincerely appreciated.

Finally, I would like to express my gratitude to my husband, my mother and my children. Without their tremendous understanding and encouragement over the past few years, it would have been impossible for me to complete this study.

Publications

The research thesis is based on the following publications.

[1] Khanam, S.A., Liu, F. and Chen, Y.P.P., 2021, July. Concept-based Topic Attention for a Convolutional Sequence Document Summarization Model. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8.

[2] Khanam, S.A., Liu, F. and Chen, Y.P.P., 2021. Joint knowledge-powered topic level attention for a convolutional text summarization model. *Knowledge-Based Systems*, *228*, pp.107273.

[3] Khanam, S.A., Liu, F. and Chen, Y.P.P., 2019. Comprehensive structured knowledge base system construction with natural language presentation. *Human-centric Computing and Information Sciences*, *9*(1), pp.1-32.

[4] Liu, F., Khanam, S.A. and Chen, Y.P.P., 2020. A Human-Machine Language Dictionary. *International Journal of Computational Intelligence Systems*, *13*(1), pp.904-913.

Abstract

Human beings usually summarize documents by reading them in entirety, developing an understanding of the meaning of the content and highlighting the main features. Automated text summarization (or text summarization) is a computerized process generating a summary for a given text. Text summarization is challenging as machines have limited human knowledge, a limited ability to understand natural language and a limited ability to grasp the main features using knowledge. In the thesis, we propose a Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation (KTSNR) to resolve the challenges in text summarization that comprise ontology-based machine-readable knowledge base (OMRKBS), a topic knowledge base (TKB) and a convolutional sequence network-based text summarization model with knowledge-powered topic level attention (KTOPAS). OMRKBS provides background knowledge about a term that is machine-interpretable and enables the machine to understand the text. OMRKBS uses natural language independent knowledge representation, rules and algorithms to transform and map the meaningful and structured information as background knowledge. TKB is a prior knowledge base which provides knowledge-powered topic information to an Abstractive Text Summarization model (ATS) model. TKB uses a knowledge-powered topic model (KPTopicM) to learn about the topic information that incorporates the background knowledge from knowledge bases (such as OMRKBS, ConceptNet and Probase) into a statistical model to produce coherent and meaningful topic information, which we call knowledge-powered topic information. KTOPAS is a text summarization model based on convolutional sequence networks with knowledge-powered topic level attention. The framework incorporates knowledge-powered topic information (which is received from TKB) with a high-level topic attention which enables KTOPAS to produce coherent and human-like summaries with word diversity. The experiment results show that (1) OMRKBS achieves higher accuracy than the other baselines, namely ConceptNet, DBpedia and WordNet; (2) KTOPAS achieves more competitive results than other baselines by generating human readable, meaningful and informative summaries; and (3) TKB improves the effectiveness of the resulting summaries by providing knowledge-powered topic information to KTOPAS and demonstrates the quality of the proposed system KTSNR.

Keywords: Deep Learning, Knowledge Base, Natural Language Representation, Topic Model, Concept, Abstractive Text Summarization, Convolutional Model, Attention Model.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Chapter 1.

Introduction

Automatic summarization means automatically summarizing text using machines. Automatic text summarization [1] focuses on compressing content into a brief version that carries the salient parts of the source articles [2]. The exponential growth of online text documents on the Internet means people have access to a huge amount of information such as news articles, scientific content, and legal documents which are much larger than the summaries of text information. Consequently, users must spend a lot of time finding their target information from the large amount of textual data. Therefore, the urge to generate concise summaries of documents is becoming increasingly important and essential. Manual summarization which is the process of rephrasing or paraphrasing the full text into its short version manually is clearly not useful for such a huge amount of text information. Hence, automatic text summarization has become an interesting research area to academics. A large body of research has been developed to improve automatic text summarization approaches over the last fifty years [1] [3-4].

Extractive and abstractive summarization are two methods of automatic text summarization. Extractive text summarization (ETS) captures a subset of either sentences or words from the source text in the generated summaries and abstractive text summarization (ATS) generates novel summaries that capture the main theme or meaning of the document and paraphrases sentences based on the experience of the previous histories of the summarization instead of selecting sentences from the original text. A large body of research has been published on extractive summarization over the past two decades [5-7]. Extractive summaries may lose the main context of the documents whereas abstractive summaries capture the actual context of the document. The ATS model represents the source text in a sequential representation, then produces a summary with new sentences that do not come from the source text. ATS produces text which is qualitatively closer to human-written phrases or sentences without being limited to phrases from the source document in the generated summaries. This is because ATS generates summaries from experience or history of the corpus and learns where and when to focus

on the specific words related to the source document. Recently, the ATS model [36][42][145][146] has shown significant improvement compared to extractive summaries since it retrieves the information from multiple documents to create a precise summary of information. Therefore, we focus on the ATS approach to generate concise and novel summaries. Recently, deep learning-based ATS approaches have been proposed which are able to meet the expectations of researchers due to their prominent features and the effectiveness of their approach such as representing sequential documents, capturing contextual and semantic information, and generating new sentences [2] [7-10].

1.1. Challenges

In general, the task of automatic text summarization is a significant challenge in the area of natural language processing (NLP) and artificial intelligence. Luhn [11] first introduced research on text summarization in 1958 which automatically generated the main theme of articles and papers. To provide summarized information of text by preserving the original meaning, many approaches have been proposed using either extractive [6], abstractive [10] or hybrid [8] approaches. These summarization approaches focus on various challenges in this research area such as: i) identifying the most important and relevant portion of the source text to be included in the generated summary [12-13], ii) generating new relevant vocabulary or phrases in the summaries [14-15], iii) multi-document text summarization [16-18], iv) summarizing text which is long in length [15][17] v) generating abstractive summaries [10][19-21] that are human readable and meaningful.

However, most approaches often fail to generate coherent and concise summaries, losing the main theme and deviating from human-written summaries [142]. We know human-written summaries are meaningful, readable, and relevant to the original document since human experts use short phrases or a small number of words to compose content, utilizing their previous knowledge and experience while focusing on the main topic and retaining the original meaning of the document. Humans usually follow three steps to summarize a document. First, they understand and summarize documents by reading them in entirety, developing an understanding of the meaning of the content, then highlighting the main features, and finally summarizing the text using short phrases or a small number of

words. By imitating the same steps taken by humans to generate summaries, it is possible that a system will be able to generate summaries similar to a human-written summary. Therefore, we focus on building a complete ATS system by following the same steps as those followed by humans: i) understand text using background knowledge, ii) capture the main ideas and original meaning of the source text iii) generate summaries using phrases or words. However, researchers are still attempting to improve the various techniques and approaches to enable machines to produce relevant summaries that match human-written summaries. This is because machines have a limited ability in terms of human knowledge and their ability to understand language, so building this system is challenging. Limited research has been conducted on constructing this kind of system which acknowledges background knowledge while generating abstractive summaries. We describe the three challenges: understanding text, capturing the main idea and generating abstract summaries in the area of the automatic text summarization of research fields.

1.1.1. Understanding Text

Humans use natural language for communication through speaking or writing. In computing, natural language represents the information of human knowledge and uses computational methods to analyze the textual data using a process known as natural language processing (NLP). This process gives computers the ability to read, understand and interpret human language. NLP investigates how to effectively generate and understand human language in text. This is not a simple task, as it involves a deep understanding of human language. To generate human-like summaries, the system needs to represent important, relevant semantic information of the source document. To extract the important relevant information, the system needs to understand the text and to understand the text, the system requires background knowledge of the text which is machine readable and interpretable. NLP techniques are used to analyze the syntactic and semantic information. Syntax refers to the way words are arranged in a sentence to allow it to make grammatical sense. Semantics refers to the meaning expressed by a text. Therefore, the domain of NLP research can be utilized in the problem of text summarization, in which the objective is to understand and shorten the source text using

the background knowledge while retaining the most important information and ensuring that the output summary is human-readable.

A machine-readable knowledge base (MRKB) is a repository that provides structured, logical, meaningful information about a term which is accessible and interpretable by the system. We call this information rich structured information (RSI). Huge efforts have been made to obtain the RSI and construct the MRKB with RSI [22-25]. Ontologies are representations of a shared conceptualization of various domains and great success has been achieved in the extraction, sharing, reuse and representation of knowledge. Recent research [24-25] which uses an ontology as a repository for MRKB has been proven effective. However, it is a long-term issue to build an ontology- based MRKB with RSI which can provide fully machine interpretable individual and structural features about a term. By resolving the challenges in constructing the ontology based MRKB, the MRKB enables the system to understand the context and the interpretation of words and the form of sentences.

1.1.2. Capturing the Main Idea

Due to the considerable amount of redundant and irrelevant information in the collection of texts, it is challenging to focus on important and relevant features while generating summaries. Therefore, ATS often fails to capture the salient information in the generated summaries. Extracting the salient features from the source text is one of the important steps in text summarization. Recently, topic models that are used for learning hidden topics have received much attention in automatic summarization research fields. A topic model is a kind of a probabilistic generative model that originated from the field of NLP and is widely applied in text mining. Latent semantic indexing (LSI), a topic model, was introduced in 1990 [26] and served as a base of the development of the topic model. However, LSI cannot be considered as a genuine topic model because this is not based on a probabilistic model. Later, probabilistic latent semantic analysis (PLSA) [27] based on LSA was introduced and is considered as a valid topic model. In 2003, a complete statistical topic model Latent Dirichlet allocation (LDA) was proposed [28] which is an extension of PLSA. Nowadays, LDA serves as a base for the improvement of the topic model and there are an increased

number of research studies which introduce the topic model based on LDA for various purposes, such as classification, clustering, feature selection, and topic identification [29-31]. LDA represents a document using the hidden labels (also called topics) as a distribution over topics where topics are distributions over vocabulary. As each topic is represented by a number of words with the highest probability from its distribution, topics can be expressed clearly. Therefore, a topic represents semantic and coherent information. LDA is an unsupervised approach and is able to effectively handle the challenges which are due to a lack of supervised and huge unlabeled data. However, the advantage of having a background knowledge of a document is not utilized properly in a statistical LDA model which often results in the failure to generate coherent and consistent information on a topic. Incorporating prior knowledge in a statistical topic model could potentially result in further improvement [32-33].

1.1.3. Generating Abstractive Summaries

Recently, many neural abstractive summarization models have been proposed that use either LSTM-based sequence-to-sequence [34-38], attentional models [39-40] or Transformer [9][41] as their backbone architectures. The attention model [39] is a mechanism to measure the weights to put attention at the encoder states for each decoder state in the neural network. These models also integrate various techniques into their backbone architecture such as coverage [41], copy mechanism [42], pretrained approach [43] and content selector module [44] to improve their performance. Furthermore, recent work has been conducted on abstractive summarization based on reinforcement learning techniques that optimize objectives in addition to the standard maximum likelihood loss [20][45]. Most models are based on the sequential recurrent neural network (S2S-RNN) ATS [36]. There are several drawbacks to using S2S-RNN: i) the current hidden state depends on the previous histories of the hidden state that prevents parallel computation within a sequence, ii) it is not trained to capture word dependency nor the patterns in the text, such as key phrases. On the other hand, the convolutional sequence network (CSN) [80] is a convolutional neural network [75] based architecture for seq2seq learning. In this model, the current state is independent which allows parallelization over every element in a sequence. Both RNN- and CSN-based ATS use the attention mechanism to decide which

source words in the inputs should be the focus when generating a summary at the decoder state. Researchers have proposed several CSN-based text summarization models. However, most approaches have a tendency to include irrelevant and ambiguous information from the source in the summaries. The reason for this is that these models consider the attention of a source word to the summary and do not consider high-level attention in terms of topic information or the background knowledge of the source text to the summary.

Limited models propose a topic-based-level topic attention where the topic information is chosen using the LDA statistical topic model [1][3]. These models incorporate the topic information in ATS using high-level attention. However, the topic information is based only on the statistical LDA model without prior background knowledge which may focus on irrelevant topic words and result in generating incoherent and redundant summaries. Conceptual information is the commonsense knowledge or external information of the source text from a concept-based knowledge base which captures the latent semantic information of the text and provides contextual information. Ontology-based knowledge is a kind of concept-based knowledge base. Using conceptual information as background knowledge in topic models would be a potential solution to enrich the novelty of topics. Currently, there has been limited research on deep learning-based ATS using conceptual information. To bridge the gap between the topic information and background knowledge of the document in the summaries, incorporating topic information based on background knowledge into a CSN-based summarization model could be effective.

1.2. Motivation

The motivation for the research is to build an automatic text summarization system which can understand documents by interpreting and apprehending the content using background knowledge, which is capable of capturing the topic information with that knowledge and which is able to generate a summary that is qualitatively close to human-written sentences. The question to be addressed is: how can a machine interpret and understand a document using knowledge, identify the topic information from the document and then summarize the text in a way that is close to human-written summaries?

1.3. Objective

In this thesis, our main objective is to build a system which enables a machine to summarize a document similar to the process followed by humans. For this, we focus on three tasks: first, understanding documents by interpreting and apprehending the content using NLP. We aim to obtain machine interpretable information (RSI) from the background knowledge of terms, and then construct an ontology-based machine-readable knowledge-based system by mapping the RSI so that a machine can read and understand the knowledge about the terms from that knowledge base. Second, we aim to obtain the salient features from the text using background knowledge. We incorporate the extracted RSI from the first step as the background knowledge of the source to the statistical topic model to reveal the hidden, contextual and meaningful topic information of the source text. Finally, we generate summaries that are qualitatively close to human-written sentences. We aim to summarize the document with a deep learning-based abstractive text summarization model (such as the CSN-based text summarization model) by incorporating the topic information generated in the second step. We propose a framework to build a complete ATS system based on deep learning called Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation (KTSNR) which consists of three steps: OMRKBS, TKB and KTOPAS.

First, we construct an ontology-based machine-readable knowledge base system (OMRKBS) using natural language independent knowledge representations to provide RSI about a term which is machine-interpretable and accessible. Natural language independent knowledge representations (NLIKR) is a scheme to represent the individual features of human knowledge that are readable by machines. NLIKR regards each word as a concept which should be defined by its relations with other concepts. ORMKBS employs algorithms and rules to transform the text into RSI. OMRKBS utilizes NLIKR to discover concepts and their relations in RSI and maps the RSI information in OMRKBS. OMRKBS helps the system to understand the text by providing the RSI as background knowledge while obtaining topic information in the next step.

Second, we construct a knowledge base to provide topic information based on background knowledge (which is retrieved from OMRKBS, ConceptNet and Probase) called a Topic Knowledge Base (TKB). We introduce a conceptualization algorithm to derive the probability distribution of concepts or the background knowledge of each word in the text. We propose the Knowledge-powered Topic Model (KPTopicM) which incorporates background knowledge using concept distribution into a statistical topic model to produce topic information. We trained KPTopicM and use this learned data as a prior repository called TKB. We use TKB as a prior knowledge base to enable the system to identify the topic information based on background knowledge while generating summaries in the next step.

Table 1.1: Example of generated output from the three-step framework

OMRKBS
Background knowledge of a concept: '*earthquake*': "An earthquake is the shaking of the surface of the Earth resulting from a sudden release of energy in the Earth's lithosphere". '*military campaign'*: "A military campaign is a long-duration significant military strategy plans incorporating a series of interrelated military operations or battles forming a distinct part of a larger conflict often called a war."
Structural information (RSI) of the concept from OMRKB: '*earthquake*': <*earthquake,* shake, surface> < *earthquake*, results from, release of energy, in, Earth's lithosphere> '*military campaign'*: <military strategy, plans> <incorporate, military, operations >< incorporate, battles> <form, conflict> <war>
TKB
Earthquake: Japan, earthquake, tsunami, disaster, shake, loss, nuclear, crisis, radiation, Asia military campaign':
KTOPAS
Input 1: A fairly large earthquake measuring a magnitude of 6.7 on the Richter scale rocked wide areas of central and western Japan Sunday, followed by four aftershocks, the meteorological agency said. *Input 2:* Barak Obama on Wednesday announced the closure of government schools with immediate effect as a military campaign against religious separatists escalated in the north of the country.
Generated Summary of Input1: Powerful earthquake shakes the wide area of Japan. *Generated Summary of Input 2*: America shutdown school because war escalated in the north of the country.

Finally, we propose a convolutional sequence network-based text summarization model with knowledge-powered topic level attention (KTOPAS) to generate a meaningful and concise summary. The framework incorporates knowledge-powered topic information (which is retrieved from the prior TKB) into a convolutional sequence text summarization model with high-level topic attention. This model introduces a tri-attention mechanism which enables the model to produce coherent and human-like summaries with word diversity. KTOPAS comprises a three-level CSN: word, knowledge and topic-level CSN

to capture the contextual alignment information from three aspects. The source inputs and knowledge-powered topic information are encoded at the encoder and the summary output at the decoder of the word and topic-level CSN respectively. The knowledge-level CSN encodes the input elements at the encoder to decode the knowledge-powered topic information at the decoder. We compute the attention jointly from the three-level CSN and combine them into one. We introduce a final probability distribution to predict the next target of the summary output at the decoder state.

In our earlier research, we first identified the topic information from the background knowledge using the classic LDA topic model and then incorporated the topic information into the CSN-based ATS model. We present this research in chapter 5. However, the classic LDA model does not consider word dependencies in topic information and only considers concepts as background knowledge. Therefore, we extend our research by proposing a topic model called the knowledge-powered topic model (KPTopicM) and constructing a prior TKB using the KPTopicM. TKB provides a knowledge-powered topic model to KTOPAS. KTOPAS considers indirect word dependencies and the direct dependencies of concepts in the topic information. We present an extension of this research in chapter 6. Once we built the system KTSNR, we trained KTSNR using the Gigaword and CNN/Daily mail datasets. Table 1.1 shows examples of the output of our three-step framework. These examples will be used in this thesis to illustrate the steps of the framework. We see from Table 1.1 that OMRKB generates the RSI for each concept, such as ‘earthquake’ and ‘military campaign’, TKB generates topic information based on the background information of ‘earthquake’ and ‘military campaign’, KTOPAS generates summaries for the two examples where these concepts are key words.

1.4. Contributions

The major contributions of the research are as follows:

KTSNR: We build a Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation (KTSNR)

to generate human readable, meaningful and concise summaries that resemble human-written summaries. We describe the contributions of the three parts of KTSNR as follows.

A. OMRKBS

- We propose a framework to automatically develop a comprehensive ontology-based machine-readable knowledge base system (OMRKBS) with RSI to provide machine interpretable, individual, meaningful and salient features with a diverse range of vocabulary.
- We propose algorithms to transform the text to RSI and devise formulas to discover concepts and their relations in the RSI and design a program to map RSI in OMRKBS.
- The OMRKBS achieved better results in terms of accuracy compared to the others KBSs, namely Dbpedia, WordNet and ConceptNet.

B. TKB

- We propose a conceptualization algorithm that retrieves semantically relevant and salient background knowledge of the document.
- We develop a KPTopicM algorithm to incorporate conceptual information in the topic model to generate coherent and relevant topic information to the source text.
- We construct a prior topic knowledge base using KPTopicM to provide knowledge-powered topic information to KTOPAS.
- KPTopicM achieved better results in terms of accuracy compared to the other topic models, namely LDA and KB-LDA.

C. KTOPAS

- We propose a deep learning-based abstractive text summarization model with knowledge-based topic level attention (KTOPAS) that incorporates generated topic information-based background knowledge using a high-level topic attention to produce coherent and human-like summaries with word diversity.
- We introduce a tri-attention channel which jointly learns the attention of the word, knowledge, and topic level attention, and then combines the three attention weights

into one and produces the final attention weight to generate semantically well-formed and coherent summaries.

- We produce the final probability distribution of the next target element in the output summary at the decoder of the word and topic level convolutional network.
- KTOPAS achieved better results in terms of accuracy compared to other state-of-the-art methods.

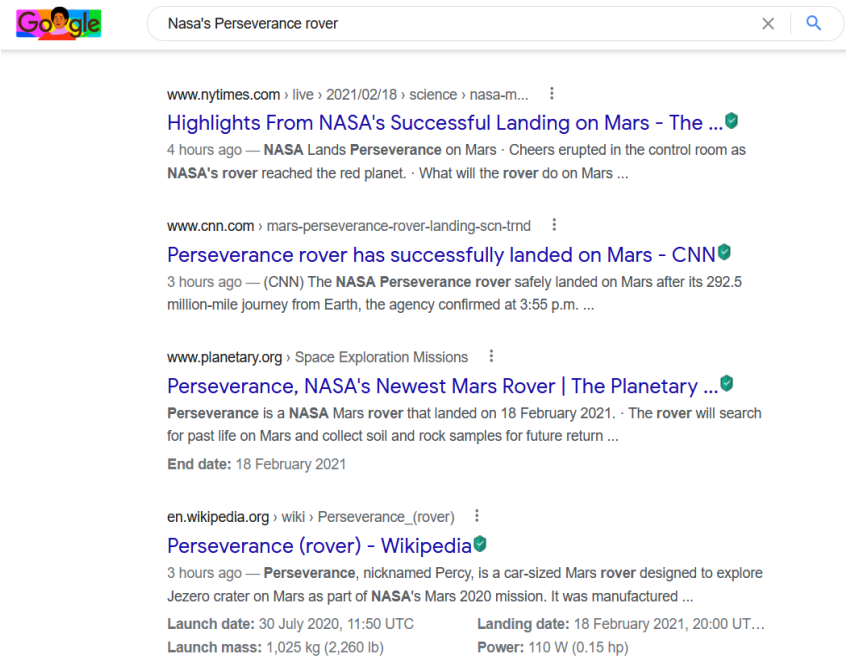


Figure 1.1: Example of the application of summarization in real technology.

1.5. Significance of the Research

As we know, there is a massive amount of data available upon request on the Internet. Most data appear to be unnecessary to the user, making it difficult for users to learn about news, events, objects, services or terms. A considerable amount of time and effort is required to extract actual facts from the large amount of textual data available. A concise and meaningful summary of a source description can give users an idea about the facts. A user does not have to spend time and effort to read the whole content which possibly contains unnecessary information. They can look at the brief and concise summary and obtain a simple idea of what the content is about. Figure 1.1 shows an example of the query result

of the search term ‘nasa perseverance rover’ from Google. ‘Perseverance is a rover which Nasa launched to explore Mars in 2020’. We can obtain news about the Perseverance rover from Google where each result contains a headline and a summary of the news. Therefore, the user can understand what the Perseverance rover is when they read the summary of each result. Moreover, a short summary, which conveys the essence of the document, helps the search engine find relevant information quickly. This is because summarization provides a way to cluster similar documents and present a summary. When a search engine obtains the summaries of each result for the query, the system clusters similar results and presents the results to the query. Summarization can be applied in various areas such as: search engine queries, generating news headlines, monitoring the media, classifying the intent of chatbots, interpreting product reviews and so on. Therefore, text summarization has attracted a high level of interest from researchers. However, ATS models have various limitations due to the challenges we mentioned earlier such as a lack of understand the content, a failure to capture topic information and a focus on irrelevant information. We focus on generating human-like summaries by exploring the advantage of following the same procedure followed by humans and resolve the challenges by retrieving the background knowledge, understanding the content, and identifying topics using contextual and semantic information.

In this thesis, we identify the research gap in utilizing background knowledge while capturing topic information in the generated summaries. Our OMRKBS provides latent and structured information which helps the model understand the content. We use this OMRKBS to retrieve the semantic and informative background knowledge and use KPTopicM to obtain coherent and consistent topic information from this background knowledge. We built a prior topic knowledge base (TKB) with the pre-trained KPTopicM model. Also, the importance of rare words is ignored sometimes since the current state-of-the-art ATS model does not consider knowledge about rare words. Conceptualization and TKB help to recognize the importance of rare words in the document using the background knowledge of the words in the document. Our summarization model KTOPAS uses this TKB to capture the coherent salient information from the knowledge background in the generated summaries. KTOPAS utilizes the properties of CSN to improve the accuracy and execution time of our proposed ATS model. The tri-attention channel makes the

KTOPAS model more efficient by capturing more relevant latent semantic and contextual information from the text while generating coherent and semantically well-formed summaries. The probability distribution helps to decide the output of each state and whether to include the information from the original or topic knowledge and reinforcement learning to maximize the output. Finally, the full KTSNR system is capable of generating coherent, concise and relevant summaries with word diversity in a way which is similar to humans.

1.6. Structure of the Thesis

Chapter 2 provides a literature review, background on the fundamental, recent research and the competitive baselines that are related to this thesis, starting with the knowledge extraction step which involves various commonly used knowledge bases, natural language processing (NLP), NLP tools and the natural language representation approach. Then, the LDA topic model is reviewed which is widely used by the suggested methods to obtain topics from data mining the literature. Finally, we discuss the fundamental background which is required to develop an ATS model, such as word embedding, neural network theory, encoder-decoder sequence network and the attention mechanism, and the recent work on automatic text summarization from extractive to abstractive summarization.

Chapter 3 presents the problem of the abstractive text summarization model in detail and gives a brief description of whole thesis methodology, describing how the methodology solves the problem step by step and details the significance (contribution) of our research.

Chapter 4 describes the construction of the OMRKBS framework to construct an ontology-based machine-readable knowledge base to provide machine readable information about a term. It starts by discussing the challenges in obtaining RSI and constructing a machine-readable knowledge base and introduces solutions to resolve these challenges. Then, it describes the related work and recent work on knowledge base system construction with RSI. We present our methodology for the construction of OMRKBS, followed by OMRKBS output representation. Then, the experiment results on the available dataset are discussed. We published a journal paper using the research in this chapter.

Chapter 5 proposes a CSN-based abstractive text summarization model called TEXSCTTA. This model incorporates topic information based on the background knowledge using a high-level attention mechanism. First, we discuss the challenges in capturing salient information due to the lack of background knowledge in the text summarization model, followed by the recent work on the ATS model. Next, we propose a convolutional sequence network (CSN)-based ATS model to resolve the challenges. Then, we conduct the experiments and evaluate our model against other models. Finally, we conclude the chapter. One conference paper has been accepted based on the research in this chapter.

Chapter 6 is an extension of the research in chapter 5. In chapter 5, topic information is chosen from the background knowledge based on the classic LDA topic model. In this chapter, we propose a new topic model KPTopicM and topic information is obtained using KPTopicM. This KPTopicM has been incorporated in the proposed ATS model discussed in chapter 5. This chapter first introduces the problem of obtaining coherent topic information and generating meaningful and concise summaries due to the failure to identify coherent information. Next, the related and recent work is discussed. Then, an approach is presented to construct a prior topic knowledge base using KPTopicM to provide topic information to our summarization model. Then, we propose knowledge-powered topic-level attention to the convolutional sequence-based text summarization model (KTOPAS) to incorporate the topic information generated from the prior topic knowledge base. Finally, we discuss the results of our experiment and the evaluation. Our experiment shows that KTOPAS improves the accuracy of the summary results compared to the TEXSCTTA model discussed in chapter 5. We submitted a journal paper based on the research in this chapter.

Chapter 7 is the final chapter where we draw a conclusion, discuss the limitations and provide suggestions for future work.

Chapter 2.

Related Work and Background

There is a large body of research on automatic text summarization from obtaining the meaningful background knowledge and topic information of the document to summarizing the document using topic information. Earlier [1] research extensively focused on the area of automatic text summarization; later significant developments were made in summarization from extractive [5] aspects to more abstractive [8] aspects. This chapter reviews the recent literature on the progress made on knowledge extraction, topic identification and automatic text summarization.

2.1. Knowledge Representation

Humans can easily understand the meaning and identify the topics of content since we have knowledge of an object or term which has been acquired through experience over years and stored in memory. We take the following sentence as an example: "A fairly large earthquake measuring a magnitude of 6.7 on the Richter scale rocked wide areas of central and western Japan Sunday, followed by four aftershocks, the meteorological agency said." As we have knowledge of earthquakes, we know this is not a person, rather this is a natural disaster which can shake the earth's surface. However, machines are unable to comprehend what they read, so it is a significant challenge for machines to understand the meaning of a text segment in natural language. To start, we describe the related work and the techniques in knowledge representation.

2.1.1. Natural Language Representation

A natural language is a tool in the representation of the individual features of human knowledge. Natural language processing (NLP) enables machines to extract and analyze information about terms. Since our interest is to structure text into machine-readable information, which is aligned with the purpose of natural language, our focus is on representing information using NLP.

2.1.2. Machine-Readable Knowledge Base

A machine-readable knowledge base (MRKB) [46] stores features and structured information on terms which is accessible by systems. The MRKB maps the structured information to enable the machine to interpret and understand the information. Researchers have made progress in extracting features and obtaining structural information from various sources. However, these methods often face challenges in obtaining information that is machine interpretable and easy for human beings to understand. Moreover, the explicit representation usually ignores the context of text and cannot capture the semantic features of a document. We know that human language requires taxonomies or ontologies to interconnect concepts in the domain to understand text. There are many knowledge bases or ontologies for a specific domain [47-48], but the availability of a useful comprehensive knowledge base is limited online. These knowledge base systems share two drawbacks: i) concept space is limited in the existing knowledge base, ii) the processes to construct a knowledge base to improve the quality of information are not accurate. Thus, there are inaccuracies and inconsistencies in the knowledge base while under construction.

A concept-based knowledge base [49-53] is a type of MRKB that utilizes taxonomies and ontologies to discover concepts and establish the relationships among concepts to map the information in the MRKB. An ontology is the formal representation of knowledge by a set of concepts that can be operated as a knowledge base in various text mining tasks such as clustering [54-55], classification [56-57], summarization [130-131] and others. OWL is the ontology representation language used and the output from Protégé-OWL is an XML-based file format, which facilitates further application and communication. The construction of a knowledge base system (KBS) based on an ontology is gaining increasing attention from the research community [22][60][46]. Most research studies provide descriptions of concepts using the relations between concepts [61-62] and a more enriched meaning [63-64]. The attributes or descriptions are from sources that are publicly available but are difficult to obtain and structure into a single KBS [46]. Currently, there are a large number of studies on the construction of these resources, some of the most commonly used being ConceptNet [50], FrameNet [65] and SUMO [52]. However, building an ontology-based knowledge base manually is a huge task that requires much time. Moreover, obtaining rich

structured information that represents knowledge about terms is tedious work [50]. There have been many efforts to transform unstructured data into structural information to improve data quality [53][66]. Additionally, some approaches are restricted to a single domain, hence they are not applicable to other domains.

Source of Knowledge-Based Systems

There are various knowledge bases from which to extract knowledge as concepts or instances. Several examples are as follows:

- BioPortal [47]: BioPortal is one of the largest repositories of biomedical knowledge based on ontologies in the world. These ontologies provide important and fundamental knowledge on various domains to accelerate the extraction, annotation, processing, integration and representation of data.
- ConceptNet [50]: is an open-source comprehensive knowledge base based on a semantic network that is designed to help systems understand the meanings of terms as humans do [53]. ConceptNet represents the relations between words such as: A net is used for catching fish. “Leaves” is the plural form of “leaf”.
- DBpedia [49]: The DBpedia is a knowledge base which contains structured information extracted from Wikipedia which covers many specific domains and background knowledge. This knowledge base is able to generate results for expressive queries. There are more than 400 domains in the DBpedia ontology.
- CRISP [48]: The Computer Retrieval of Information on Scientific Projects (CRISP) thesaurus contains terminology used for indexing biomedical information. It comprises more than eight thousand preferred terms that are categorized hierarchically into eleven domains.
- Probase [51]: Probase is a probabilistic taxonomy-based universal and comprehensive knowledge base system containing conceptual information which is the probability of the concept set belonging to a term or word. Probase reads a large number of documents to discover extensive concepts and obtain instances and attributes for each concept that build the relationship among them. Furthermore, Probase measures the weight scores of the concepts, instances, attributes and their relationships and these scores can be used to make inferences over textual

information. Then, the most likely concepts are derived from a set of words or a short text.

- WordNet [66]: WordNet is a large knowledge base based on an English lexical database. In WordNet, a concept is represented by parts of speech which is categorized into sets of rational synonyms (Synsets). Synsets are interconnected by establishing relations between semantic and lexical concepts. WordNet is also a publicly available database which can be downloaded from the link https://wordnet.princeton.edu/. The structure of WordNet makes it a useful tool for natural language processing and computational linguistics.

2.1.3. Natural Language Processing

Natural language processing (NLP) [67] is a computational technique to learn, understand, and produce human language content. NLP systems can be used to facilitate human-to-human (i.e., machine translator) and human-to-machine (i.e., chatbot agents) communication. Moreover, using the advantage of both machine and human knowledge (such as the immense volume of human language online as content) allows people or systems to understand unknown language. Over the last two decades, research in the NLP fields has been applied to practical technology for consumer businesses (such as Facebook, Google, Twitter, Amazon, and Chatbot). These developments have been motivated by the following: (i) an exponential growth in computing functions, (ii) the accessibility of gigantic amounts of data (iii) improvements in deducing the structure of natural language and its contextual information and (vi) significance progress in machine learning technology. These achievements present a computational approach to the Semantic web that combines statistical analysis and ML with a knowledge of natural language.

Earlier, NLP research focused on automation of the reasoning of the semantic and syntactic structure of language and the development of base methods, such as voice recognition and translation. Currently, the attention of researchers is focused on the development of real-world applications of these methods such as generating chat and conference systems, voice-to-voice translation engines, mining such as the classification and summarization of

massive amounts of information on health, finance, research, new articles etc., obtaining the topic and sentiment (positive or negative) of reviews on products and services.

Preprocessing

We describe the process of extracting structured information from unstructured text using NLP in the following.

- **Sentence Segmentation:** This step breaks the text into separate sentences.
- **Tokenization**: This step divides the text into sentences and the sentences into words. First, words are converted to lowercase and then punctuation is removed from words. Words with less than three characters are removed from the vocabulary.
- **Stop word removal**: This step removes stop words.
- **Lemmatization:** is a process of transforming the different forms of a single word to its base or dictionary form that have the same meaning.

  Example 1: Let *a* be the various forms of *b* base word. ***a***→ Lemmatization→b means that lemmatization transforms *a* to *b*.

  troubled → Lemmatization → trouble
  constructing → Lemmatization →**c**onstruct
  better→ Lemmatization→ good

  A lemma of a word is its dictionary or canonical form. To extract a lemma correctly, lemmatization identifies the part of speech of the words (such as noun, verb, adjective, or other) in the text. Also, the contextual meaning in the document needs to be considered when extracting the lemma.

  Example 2: Take the word "shaking" as an example. The word is a noun for the sentence "a sudden, violent shaking of the earth's surface is caused by a powerful earthquake", a verb for the sentence "An earthquake is the shaking of the surface of the Earth resulting from a sudden release of energy". Lemmatization returns "shaking" when the word is a noun and returns "shake" when the word is a verb.

- **Name entity recognition:** The aim of name entity recognition is to identify and tag the concepts as nouns if they represent the predefined groups such as a person, organization, place, country, etc.

NLP Tools

In this section, we describe several important NLP tools which we use to implement our system.

- **Stanford CoreNLP:** Stanford CoreNLP is a widely used tool for natural language preprocessing. It is a flexible NLP tool that offers multiple annotators, such as POS taggers, lemmatizes, named entity annotators, sentiment and coreference annotators, annotators for constituency and dependency parsing [68]. There are instructions for NLP that can be executed using the command line through the Java API or Python packages.
- **Protégé:** Protégé is an open-source software tool for editing ontologies through which users can construct and update a knowledge base [69]. There are many plugins for various services to manipulate ontologies, such as the integration and management of multiple ontologies, visualization of the graph of the ontology, inference query engines, importation of large data and so on. The interface for direct manipulation allows programmers to build and update the ontology domains that consist of important concepts and their relationships in the knowledge base. Cellfile is a Protégé plugin which is used to import CSV data into the ontology, however, this plugin cannot load large data.
- **OWL API:** There is an OWL API based on JAVA for implementation such as creating, manipulating, and serializing the ontology-based OWL. The tool supports the construction and editing of OWL ontologies, inferring over ontologies, and the utilization of ontologies in the knowledge base [65].
- **Mapping Expression:** Mapping expression is a mapping language which transforms the data enclosed in the spreadsheets to the OWL. However, the current techniques in mapping often experience challenges in converting data. First, most approaches are designed to process simple data in spreadsheets [61]. Generally, they consider that the data in the spreadsheet table complies to the relational rule.

The rows define various entities, and the columns define the values of the correspondence entities in the table. We call this rule the 'value-per-column'. However, the practical or real spreadsheet data do not comply with this simple rule since there are various widely used spreadsheet tools which are formable without the restriction of specific tabular structures. Recently, researchers have focused their attention to overcome the value-per-column challenges and to support mappings for inconsistent spreadsheet data [49]. However, most techniques employ an RDF-based technique to map the mapping expression. This is effective for mapping spreadsheet data to RDF but is very inconvenient for mapping data in OWL due to its redundant RDF serialization.

SPARQL

SPARQL is a semantic query language and protocol to extract and process data stored in the Resource Description Framework (RDF) format. SPARQL has the ability to query the required and optional network relations along with their associations and disassociations [70]. The system queries are responsive since variables can exist in complex class expressions and relate to the class or property names [71].

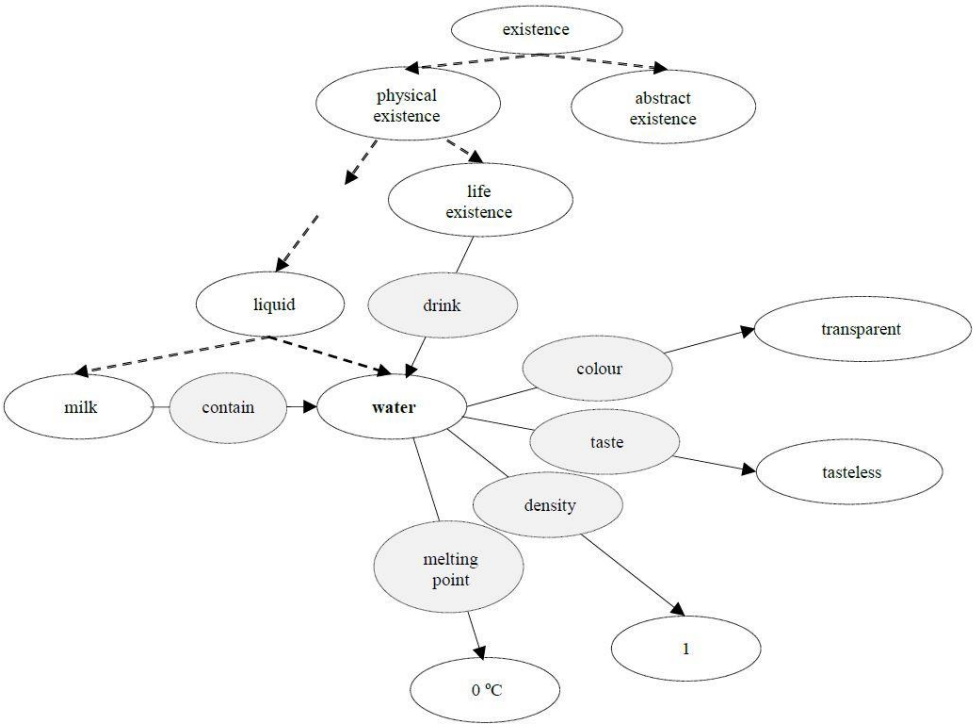


Figure 2.1 : An example of concept representation in the NLIKR scheme

2.1.4. NLIKR Scheme

Concepts, also called classes, are a core component of most ontologies. A concept represents a group of different individuals who share common characteristics which may be specific. NLIKR is a scheme which is based on a dictionary theme where the user can search a dictionary and information will be more meaningful and logical. This scheme was proposed by Liu and Chen [72] where each English word is represented as a concept. Description does not define each concept or word. Word or concept are defined by its properties (i.e., its relationships with other concepts). A characteristic or nature of a concept is represented through its relationships with other concepts. As a result, concept definition can go beyond human language since every word is a concept and is defined with another concept. A concept succeeded its super concepts properties. For example, 'water' is a sub-concept of 'liquid'. Therefore, 'water' inherits characteristics of 'liquid', such as liquid has no fixed shape. Figure 2.1 details the association between 'water' and other concepts such as '<water', 'colour', 'transparent>', '<water', 'taste', 'tasteless>' and '<water', 'density', '1>' represent the properties of 'water'. In the associations, 'color', 'transparent', 'taste', 'tasteless' are all concepts.

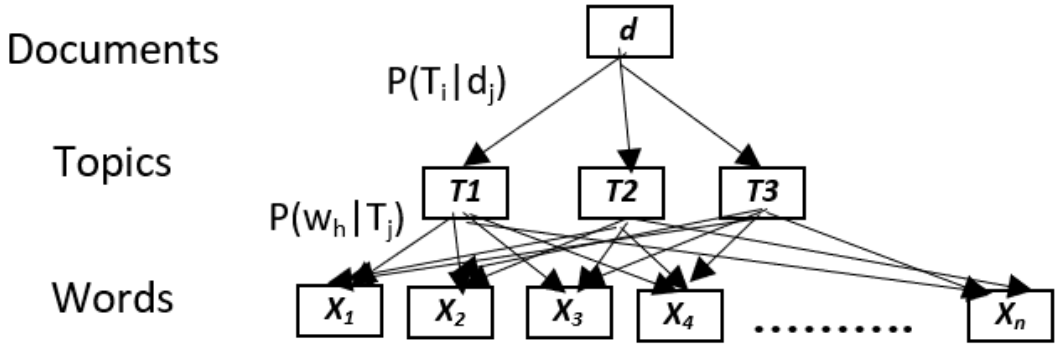


Figure 2.2: Three-layer hierarchical LDA technique

2.2. LDA Topic Model

Topic models specify the task of obtaining salient latent information that best defines a set of documents. We call this salient information topics. These topics are generated during the process of topic modelling. Latent Dirichlet Allocation (LDA) [28] is a widely known topic modelling technique. LDA is an unsupervised generative probabilistic method for modeling

a corpus. The idea of LDA is very straightforward. LDA assumes a definite set of topics and each topic is represented by a set of words. The aim of LDA is to map the documents to the topics where these topics represent most words in each document. Figure 2.2 shows the dependencies of the topic on the word and the document on the topic to measure the probability distributions among them. In the LDA model, first, the random assignment of latent topics represents the documents. A topic is defined by a probabilistic distribution over words and each document is represented by a probabilistic distribution over topics. A topic with a set of words which has the highest probability is able to define a topic well in LDA. In the LDA model, Dirichlet distribution is the distribution over a set of topic distributions where each topic distribution is themselves a distribution per document, or the distribution over set of word distributions where each word distribution is themselves a distribution per topic. Here, α and β are the parameter of Dirichlet priors over the topic distributions per document and word distributions per topic respectively. α controls how topic represent the documents. High score α indicates most topics represent a document or low α indicates few topics represents a document. β controls how words represent the topics. High score β indicates most words represent a topic and low β indicates few words represent a topic. We refer to Dirichlet as *Dir*. Multinominal distribution means the distribution of the number of outcomes, and we refer to multinominal distribution as Mult. Let *k* topics describe a set of documents and the set of topics in each document d are represented by a *k*-Mult. The Dirichlet is a probability distribution over the k Mult of a topic set. Dirichlet distributions encode the intuition that documents are related to a few topics.

2.2.1. Data Pre-processing

Documents are preprocessed through sentence segmentation, tokenization, stop word removal, lemmatization and removing the empty or null value.

2.2.2. Allocation

Once the Dirichlet is obtained, the topics of the documents and the words of the documents to topics are allocated. The Dirichlet parameter supervises whether each word in a topic has the same probability or if some words have a bias to a topic. Similarity, the Dirichlet

parameter supervises the distribution of topics in a document. Let D ∈ {d_1, d_2,…, d_m}be a corpus of m documents and n_d be the number of words in document d. P(z|d) is the probability distribution of topics z ∈ {t_1, t_2, …, t_k} in document d, also denoted as θ_d and P(w|z) is the probability distribution of words w ∈ {w_1, w_2, …, w_n} in topic z, also denoted Φ_z. Table 2.1 shows all the notations which are used to define the LDA model in this chapter. The probability of a word w given in the document d, P(w|d) is equal to

$$\sum_{t_i \in t} p(w \mid t_i) p(t_i \mid d) \qquad 2.1$$

where *t* is the set of topics.

Table 2.1: Notations are used for the topic model in this chapter.

V	the size of the vocabulary
k	the number of topics
d	a document
w_i	a single word in the document d at position i
w_d	the vector of word assignments in document d
z	a topic (label)
z_i	the topic assignment to a word token w_i
z_d	the vector of topic assignments to all word tokens of a document
	is assigned to topic z
n_d	number of word tokens in d
n_{dz}	number of word tokens in d that have been assigned to topic z
ϕ_z	word distribution for topic z
θ_d	topic distribution for document d
α_z	Dirichlet parameter on θ for topic z
β_w	Dirichlet parameter on ϕ for word w

2.2.3. Algorithm

To learn the weights of these two matrices $\theta_d$and ϕ_z, LDA models document D through the following generative process:

- Probe each document and assign each word in the document to a topic randomly. This generates a random topic distribution over documents and word distributions over topics.

- The distributions are improved by adjusting metrics. To adjust the metrics, probe each word *w* in each document *d* and measure: P (z |d) and P (w| z):
- This generative model predicts all the assignments of words except word w to the current topic are right. Reassign word w a to new topic z with the probability P(z|d) * P(w|z).
- Repeat this step for the entire document.

When the last step is iterated a large number of times, a steady state is reached where topic assignment fits well. The topic sets of each document are obtained by utilizing these assignments. Therefore, these assignments are used to estimate a topic set of a document d using the probabilities of words assigned to each topic in document d and estimate the association of words to each topic using the probabilities of words assigned to each topic overall. The formal way to define this algorithm is as follows:

(a) sample a *word distribution* for each topic $z \in \{t_1, t_2, \ldots, t_K\}$ which is ϕ_z from a *Dir* with parameter β

(b) select a *topic distribution* for each document $d \in \{d_1, d_2, \ldots, d_m\}$ which is θ_d from a *Dir* with parameter α.

(c) For each word *w* in each document $d \in \{d_1, d_2, \ldots, d_M\}$

i Draw a topic z_n from *Mult* (θ_d).

ii Draw a word w_n from *Mult* $\phi(z_n)$.

2.2.4. Gibbs Sampling

Gibbs sampling [73] is an algorithm to sample the conditional distributions of variables successively and this converges to the steady state through the long run iteration. This algorithm is used for LDA to adjust the parameters θ and ϕ, and to operate assignments on the topic variable z_n. This algorithm modifies the assignment z_n of word w_n in document d to a topic $j \in \{t_1, \ldots, t_k\}$ for a large number of iterations. The conditional probability of a word w_n in document *d* that represent a topic j is computed by the following equation.

$$P\left(z_{d,i} = j \mid \vec{z}_{-d,n}, w_i, \alpha, \beta\right) = \frac{n_{d,j} + \alpha_j}{\sum_x^k n_{d,x} + \alpha_i} \frac{v_{j,w_i} + \beta_{w_i}}{\sum_x v_{j,x} + \beta_i} \qquad 2.2$$

where $n_{(d,x)}$ denotes the number of times topic *j* is used to represent document d, $v_{(j,x)}$ denotes the number of times the given word *w* is used to represent topic *j*. We can see that the equation has measure weights of two aspects. In the first part, the weight provides the strength of the association between the topic and document that expresses how much a topic represents a document and in the second part, the weight provides the strength of the association between a word and topic that expresses how much a word represents a topic. A vector is obtained that explains how likely it is that this word belongs to each of the topics. We see from the equation that the Dir parameters α and β serve as a smooth function which gives scope to a word to represent a topic in the future, even though the value of $n_{d,j}$ or $v(j,w)$ is zero.

2.2.5. Example of LDA

We describe an example of the LDA technique for illustration. We use four sentences as documents to show how LDA works.

Document 1: A fairly large earthquake measuring a of 6.7 on the Richter scale rocked wide areas of central and western Japan Sunday, followed by four aftershocks, the meteorological agency said.
Document 2: Barak Obama on Wednesday announced the closure of government schools with immediate effect as a military campaign against religious separatists escalated in the north of the country.
Document 3: An earthquake is the shaking of the surface of the Earth.
Document 4: A military campaign is a long-duration significant military strategy plan incorporating a series of interrelated military operations or battles forming a distinct part of a larger conflict often called a war.

First, we choose a number of topics and randomly assign a topic to each word in the document.

t_3	t_2	t_1	t_3	t_2
earthquake	military campaign	country	shake	Japan

Second, we repeat this for each document in the corpus to find the total count of words in each document associated with the topics.

	t_1	t_2	t_3
Document 1	1	1	2

Then, we find the total count of each word in the corpus associated with the topics.

	t_1	t_2	t_3
earthquake	15	0	35
military campaign	7	50	1
country	42	10	15
shake	5	0	20
Japan	10	8	15
war	5	50	0

Again, we reassign a word such as 'Japan' from a document to a topic randomly. We show the effect of the reassignment in the count for the 2^{nd} iteration as follows. To reassign, the word 'Japan' is removed from topic t_2.

t_3	t_2	x	t_3	t_1
earthquake	military campaign	Japan	shake	Country

As a result, the count of the word 'Japan' in document 1 that belongs to topic t_2 decreases to zero.

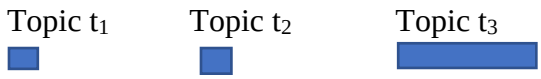
	t_1	t_2	t_3
Document 1	1	0	2

Also, the total count of the word 'Japan' that belongs to topic t_2 decreases.

	t_1	t_2	t_3
earthquake	15	0	35
military campaign	7	50	1
country	42	10	15
shake	5	0	20
Japan	10	7	15
war	5	50	0

Then, we compute the weight which tells us how much each topic represents a document based on the assignments. Let n_i be the number of words in document i, n_{iz} is the current assignment of $z \in \{t_1, t_2, \ldots, t_K\}$ topics in document i, α is the Dirichlet parameter and k is the number of the topics. The weight is calculated using the following equation which shows the association of a document to topics. This is the first part of equation 2.2.

$$\frac{n_{iz} + \alpha}{n_i - 1 + k\alpha} \qquad 2.3$$



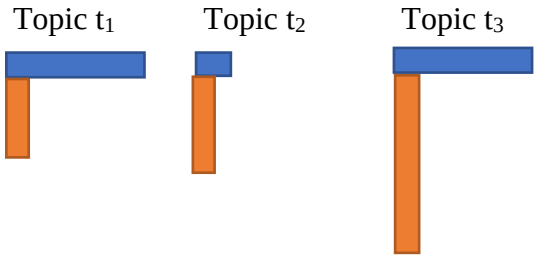
The blue horizontal bars show the weight scale of how much document i represents each topic using the weight obtained from part 1 of the equation. We can see that topic t_1 and t_3 represent document 1. Let $c_{w,z}$ be the assignment corpus-wide of a word w to topics z. The weight is calculated to measure the association of a word to each topic using the following equation.

$$\frac{c_{w,z} + \beta}{\sum_{w \epsilon V} c_{w,z} + \beta w} \qquad 2.4$$

We compute the weight which tells us how much a word such as 'Japan' represents each topic based on other documents in the corpus.

	t_1	t_2	t_3
Japan	9	7	15

We calculate the weight association of a word in document i using equation 2.2.



The orange bars show how much each topic represents a word in the document using the weight obtained from part 2 of equation 2.2. We can see that topic t_3 fits the word 'Japan' and document *i*. Topic t_2 fits the word 'Japan' but does not fit document i. Topic t_1 fits document *i* but does not fit the word 'Japan'.

We reassign the word 'Japan' to topic t_3 since this word represents topic t_3 the most.

t_3	t_1	t_2	t_3	t_3
earthquake	country	military campaign	Japan	shake

Incremental count based on the new assignment.

	t_1	t_2	t_3
Document 1	1	0	3

Total count of the assignment of each word in each topic.

	t_1	t_2	t_3
earthquake	15	0	35
military campaign	7	50	1
country	42	10	15
shake	5	0	20
Japan	10	7	15
war	5	50	0

By repeating the method, the topic model reaches a steady state which means each topic is represented by topic words well and each document is represented by topics well.

2.3. Abstractive Text Summarization

Text summarization is abstractive when the sentences do not appear in the original source text in the generated summaries. Instead, summary sentences are produced from the paraphrase of the main sentences of the source text. The paraphrase or new vocabulary that is different from the original document is learnt from the given dataset collected from the human-written content such as articles, product reviews, and news using artificial intelligence techniques such as artificial neural networks. Artificial intelligence is a technology that enables computer systems to perform a task which usually requires human intelligence. Since the dataset is coming from the written summaries of human knowledge, artificial intelligence techniques are capable of learning to generate summaries as humans do. Humans build a semantic representation of text in their brains, then choose the words from their vocabulary of general knowledge that suits the semantics to generate a summary to represent the main theme of the content. However, developing this kind of artificial intelligence technique is not easy. This requires the natural language generation technique. Recently, a deep learning-based neural network which is part of the artificial intelligence approach has been proven very effective for natural language generation techniques such as summarization, classification, and so on. Deep learning is able to resolve the challenges in representation learning by introducing contextual alignment. We describe the fundamental theories for a simple neural network, deep learning network, and the recent research on abstractive text summarization models based on the deep learning network.

2.3.1. Word Embeddings

A word embedding [74] is a pre-trained model to represent word vocabularies of the document. This represents the words in a similar group if they have a similar meaning. This is a method which represents words as a vector in a predefined vector space. Each word is embedded to one vector and a similar mechanism of a neural network trains this vector. This is a key advancement in the research on deep learning and can resolve the challenges in natural language processing. Word embedding is able to capture the contextual information of a word in a document such as semantic and syntactic similarity, the relation among the words in the text and so on. Word2Vec is a word embedding algorithm based on a statistical approach to learn an independent word embedding model from a large dataset efficiently. This drives the training of the neural network-based model, such as classification, translation, and text summarization, more efficiently. In the neural network-based model, input and output are embedded using the word embedding model at the encoder and decoder, respectively to capture the contextual information of the text to achieve more efficient results.

2.3.2. Simple Neural Network

An artificial neural network (NN) [75] is a model that is inspired by the mechanism of the human nervous system (such as the information processing mechanism in the human brain) to process the information. First, the feedforward neural network was proposed by [76] which is a simple artificial neural network model. A NN consists of multiple neurons also called nodes that are structured in layers. Nodes of neighbor layers are connected via edges which have weights to represent their association. A feedforward neural network contains three kinds of nodes: input, hidden and output nodes. Figure 2.3 shows a model of a simple neural network.

Input Nodes

The information from the original source is fed into the nodes of the network that are arranged together in the input layer. First, the input nodes do not need to compute any

parameter values as they feed the information to the hidden nodes of the next layer. Let x_i be the input embedding and y_i be the output embedding.

Hidden Nodes

There is no direct connection between hidden nodes and the source text. First, each word in the source text embeds into the input nodes and then passes the information through the computation from the input nodes to the hidden nodes in the next hidden layer. The hidden layer represents the collection of hidden node forms. There is an indirect connection with the input nodes and the hidden nodes. Feedforward networks must have a single input and output layer and can have no or multiple hidden layers. The computation of a hidden node is called a hidden state. Let a_i^l be the state of *i-th* node in *l-th* hidden layer.

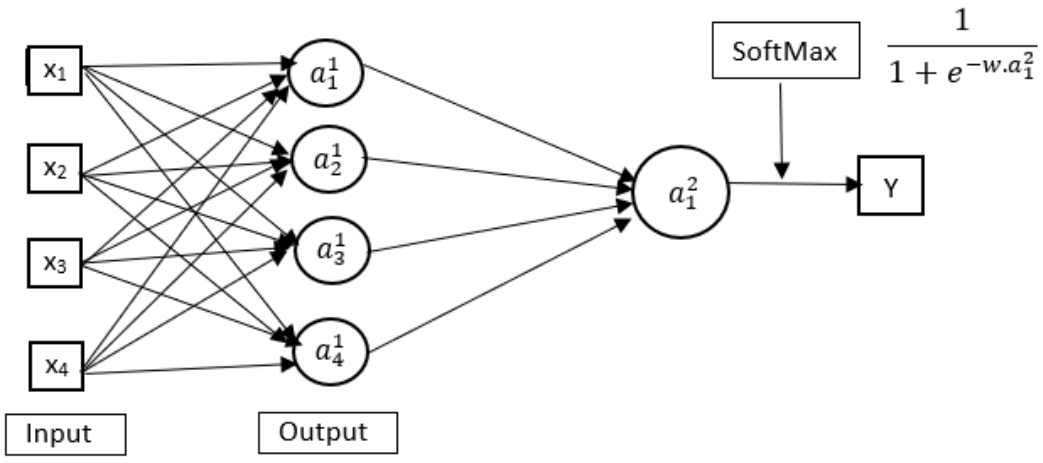


Figure 2.3: Simple neural network model.

Each circle defines a node, and that node has a computation called state '*a*'. An arrow connects the node from the previous layer to the current node using the weight. SoftMax is an activation function which normalizes the weight of each node in each layer.

Output Nodes

The output layer represents the collection of output nodes that carry the information from the network to the outside world through computation. Let a_i^l be the computation of the *i-th* hidden node of the lth layer. We call this computation hidden state. *f ()* is the activation function: sigmoid or SoftMax. The SoftMax function [139] is applied using the following equation.

$$a_1^1 = f(x) = f(w1 * x1 + w2 * x2 + w3 * x3 + w4 * x4) \quad 2.5$$

$$x = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 \quad 2.6$$

$$a1^{(1)} = \frac{1}{1 + e^{-x}} \quad 2.7$$

We use the SoftMax function to normalize the weight of the nodes for each layer which transforms the weight of the nodes of each state into probabilities where the sum of their probabilities is one. This function generates the probability distributions of a set of possible outcomes which is represented by an output vector. In the feedforward network, the information is passed in one direction (forward only) from the input nodes through the hidden nodes (if any) to the output nodes. A multi-layer network means when a feedforwarding network has one or more hidden layers over one input and one output layer. A multi-layer network with a single hidden layer is shown in Figure 2.3. The connections between nodes carry weights which express their strength of association.

Back Propagation and Weight Updating

Initially, all the weights in the network are assigned randomly. The errors are calculated at each of the output nodes for the input nodes and are propagated back through the network using a backpropagation algorithm. The algorithm calculates the total error at the output nodes and feeds the errors back through the network. Then, the gradient descent method is applied to adjust or update all the weights in the network to reduce error at the output layer.

Activation Function

Activation functions introduce non-linearity into the output of a node. This is a very important mechanism in NN that defines the output of that node given an input or set of inputs. In general, this is computed by adding the total sum of the weighted parameters of its input and bias. This decides whether the node is activated or not. In a neural network, we update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions enable the process of back-propagation by passing the gradients along with the error to update the weights and biases. Several activation functions are frequently used in NN as follows:

- Sigmoid function: this function receives a real-valued input and provides the output to a range between 0 and 1. This is calculated using the following equation.

$$\sigma(x) = 1 / (1 + \exp(-x)) \quad (2.8)$$

- Tanh function: this function receives a real-value input and provides the output to the range [-1, 1]. We calculate this using the following formula.

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.9)$$

- SoftMax function: this function is a type of sigmoid function that converts the weight into probabilities where the sum of their probabilities is one. This produces an output vector that represents the probability distributions of a set of potential outcomes.

Learning

Learning is a process to train the model using a given large dataset which automatically learns and improves from the experience/history of the dataset without explicit programming design. Adjusting the weights is the main objective in the learning process in neural networks. There are no cycles in the feedforward network which means the output of the output layer is not fed again to the input nodes. A feedforward network does not consider the previous histories of the process of information and processes the next input independently. Therefore, feedforward networks do not learn the sequences or the temporal dependency between inputs.

2.3.3. Deep Neural Networks

A deep neural network is a neural network with more than one hidden layer. Each node in the hidden layer is connected to nodes of the next hidden layer. The arrow from each node carries a weight property to the next connected node which supervises how much that node affects the activation of the others connected to it. The network is described as deep because of the features of the deep hidden layers and the derivation of its effectiveness from the deep hidden layers.

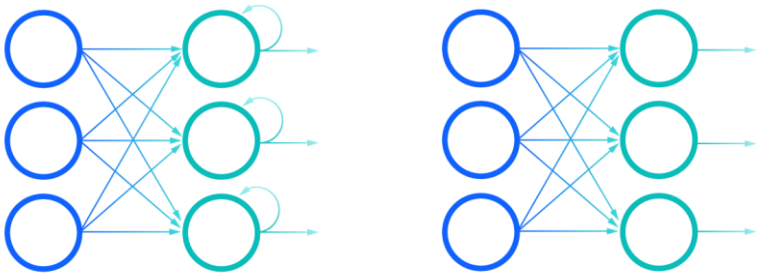


Figure 2.4: Recurrent neural network (left) and feedforward neural networks (right).

2.3.4. Recurrent Neural Network

Recurrent neural networks [74] are introduced to utilize the output obtained through the hidden layers to process future input. In contrast to the feedforward network, RNNs not only propagate the information forward, but they also propagate the information backward from later processing stages to earlier stages. This mechanism allows the network to express the dynamic temporal behavior. RNNs are able to use their internal memories to process the sequences of inputs. This feature enables them to be applied to tasks such as handwriting recognition, translation, speech recognition and so on. Figure 2.4 compares the feedforward network and RNN. RNN not only connects nodes but also keeps their internal memory for future processing.

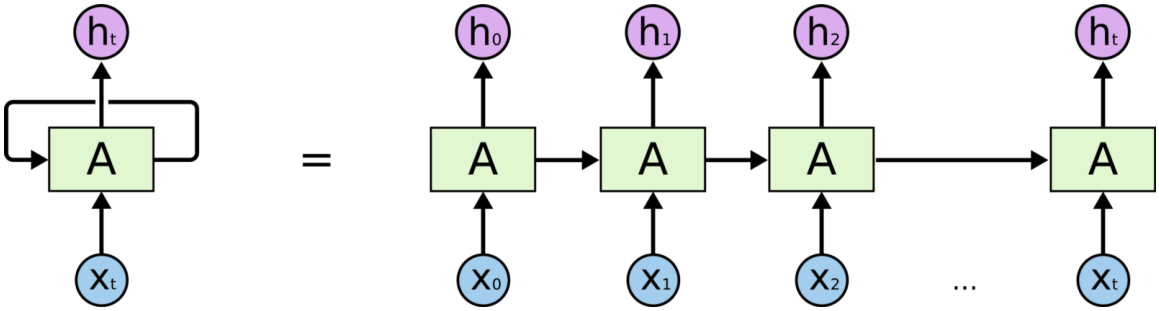


Figure 2.5: Mechanism to keep the previous histories in the current state.

Figure 2.5 shows how the state of a node keeps the memory of the previous history of information processing. RNN keeps the memory of the information processing of the input sequence to the output each time and feeds this information for the next prediction of the input elements. It is able to predict the next target element for a given sentence using the

memory of the previous words processing information. The RNN remembers all these relations among the words in sentences during the training.

2.3.5. Encoder-Decoder Sequence to Sequence Model

A neural network architecture based on a recurrent network is introduced in [77] which comprises two parts: the encoder and decoder. This model learns to encode a variable-length sequence into a fixed-length vector representation and to decode a given fixed-length vector representation back into a variable-length sequence. Figure 2.6 illustrates the architecture of the encoder and decoder sequence-to-sequence model.

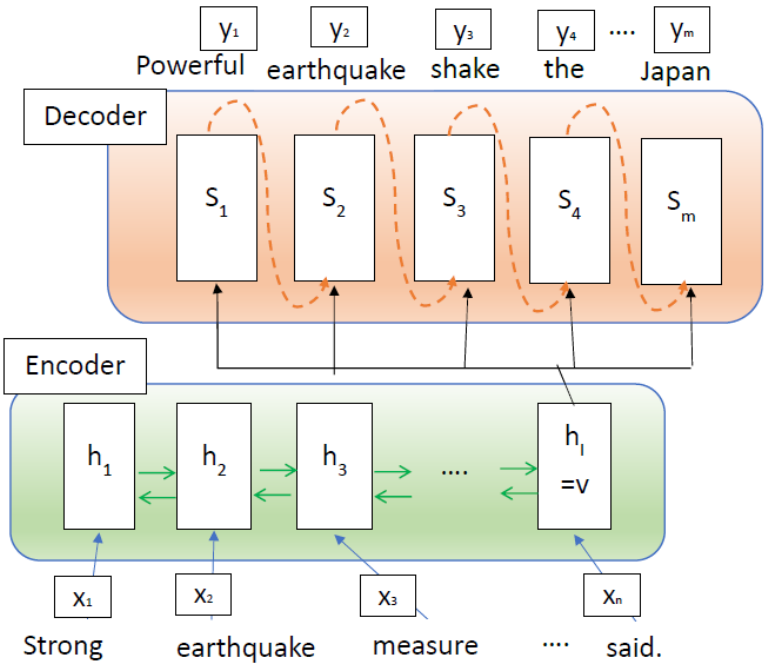


Figure 2.6: Encoder and decoder architecture based neural network.

h defines a state of the encoder part and s defines a state of the decoder part. x= {x_1, x_2, …} are the input elements and y= {y_1, y_2, …} are the output elements. Encoders represent the input sequence and decoders represent the output sequence for the input elements.

Encoder

An encoder presents the input sequence into a vector with a fixed length using RNN. There are different types of units, such as LSTM [35] or GRU for RNN is used in RNN for better performance where each unit receives a single element of the input sequence and

concentrates the information for that element and feeds it forward. The input sequence is a set of words in a sentence or a document. Let x_i represent a word in the sentence at encoder state i. The encoder maps the source sequence to a vector. Let h_i be the current state of an encoder, the previous state of an encoder is h_{i-1}, the weight between the current node and the previous state is W_{hh}, the input state is x_i, and the weight at the input node is W_{xh}. The current state h_i can be calculated using the following equation.

$$h_i = \mathrm{f}(h_{i-1}, x_i) \tag{2.10}$$

$$h_i = \tanh\,(W_{hh}h_{i-1} + W_{xh}h_i) \tag{2.11}$$

$$v = h_i \tag{2.12}$$

where v is the final hidden state at the encoder. This network represents the input of a simple RNN where appropriate weights are applied to the previous hidden state $h_{(i-1)}$ and the input vector x_i. We can see that the last hidden state v is a vector that is produced at the encoder part of the model. This is computed using the formula above. v captures the information for the input elements to help the decoder provide correct predictions. This vector feeds the initial hidden state to the decoder part in the model.

Decoder

A decoder comprises a collection of units of a RNN where each unit predicts an output y_m at a time step m. Each unit receives information on a hidden state from the previous unit and generates its own hidden state and the output. The output sequence is a set of words in summary. Let y_m be an output element or word where m is the position of the element in the summary output. The state of the decoder is computed using the following formula:

$$s_m = \mathrm{g}(s_{m-1}|y_{m-1}, v|) \tag{2.13}$$

As we can see, the previous hidden state is used to compute the next state.

Prediction

The next target element is predicted using the following formula:

$$p(y_1, \ldots, y_m|x_1, \ldots, x_n) = \text{SoftMax}\,(W_{ys}s_m + b) \tag{2.14}$$

where output y_m at time step m and W is the learnable weight parameter.

2.3.6. Attention Model

A potential challenge in this encoder-decoder sequential method is that all the words in the source sentence need to be encoded into the vector with a fixed length. This method faces challenges handling the long sentences in the NN. To address this challenge, an extension to the encoder decoder model is introduced which learns the summary output and the alignments between the input and output elements jointly [78]. This model searches (soft) a set of positions in a source sentence to capture the important information for a generated output element of each state at the decoder. During this process, a context vector is constructed which captures the association of the source positions for the output target element of a current state and all the histories of the generated target words in the previous states at the decoder. This model predicts a target output element based on the context vector. Compared to the basic encoder–decoder sequence model, this model does not require the input sentence to be decoded entirely into a single vector with a fixed-length. Rather, the input sentence is encoded into a sequence of vectors and a subset of the vectors is chosen during the generation of the output element for each state at the decoder. Figure 2.7 shows the architecture of the RNN-based sequential attention model.

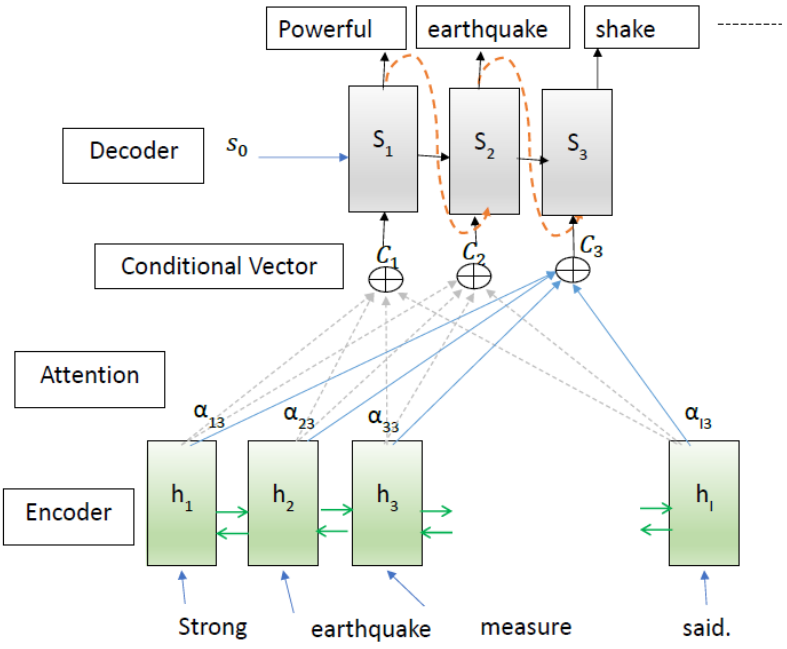


Figure 2.7: Architecture of the attention model.

c= {c_1, c_2,} represent the context vector for each state at the decoder.

Attention Distribution

The weighted parameters are learnt and should find the most relevant encoder positions. The attention of *j* input of the encoder to the *i* state of the decoder, α_{ij} is computed by the following equation.

$$\alpha_{ij} = \frac{\exp\left(sim\left(h_i, s_{j-1}\right)\right.}{\sum_{i=1}^{I} \exp\left(sim(h_i, s_{j-1})\right.} \quad 2.15$$

$$sim\left(h_i, s_j\right) = W^T \tanh\left(w_h h_i + W_s s_j\right) \quad 2.16$$

The encoder states are weighted to obtain the representation relevant to the decoder state.

Context Vector

The context vector $c = \{c_1, c_2, \ldots, c_n\}$ where c_j is computed using the above attention α_{ij} as a weighted sum of these annotations h_i.

$$c_j = \sum_{i=1}^{I} \alpha_{ij} h_i \quad 2.17$$

2.3.7. Convolutional Neural Network

A convolutional neural network (CNN) [75] contains layers based on a feedforward neural network. The potential of a CNN appears from the base block structure called a convolutional layer. CNN consists of a number of layers that are stacked on top of each other. This layer is able to capture the long-range dependencies in large text. The utilization of convolutional layers in a CNN depicts the morphology of part of the human brain visual cortex where a series of layers is processed into an approaching vision and obtains highly complex features.

Convolutional Neural Network Design

The architecture of a CNN consists of multiple layers based on feedforward neural networks built by piling up a large number of hidden layers in a sequence. CNN is able to learn the hierarchical features through this sequential network. The activation layers are dependent on the hidden layers and the pooling layers are dependent on a hidden layer.

Yann [79] proposed the CNN-based model to help understand the fundamental concept of CNN, called LeNet which is able to recognize handwritten characters. Figure 2.8 illustrates the architecture of the CNN model for LeNet.

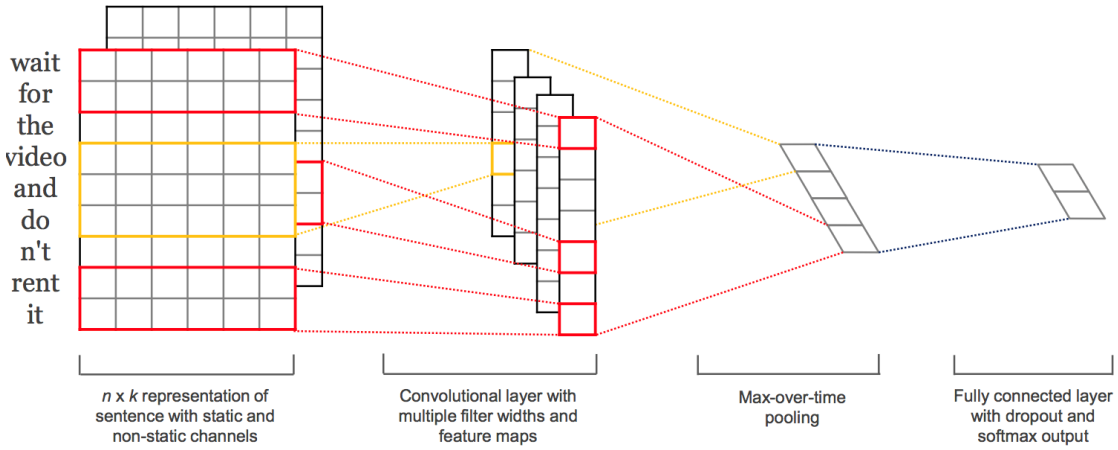


Figure 2.8: Convolutional neural network design architecture

Definition of CNN Elements for ATS

Abstractive text summarization employs convolutional layers to embed input elements, then a one-dimensional CNN, pooling layer and finally, the output layer to predict the output element.

- ***Convolutional Layer***

A convolutional layer is a kind of unit of the convolutional structure to build a CNN. A convolutional layer is presented using convolutional k kernels which probe the text and anticipate the patterns in the sequence of text. When a sequence of the text matches the pattern of a kernel, the kernel is assigned a positive value, and when there is no match, the kernel is assigned a zero or a negative value.

- ***Pooling Layer***

The pooling layer reduces the dimensional size of the representation to decrease the required amount of computation and the parameters in the network.

- ***Fully Connected Layer***

In CNN, the fully connected layer means the inputs of the lower layer are connected to the next top layer. This consists of a series of fully connected layers and represents the hierarchical representation of the text using the fully connected layer. Thus, the output of the fully connected layer represents the high-level dependencies in the large text. The fully connected layer helps to map the representation between the input and the output. Each convolutional layer learns the dependencies of the text to its immediate bottom layer and the fully connected layer learns the high-level dependencies in the text.

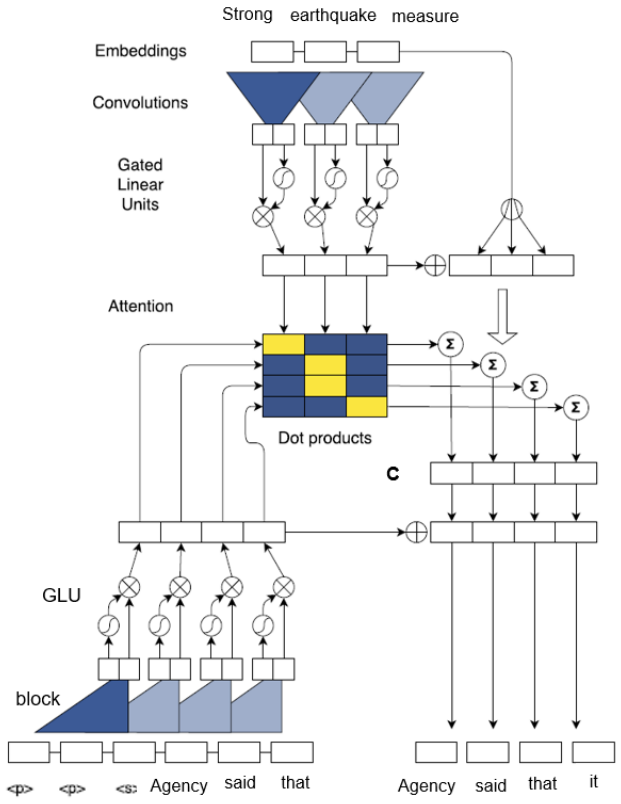


Figure 2.9: Architecture of the convolutional sequence network.

The encoder embeds the source text (top) and measures the attention weight (center). The contextual representation decoder (bottom left) and encoder are used to compute the attentions via their dot product. C is the conditional inputs which are computed using the attention (center right) to the decoder states to estimate the target output element (bottom right). The gated linear units (GLU) contain sigmoid and multiplicative boxes.

2.3.8. Convolutional Sequence-to-Sequence Learning

This is an architecture for sequence modeling based on a convolutional neural network. We call this the convolutional sequence network (CSN) [80]. CSN utilizes the gated linear

units (GLU) and residual connections. Figure 2.9 shows the CSN model. A convolutional neural network (CNN) is used to compute the transitional states of the encoder and decoder.

Position Embeddings

Position embeddings encode the absolute position of each source word within a sentence. Let document d be represented as a sequence of words ($w_1, w_2, \ldots, w_n$) with a total of n words. The input elements are embedded with n words into distributional space $x \in \{x_1, x_2, \ldots, x_n\}$ where $w_j \in R^f$ is the row in an embedding matrix $M \epsilon R^{Vxf}$ (V is the size of the vocabulary) and f is the dimension of a word vector in the embedding matrix. The absolute position of the input elements is embedded in document $p = (p_1, \ldots, p_n)$ to preserve the sequence order where p_i is the position embedding of word w_i at position i in the input sequence. Finally, we represent the input elements along $e = (e_1, e_2, \ldots, e_n)$ by combining word and position embedding, $e=w_1+p_1, \ldots, w_n+p_n$. Similarly, the output elements with m word generated by the decoder are represented along $g = (g_1, \ldots, g_m)$ and are fed to the next step.

Convolutional Structure

A simple layer is shared at the encoder and decoder networks to compute their intermediate states through an absolute size of input elements. Let the output of the *l-th* layer be $d^l = (d^l_1, \ldots, d^l_n)$ and $e^l = (e^l_1, \ldots, e^l_m)$ for the decoder and encoder network respectively. Each layer consists of a convolution unit and a non-linearity. k elements are presented through kernel width k in a single layer of CSN. Each decoder state d^l_i represents the number of input elements in a state by stacking several k-input element layers on top of each other. For example, 5 layers with k = 4 in a decoder state represents 16 input elements. i.e., 16 input elements are dependent on each output. Non-linearities help the CSN to use all the input elements or to attend to specific elements when required. Let $W \in R^{2d \times kd}$ and $b_w \in R^{2d}$ be the parameters, d dimensions, and input $x \in R^{k \times d}$ is an accumulation of the k input elements for each convolutional kernel. Then, all the convolutional kernels are mapped to a single output element $y \in R^{2d}$. The k elements of a layer are operated over successive and precedent layers. Gated linear units are used as non-linearity in CSN to implement a simple gating mechanism over $y = [A\ B] \in R^{2d}$:

$$v([A; B]) = A \otimes \partial(B) \quad (2.18)$$

where the inputs to non-linearity are defined as $A, B \in R^d$, $\otimes$ is the point-wise multiplication. The relevance of inputs A in the current context are supervised via gates ∂ (B). Oord [81] introduced a similar nonlinearity where the tanh function is applied to A, however, Dauphin [82] proved that the context of language modelling achieves better results by utilizing GLU. A residual connection is added from the input of each convolution to the output of the layer to enable the deep convolutional networks.

$$d_i^l = g\left(W^l\left[d_{\frac{i-k}{2}}^{l-1}, \ldots, d_{\frac{i+k}{2}}^{l-1}\right] + b_w^l\right) + d_i^{l-1} \quad (2.19)$$

where g is the function composition operator. Finally, a probability distribution over the k possible next target elements y_{i+1} is computed by providing the top output d_i^L of decoder via a linear layer with weights W_Y and bias b_y to a SoftMax classifier:

$$p(y_{i+1}|y_1, \ldots, y_i x) = softmax(W_Y h_i^L + b_Y) \in R^T \quad (2.20)$$

Multi-step Attention

A separate attention mechanism is used to perform multiple attention "hops" per time step and provide access to previously attended words. The decoder attention is computed using the following formula which combines the current decoder state h^l_i with a previous target element embedding q_i:

$$d_i^l = W_d^l h_i^l + b_d^l + q_i \quad (2.21)$$

Weight matrix $W^l_d \in R^{dxd}$ and bias $b^l_d \in R^d$ are the learnable parameters. Let α^l_{ij} be the attention weight of decoder state i and source input element j and z^{uo}_j the output of the last encoder block u_o. The weights are computed as a dot product of d^l_i and z^{uo}_j, namely,

$$\alpha_{ij}^l = \frac{\exp\left(d_i^l .\ z_j^{u_o}\right)}{\sum_{t=1}^m \exp\left(d_i^l .\ z_t^{u_o}\right)} \quad (2.22)$$

The conditional input $c^l_i \in R^d$ of the current decoder state is calculated as

$$c_i^l = \sum_{j=1}^m \alpha_{ij}^l (z_j^{u_0} + e_j) \quad (2.23)$$

where the embedding of the input elements is e_j which provides order information about a particular input element. When c^l_i is obtained, it is fed to the output of the corresponding decoder state h^l_i and serves as a part of the input to h^{l+1}_i.

2.4. Summary

Text summarization is classified into two approaches: extractive text summarization (ETS) and abstractive text summarization (ATS). Extensive research has been conducted on these two summarization approaches. ETS selects important chunks or phrases of the original sentences using scores which are computed by either linguistic or statistical features. When writing a document summary, humans may include new words, phrase or sentences which are not in the original text. ETS has a limited ability to generate new keywords, phrases and sentences in summaries, so a summary generated by ETS is quite different to one produced by a human. whereas ATS produces a brief version of the document by generating new phrases with words that may not come from the original text. ATS aims to produce short and concise summaries that capture the salient information and overall meaning of the document which helps to generate human-like summaries. ATS generates better results than ETS since an ATS-generated summary is comparatively close to human-written summaries, which makes the summaries more meaningful [142]. A reasonable ATS should maintains the sequence of the main theme and concepts presented in the document, minimize repetition, ensure sentences consistent and coherent, and capture the meaning of the text, even for long documents. Moreover, the generated summary needs to be brief while conveying the salient information of the main text [143]. Therefore, recently, ATS has become a popular research topic and has achieved excellent progress using advanced machine learning techniques. Deep learning approaches were applied in the ATS for the first time in 2015 [144] which was the encoder-decoder architecture-based ATS model. Deep learning approaches have made significant advances in the research on ATS and have been extensively utilized in recent years.

The attention mechanism has been proposed to improve the basic RNN seq2seq [145] and so the application of this attention based RNN seq2seq model to ATS has become standard architecture (Nallapati 2016 [36]; See, Liu, and Manning 2017 [42]; Cohan et al. 2018

[146]; Lin and Ng 2019 [10]). The attention mechanism learns the importance of words at each state of the decoder for a given input sequence at the encoder. In addition, these modes have been utilized to handle the issue of unseen words in the training dataset using a pointer generator network [36] [42]. Moreover, See, Liu, and Manning [42] introduce an attention with a coverage mechanism to avoid the duplication of words in the summary. Also, Lin et al. (2018) [117] propose an extensive encoding model to resolve the issue of copying. The encoder-decoder architecture based on a transformer using an attention mechanism and a model for extracting salient information to create summaries, is introduced in Devlin et al. (2019) [43]. Recently, the transformer-based pretrained model and the transformer itself have become dominant approaches in ATS because these models incorporate self-attention to reduce the computational cost by parallelizing the computations in the training step (Vaswani et al. 2017 [39]; Zhang, Xu, and Wang 2019 [147]; Devlin 2019, [42], Xu et al. 2020 [134]; Pilault et al. 2020 [30]). The limitation of these architectures is that the maximum-likelihood loss function is minimized whereas the evaluation of ATS is mostly ROUGES metrics-based.

In general, several ATS approaches have been presented which evolved with different architectures, such as the attention mechanism, transformer-based, reinforcement learning (RL) and sequential learning. Moreover, evaluations have been performed regarding processing, embedding, validation and training. However, there are a limited number of approaches which are able to obtain knowledge of the document to understand the document and identify topic information using that knowledge. Shi et al. [116] surveyed various ATS models, which are based on convolutional and RNN sequence-to-sequence encoder-decoder architecture, the main purpose being to examine network infrastructure, training techniques and the algorithms utilized to generate a summary. The basic form of the RNN-based ATS model suffers due to dropping the dependencies of the long-range sequence called gradient vanish, whereas LSTM-based ATS models are able to resolve this problem by utilizing the gate mechanism to learn which information to forget from the history and which new information to remember from the current state. But the RNN-based ATS has limitations in parallel computation since each state is required to wait for the previous state's computation. Furthermore, RNN-based models are still constrained when handling very long sequences (length > 200) Compared to RNN, the infrastructure of the

CNN enables the ATS model to explore the hierarchical structure in sequences more easily. Convolutional networks do not depend on the computations of the previous time step. Consequently, this allows parallelization over every element in a sequence and has a shorter path to capture long-range dependencies.

However, the weakness of these approaches is that they do not utilize high-level attention to capture the salient information of the document which makes it difficult to generate an abstractive summary which is near to a human-written summary. There are a limited number of approaches to capture topic information using a high-level attention mechanism [3] [113]. Generating abstractive summaries requires factual knowledge to understand the document and salient semantic information to capture the topic information. Advanced natural language processing (NLP) can help to understand the content while extracting the background information from the knowledge base. But the aforementioned models do not consider the important background knowledge of the document and may capture irrelevant information as topic information in the generated summaries.

In this chapter, we discussed the works related to our proposed system, the tools which are useful for implementing our work, and the recent progress in the field of knowledge extraction, topic modeling and abstractive text summarization. In knowledge representation, we first explain the reason for focusing on natural language representation, then we define the machine-readable knowledge base used to represent the knowledge. After this, we discuss some widely known effective knowledge base systems such as DBpedia, ConceptNet and so on. Next, we describe the functionality of natural language processing and the contribution of NLP tools to natural language representation. We also discuss the SPARQL query to retrieve information from an ontology-based knowledge base. Finally, we present our previous work, the NLIKR scheme, which we utilize to build our OMRKBS. We also explain the classic LDA model which has recently been proven effective for use as a base model to improve the topic model. We use the LDA model to improve our topic model using background knowledge. We describe the preprocessing of text, the mechanisms, and the algorithms of the LDA model. We discuss the Gibbs sampling technique to sample, assign and compute the information of a topic. Finally, we provide examples of the LDA technique to illustrate the model. We describe some

fundamental information and the background of word embedding, neural networks, and the deep learning architecture. Finally, we discuss the recent developments in the ATS approach, such as recurrent neural networks, the encoder-decoder sequential-to-sequence network and the attention-based ATS approach, highlighting their effectiveness and drawbacks. We detail the fundamentals of convolutional neural networks which is related our work. We discuss the background of convolutional sequence networks since we use this as the base model to improve the summarization model.

Chapter 3.

Problem Definitions

Although there has been significant progress in deep learning-based ATS models in relation to summarizing documents, it is still challenging to produce human-like and high-quality abstractive summaries since machines have a limited ability to understand the meaning of content and a limited capability to highlight topic concepts while generating summaries. Moreover, the generated summaries often do not have an appropriate syntactic or semantic structure. Therefore, developing an automatic text summarization system by resolving these challenges to generate human-like summaries is the target of this thesis. The problem to be addressed is how can a machine interpret and understand a document, identify the topic using background knowledge and generate summaries by associating this relevant topic information while maintaining a syntactic and semantic structure that is close to human-written summaries. To address this problem, we present a complete deep learning-based abstractive text summarization system called Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation (KTSNR) which is able to produce coherent and meaningful summaries similar to the ones written by human beings.

3.1. Building the KTSNR System

We built a system using a three-step approach as follows: i) we construct an ontology-based machine-readable knowledge base system (OMRKBS) to provide semantic and informative background knowledge about text that helps the system to understand the text; ii) we construct a topic knowledge base (TKB) to provide topic information based on the knowledge background of the source text (which is retrieved from the first step) to the system so that it can learn the salient and relevant information of the source text. We refer to this topic information as knowledge-powered topic information; iii) we develop a convolutional sequence network-based text summarization model with high-level topic attention that incorporates the knowledge-powered topic information (called KTOPAS) to

produce coherent, concise and human-like summaries with word diversity. Each approach will be explained step-by-step in the following sections.

3.1.1. OMRKBS

This is a knowledge base system which enables the system to understand the text by providing the background knowledge of the text. We propose a framework to construct an OMRKBS that comprises extraction, preprocessing, and mapping processes. Extraction retrieves the external information or background knowledge in a textual data format from various trusted sources such as DBpedia [49], ConceptNet [50] and WordNet [66]. Preprocessing transforms the textual data into meaningful, informative, structural, and individual features so that the system can interpret and read the information. We call these features rich structured information (RSI). Obtaining RSI is still a challenging task while constructing OMRKBS. We propose algorithms and rules to handle these challenges. Finally, the last step is mapping the process-built RSI into the OMRKBS system. One of the issues in constructing OMRKBS is to map the RSI in OMRKBS in a way that machines can read the information from OMRKBS. The mapping process utilizes a Natural Language Independent Knowledge Representation (NLIKR) scheme to map the RSI of human knowledge in OMRKBS. This scheme represents a word as a concept and define a concept by its relations with other concepts. To map each word as a concept and the relationships among concepts in the RSI, we develop formulas to discover concepts and their relationship in OMRKBS. After discovering the concepts and relations in RSI, we propose an algorithm to map the RSI using the concepts and their relations in OMRKBS. Since each word is mapped as a concept and RSI is defined by relating their concepts in RSI, machines can read each concept and interpret their relations in a straightforward manner. Therefore, OMRKBS is capable of generating machine-readable information about a term.

3.1.2. TKB

This is a knowledge base which provides knowledge-powered topic information based on the source text. We propose a framework to construct a topic knowledge base that

comprises conceptualization, a knowledge-powered topic model (KPTopicM) and learning the KPTopicM. The conceptualization algorithm is proposed to derive the concept distribution for each word in the text. We retrieve the background knowledge or concepts using OMRKBS [83], ConceptNet [50] and Probase [51]. KPTopicM is a topic model which incorporates background knowledge to determine topic information using the distribution information. This model interrelates the topic information and background knowledge of the document to handle the challenges in obtaining relevant and salient information due to the lack of background knowledge. Classic LDA is three-layer hierarchical model where topic information is associated with words directly but does not consider background knowledge. The KPTopicM model includes one extra layer compared to the LDA statistical topic model which allows the direct association of background knowledge or concepts and the indirect association of words in the topic information. Therefore, this model is able to generate coherent and informative salient information. Finally, we train the KP-Topic model using the dataset and use the learned data as TKB.

3.1.3. KTOPAS

KTOPAS is a convolutional sequence network-based text summarization model which incorporates knowledge-powered topic information to generate abstractive and coherent summaries. This model comprises three CSNs: word, knowledge, and topic-level CSNs, a tri-attention channel, and final probability generation and learning. The word and topic-level CSN encoders associate input elements and knowledge-powered topic elements with the summary elements that predict whether the summary element captures the input element or topic element. The knowledge-level CSN encoder associates elements with decoded knowledge-powered topic elements to obtain the coherence of topic information in relation to the source text. The tri-attention channel first measures the attentions from three aspects of the CSN level and then combines them using the SoftMax function to jointly learn the attention from the three aspects of CSN. The final probability generation produces the probability distributions to predict the next target element in the output summary at the decoder of the word and topic-level CSN. Finally, the KTOPAS model is learned using the mixed training objective function [45] to maximize the model. The KTOPAS model handles one of the major challenges in generating coherent, relevant, and

meaningful concise summaries due to the gap in providing salient background knowledge to the ATS model.

3.2. Generating Summaries using KTSNR

Once the KTSNR is constructed, it performs the following tasks while generating summaries or training the KTSNR. First, KTSNR preprocesses the source text using natural language processing and retrieves the background knowledge from the knowledge base system. Then, it retrieves the background knowledge from the knowledge bases, such as OMRKBS, ConceptNet or Probase. Next, we obtain the topic information using the background knowledge from TKB. Finally, we incorporate the topic information into the KTOPAS model. Figure 3.1 illustrates the complete system of the deep learning-based text summarization model.

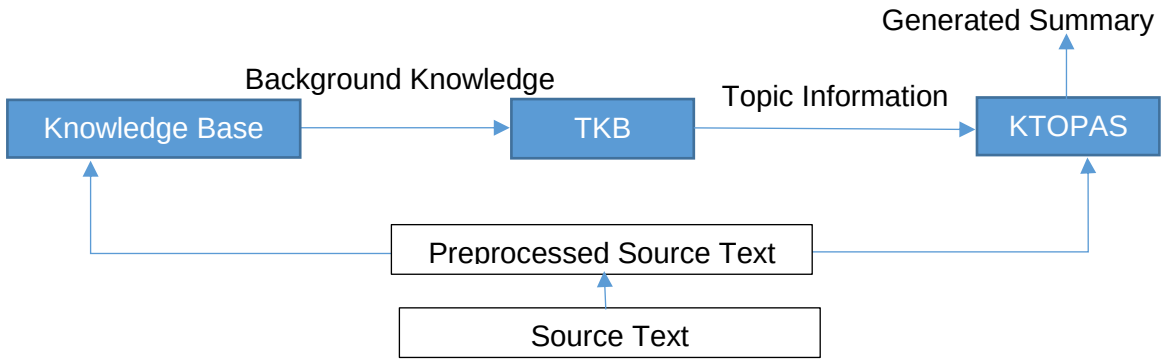


Figure 3.1: Architecture of the KTSNR System.

KTSNR is our proposed CSN based ATS system which use a joint knowledge-based topic level and natural language representation. OMRKBS is ontology-based knowledge-based system, TKB is a repository of learned topic model that serve as knowledge base. KTOPAS is CSN based text summarization model to generate summary. a→b indicates that a is the input and b is the output of a process. Each blue rectangle defines a process.

This system resolves the challenges of generating summaries in a human-like manner, that is understand the content, identify the topic information, and generate summaries. The KTOPAS model is able to generate abstractive, coherent and human-like summaries with large word diversity without losing the original intent of the article. This system assists users to learn the essence of the original document without reading the entire article which saves the user a lot of time and effort. This system helps the search engine to find the

required information from the summaries instead of the source document in less time. This is an effective system which is able to generate important information from the original text and produce a significantly shorter version than the original text. We can use this summarization system for massive information in the areas of health, finance, research, new articles etc., to obtain topics, search engines, business analysis, and market reviews of products and services.

Chapter 4.

Comprehensive Structured Knowledge Base System Construction with Natural Language Presentation

Constructing an ontology-based machine-readable knowledge base system from different sources with minimum human intervention, also known as ontology-based machine-readable knowledge base construction (OMRKBC), has been a long-term outstanding problem. One of the issues is how to build a large-scale OMRKBC process with appropriate structural information. To address this issue, we propose Natural Language Independent Knowledge Representation (NLIKR), a method which regards each word as a concept which should be defined by its relations with other concepts. Using NLIKR, we propose a framework for the OMRKBC process to automatically develop a comprehensive ontology-based machine-readable knowledge base system (OMRKBS) using well-built structural information. Firstly, as part of this framework, we propose formulas to discover concepts and their relations in the OMRKBS. Secondly, the challenges in obtaining rich structured information are resolved through the development of algorithms and rules. Finally, rich structured information is built in the OMRKBS. OMRKBC allows the efficient search of words and supports word queries with a specific attribute. We conduct experiments and analyze the results of relational information extraction, with the results showing that OMRKBS had an accuracy of 84% which was higher than the other knowledge base systems, namely ConceptNet, DBpedia and WordNet.

4.1. Introduction

Machine readable knowledge bases are used to store datasets so that these datasets can be accessible through systems. Machine-readable knowledge base construction involves the automated extraction and integration of data from different sources and generating meaningful information with interoperable knowledge [49]. There is a large body of research on the automatic extraction of information for MRKBC. Initially, the research focused on syntactic information extraction [61][84], but more recently, the extraction of

lexical semantic information has received more interest from the research community [62-63]. Knowledge base systems (KBS) which use traditional databases are not effective due to the limited operational and analytical workload and latency for retrieval [64]. On the other hand, ontologies which provide descriptions of terms important to a specific domain [63] are often used as a resource and have become an alternative to KBS in applications where elements are defined using the relations between concepts [69]. The mechanism of building an ontology-based machine-readable knowledge base system, also known as ontology-based machine-readable knowledge base construction (OMRKBC) is gaining more attention from the research community. While developing this process, most research studies include defining the ontological elements in a machine-readable way [64-65], providing descriptions of concepts using the relations between concepts [4] [60] and a more enriched meaning [52][68]. The attributes or descriptions are from sources that are publicly available but are difficult to obtain and structure into a single KBS [65]. There are several publicly available knowledge bases that are extremely reliable and commonly used such as DBpedia [49], ConceptNet [50], FrameNet [53] and WordNet [66]. Reusing these reliable knowledge bases is one way to facilitate the assignment of meaning to the terms of a domain [85]. However, the construction of ontologies is time-consuming and requires a thorough knowledge of the domain [86]. Furthermore, building an appropriate structure that represents information about terms is not a trivial task [87]. Additionally, some approaches are restricted to a single domain, hence they are not applicable to other domains.

The main objective of OMRKBC is to obtain knowledge about each term from different sources through appropriate structured information and by representing the information to be queried in a meaningful and logical way. When terms and definitions are mapped to an ontology, they are often richly structured with different relations, attributes and simple relationships between concepts. Well-structured information or definitions support the efficient access of data from our OMRKBS which returns meaningful results. Before such a system can be used, an ontology needs to be created based on the existing data. For this purpose, first, we manually build a base ontology from two sources: BioPortal [47] and CRISP [48]. Then, we automatically build OMRKBS based on this base ontology from three reliable KBS: DBpedia [49], ConceptNet [50] and Word-Net [66]. This research

focuses on automating OMRKBC to obtain high-quality data and increase its effectiveness. We present a method to obtain a base ontology with important concepts. Once the important concepts are established in a base ontology, they can be used to define more complex concepts automatically from sources.

More broadly, with purpose of representing the knowledge as machine interpretable individual feature, this section proposes Natural Language Independent Knowledge Representation (NLIKR), a scheme for an ontology based KBS. This scheme represents each English word as a concept in KBS. A word or concept is defined by its properties (i.e., its relationships with other concepts). The characteristics of a concept are indicated by its relationship with other concepts. As a result, a concept definition can go beyond human language since every word is a concept and is defined by another concept. For example, 'earthquake shake the surface' is one feature of earthquake, the association between 'earthquake' and other concepts (i.e., 'shake', 'surface') in the feature represents the properties of 'earthquake' such as <earthquake, shake, surface>. In the associations, 'shake', 'surface' are all concepts. A concept inherits the properties of its super concepts. For instance, 'earthquake' is a sub-concept of natural disaster. Therefore, earthquake shares the characteristics 'natural disaster'. Therefore, earthquake shares the characteristics of natural disaster, such as 'causes great damage or loss of life'. Our research develops a program for an ontology based KBS where the definitions or features of concepts are structured so they can be entered into the ontology using the NLIKR scheme.

We observe that the process of OMRKBC is iterative: enriching the knowledge base by importing concepts, instances and relations and defining concepts from various sources. This motivated us to develop a program to automatically import data from different sources. In one part of the program, we import instance datasets which are available in CSV format from DBpedia. However, several problematic issues were identified while importing CSV instances into an ontology, such as its time-consuming nature and it consumes a large amount of space. We propose several algorithms and techniques to resolve these issues. The program performs the following operation to import instances by resolving the issues. First, a pre-processing algorithm is executed to process the large data file of instances and then a mapping algorithm is executed to automatically create the

mapping expression to embed the instances in OMRKBS. Finally, a program loads the instances with a mapping expression and embeds the instances in the system using OWL API. In the other part, we import a definition for each concept in OMRKBS. First, this program pre-processes a definition to turn the long text into features using the OpenIE [70] and some rules. Then, the program discovers each word in the text as a concept in the system and creates a mapping expression to embed the features. Finally, the features are implanted using a mapping expression in OMRKBS.

Our primary improvements to the program are defining each concept with a description, features and instances through appropriately structured information. These features and instances of concepts are richly structured due to the advantages obtained by using NLIKR. This advantage implies that the features of a concept are structured in such way that each word in the structure of a feature is a concept and all concepts in the structure are linked as stated in the feature. We identified individual or unique features from the definition. Then, we embedded each feature in the system by its interrelationships with other concepts, relations and attributes as these features would be inherited to subclasses of the concept and the concept itself. These individual features with relations and/or attributes that are embedded in the system are called rich structured information (RSI). Consequently, each feature is machine interpretable since machines can discover each concept and find the interrelationship of concepts through the structure of features. Next, we concentrated on the retrieval and presentation of information of the concept being queried using simple SPARQL [71] queries which is mentioned next section.

Our major contributions are as follows:

A. We propose Natural Language Independent Knowledge Representation (NLIKR), a method which regards each word as a concept which should be defined by its relations with other concepts to represent the information or knowledge as machine interpretable features.
B. Using NLIKR, we propose a framework for the OMRKBC process to automatically develop a comprehensive ontology-based machine-readable knowledge base system (OMRKBS) using well-built structural information to provide machine

interpretable, individual, meaningful, and salient features with a diverse range of vocabulary.

C. We extract useful and common-sense information and relations from knowledge bases such as DBpedia and ConceptNet, transform this into rich structured information (RSI) based on the NLIKR method and incorporate RSI into the OMRKBS. OMRKBS is able to effectively search for a term and queries a term with a specific attribute.
D. We present formulas and rules to discover concepts and their relations to documents and propose mapping algorithms to obtain RSI in the OMRKBS. RSI are the features of a concept where each feature is structured by associating concepts through relations and attributes which are machine interpretable.
E. We present a SPARQL query to retrieve information about a term efficiently from our knowledge base OMRKBS. These queries retrieve information on a concept which shows the features not only of a concept class or instance (such 'Barak Obama') but also super classes of the concept or instances (such as 'president') from OMRKBS which relate to the concept.
F. We also present a process query to allow users to ask a question about a word which has a specific property or attribute using two keywords.
G. We evaluate the proposed OMRKBS, and the experiment results show that OMRKBS achieves better an accuracy (84%) than the other KBS, namely ConceptNet, DBpedia and WordNet.

4.2. Related Work

Recently, several approaches that reuse existing knowledge bases to automate ontology construction from unstructured text have been proposed [89-91]. The drawbacks of these approaches include labor costs to construct the dictionary, its domain-specific nature, and the limited number of patterns. Several approaches to ontology-based knowledge bases have been proposed to reformulate knowledge representation in ontologies [92-95]. However, semantic searches in knowledge bases still face difficulties, such as the lack of a detailed methodology that guides the ontology learning process from text. Portage [96]

supports plugins to import datasets from various sources to construct an ontology, however they are costly to assemble, and continuous human effort is needed to keep them up to date.

Automatically constructing a KBS from sources is an important and challenging task. A large body of research exists on automatically obtaining large and quality (but textual) information from Wikipedia. The DBpedia [49] extracts structured information from Wikipedia covering many specific domains and general world knowledge [53]. But the extracted knowledge is mostly limited to named entities or concepts with proper names, such as cities, persons, species, movies, organizations etc. The linguistic relation between such concepts that are more relevant for ontology mappings is absent in DBpedia. YAGO [97] is identical to DBpedia in that each article in Wikipedia becomes an entity in YAGO. YAGO mainly extracts a smaller number of relations between concepts. Nevertheless, YAGO does not interrelate concepts if WordNet does not contain the concepts. BabelNet [68] is similar projects that collect crowd-sourced knowledge from similar sources. In these KBS, a large, structured, multilingual taxonomy is created from a combination of Wikipedia's structured knowledge and WordNet [66]. However, a large amount of information is still being hidden in the text of the Wikipedia articles which is not covered in DBpedia, YAGO or BabelNet. The automatic extraction of semantic concept relations from raw text in KBC, even for concepts that are not yet listed in an existing repository such as WordNet, is a still challenging issue.

Numerous research efforts aim at extracting knowledge from text corpora but research on the exact purpose of common sense knowledge (common sense knowledge presents facts or individual features about the concept, such as 'Lemons are sour') which is machine-readable, is comparatively rare [98]. Automatically inferring missing facts from existing ones has thus become an increasingly important task. Cyc [99] is an AI platform with human reasoning, knowledge, and logic on an enterprise scale. To reason about text using Cyc, mapping the text into its proprietary logical representation is required using its own language Cyc. However, this mapping process is quite complex because the inherent ambiguity in natural language must be resolved to produce the unambiguous logical formulation required by Cyc. Wordnet [66] is an original and prominent linguistic resource. Words can point to one or several synsets and synsets can be referenced by one or several

words in WordNet. However, WordNet focuses on the formal taxonomies of words. In contrast, ConceptNet [50] which has been created from reliable sources, is a freely available large-scale commonsense knowledge base that focuses on a richer set of semantic relations between compound concepts and supports many practical textual reasoning tasks over real-world documents. ConceptNet can best be seen as a semantic resource whose scope of contents is general world knowledge in the same vein as Cyc.

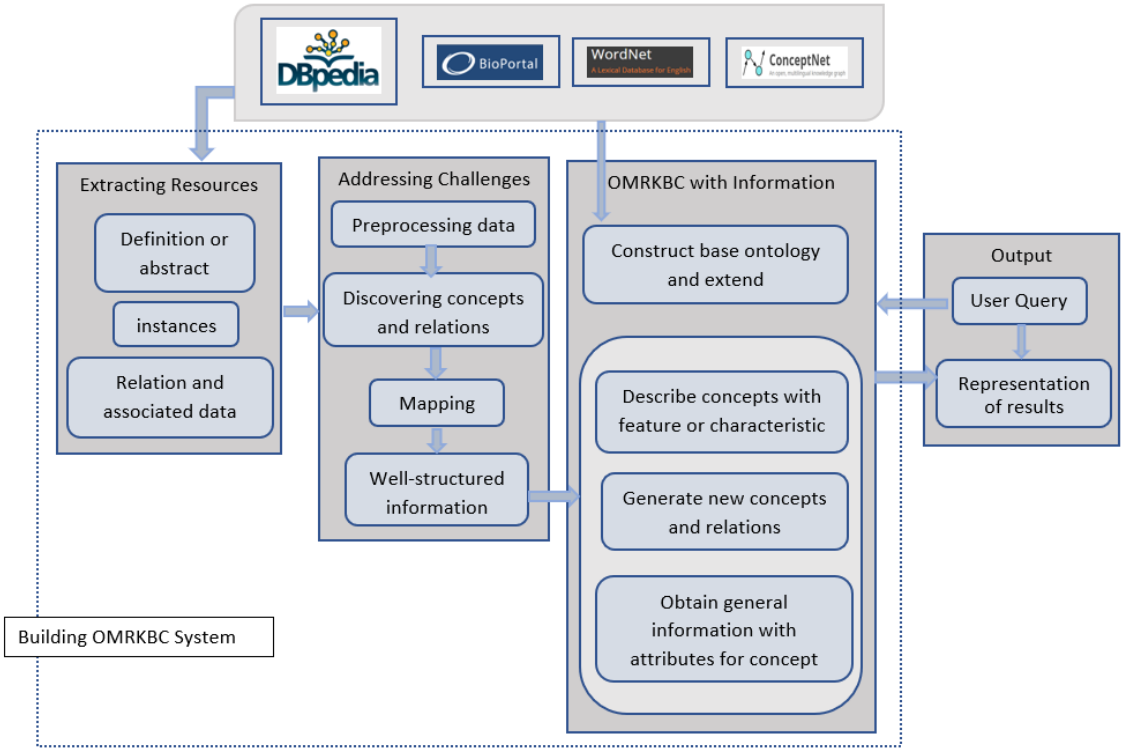


Figure 4.1: The proposed framework of OMRKBC.

These KBs store common-sense facts in a machine-processable way and more recent work puts a focus on human interaction such as building question answering systems [88] [100]. However, facts can exhibit their properties in multiple aspects and fact expression has lost some properties or attributes through these KBSs. Moreover, not all the words in fact expression are interrelated in these KBSs, rather they present as a whole statement in KBSs. Therefore, these KBSs are not fully machine interpretable. Currently, knowledge base construction solutions have focused on obtaining rich structured information from text [101-106]. These KBCs already support a broad range of downstream applications such as information retrieval, question answering and medical diagnosis. However, the essence of

the information (individual features) remains latent in knowledge representation, where relations and attributes are expressed via combinations of textual and structural information. Moreover, the entities extracted by these systems have not been integrated into a single homogenous ontology. In this chapter, we design an OMRKBC process that defines concepts automatically with definitions and instances from reliable sources to build a comprehensive OMRKBS. Our approach acknowledges the facility of three reliable KBSs: Dapedia, ConcpetNet and WordNet and integrates various types of knowledge such as features and instances from these resources into OMRKBS through rich structured information that helps to define the object from various perspectives. Concepts are linked with attributes and relations in the rich structured information. The features of the concepts are built through rich structured information so that the system can return logical, meaningful, and informative results to the user's query. We construct an ontology as a whole KBS, not as a domain, which facilitates the process of defining words and represents the query data in an informative way.

4.3. Definition of NLIKR

We propose NLIKR scheme where each existence is a concept. For instance, 'water,' 'liquid,' 'president,' 'politician' and 'war' are all concepts. The set of all concepts is the CS which is a huge hierarchical structure formed by concepts being bound in two types of relations: inheritance and association.

Definition 1. In NLIKR, each existence (physical or abstract) is a concept (denoted as c). All concepts form a set. The set is named *CS*. $CS = \{c_1, c_2, \ldots, c_n\}$ (n is a finite integer and $n > 0$). Each concept is an element of the CS. The CS has a finite number of elements.

Definition 2. Let c_1 and c_2 be two concepts in *CS*. c_1 is defined as a sub-concept (descendant) of c_2 (denoted as $c_1 \subseteq c_2$) if c1 is a type of c_2, in which case, c_2 is called a super-concept (ancestor) of c_1. For instance, 'water' is a sub-concept (descendant) of 'liquid' and 'drink.' 'liquid' is the super-concept (ancestor) of 'water' and 'drink'. A concept inherits properties of its super concepts. The *CS* is unique in terms of its hierarchical structure. The inclusion operator $\subseteq$ is transitive. That is, for concepts c_1, c_2 and

$c \in CS$, $c_1 \subseteq c_2$ and $c_2 \subseteq c$ infers $c_1 \subseteq c$. Being a sub-concept of c_2, c_1 possesses (inherits) properties/characteristics of c_2. This means that ∀d and $e \in CS$ if $c_1 \subseteq c_2$ and there is an association then the association also exists. Therefore 'water' exhibits characteristics of 'liquid.' Suppose the definition of liquid is "liquid has no shape": <liquid, no, shape>. Since water is subclass of liquid, water will have the characteristic as well such as <water, no, shape>. Also, it is the relations between 'water' and 'liquid' is 'transparent' (in color) and 'tasteless' (in taste), etc., define characteristics of 'water.' As a result, the definition of 'water' can be expressed as a set of relations without the involvement of a human language. For example, <water, no, shape><water, color, transparent><water, no, taste>. It is obvious that ⊆ is an order on *CS*, hence is an ordered set [15]. We will now closely examine the structure of the *CS* and the two types of relations: inheritance and association.

4.3.1. Inheritance creation of the hierarchical structure

Inheritance reflects the "…is a…" relation. It refers to the phenomenon that an association possessed by the super-concept is also possessed by the sub-concept. Inheritance can be multi-dimensional. This refers to the fact that a concept can be divided in different ways. For instance, the 'liquid' concept can have sub-concepts such as 'water,', 'milk', 'blood,' 'gasoline,' 'wine' and 'urine' which divide the 'water' concept based on taste or element. It can also have the sub-concepts 'drink' and 'non drink' which split the concept into two distinctive categories.

4.3.2. Association Establishment of Properties

An association between concepts reflects a property/characteristic of the concepts. For instance, 'water' is associated with 'transparent' through 'color.' The triple <water, color, transparent> forms an association and this association reflects a characteristic of 'water.' Of course, it also reveals a characteristic of 'color' and 'transparent.' Each concept may have associations with millions of concepts. These associations describe the concept and establish the properties of the concept. For example, '<water, color, transparent>' and '<water, taste, tasteless>' define the physical properties of 'water.' As a result, an

application can refer to these associations when processing a text that contains the string ‘water.’ Consequently, the application’s understanding of ‘water’ is far beyond the simple string W-A-T-E-R. Not only is the application aware of the physical and chemical properties of ‘water,’ it also possesses information about the sources, usages and applications of the concept. Ultimately, we let concepts describe/define each other in a machine language

4.3.3. End Concepts

At this stage, a question arises: if we expect concepts to define each other, in which way can precise values be obtained? For instance, the association ‘’ defines the color of water, which is ‘transparent.’ But precisely, what is ‘transparent’? In which way should ‘transparent’ be represented? ‘transparent’ as a color can be represented precisely as a triple (0, 0, 0) in the CS. Each number represents the level of ‘red,’ ‘green’ or ‘blue’ components. ‘transparent’ is named an end concept.

Definition 3. A concept that can be defined by a set of values and that does not have any sub concept is an end concept. Also referred to as a terminal concept in ontology, an end concept can be precisely defined by a value or a sequence of values. End concepts play a vital role in concept definition. This is because by simply letting concepts define concepts without any quantified information, the definitions may become a set of definition loops. Such definitions may not be valuable.

4.3.4. Abstract Concepts

A concept can be abstract. An abstract concept does not have a model in the real world. Instead, it represents a collection of concepts. In *CS*, an abstract concept normally acts as a placeholder to represent a type of concepts. For example, ‘taste’ is an abstract concept. taste is not an object in the real world. Meanwhile, the existence of ‘taste’ in *CS* is important. It acts as the parent of edible things, and it has its own characteristics. The question now becomes, how many concepts and how many associations should be included in *CS*? The answer is all concepts, and their associations should be included. Only in this way, human knowledge about things can be completely translated into machine knowledge.

4.4. The Framework of OMRKBC

We propose a framework to build the OMRKBC process efficiently. First, we create the base ontology manually from existing ontologies such as CRISP [48]. Then, we extract information about concepts from DBpedia, WordNet and ConcpetNet and design a program to build the OMRKBC system with this information. A description of how the OMRKBC system is built is given in "Building the OMRKBC system" section and we represent the search results using efficient queries from our KBS in "System output" section. Figure 4.1 shows an overview of the framework of OMRKBC. The main purpose of OMRKBC is to define the concepts of the base ontology automatically from various types of structured information such as descriptions, instances, and relations. In Dbpedia and ConceptNet, such information is available in CSV format. We extract information from these large sources and turn this information into a knowledge base. For this, we propose a program to build the OMRKBC process in three phases: extracting resources, addressing the challenges, and embedding information in OMRKBS. Each phase is defined as follows.

4.4.1. Extracting Resources

The abstract (DBpedia provides a short abstract for each article, and we used this abstract as definition in OMRKBC) and instances corresponding to concepts are extracted from DBpedia. Also, we extract relations and their corresponding data associated with concepts from ConceptNet. Some descriptions of concepts are extracted from CRISP.

4.4.2. Addressing the Challenges

In this process, the abstract or a description of a concept is turned into a set of individual features, and instances are converted into general information with attributes. We call this rich structured information. We focus on three challenges in relation to processing the information into RSI (rich structured information). Firstly, data must be pre-processed before being converting into RSI. Then, each word is discovered or allocated as a concept

and possible groups words/ phrases are discovered as relations in the OMRKBC system. Thirdly, data are mapped to convert into RSI. Finally, the well-structured information is ready to be entered into the ontology. Table 4.1 show the example of the structural information from the knowledge base for the concept 'Bara Obama'.

Table 4.1: Examples of rich structured information for various knowledges of a concepts.

Example 1: Take as an example the word ''water' which is described as follows: "H2O, tasteless, colorless, odorless compound present in all tissues, and the most universal of solvents. The density of water is 1". Structured information input: '<water, H20><water, no, taste><water, no, color><water, no, odor><water, compound, present in, organic, tissue><water, solvent><water, density, 1>
Example 2: An earthquake (also known as a quake, tremor or temblor) is the shaking of the surface of the Earth resulting from a sudden release of energy. Structured information input: <earthquake, known as, quake><earthquake, known as, tremor ><earthquake, shake, surface><earthquake, resulting from, sudden release of energy
Example 3: Barack Obama is an American politician who is the 44th President of the United States. He is the first African American president born in Honolulu, Hawaii. *Structured information input:* <Barack Obama, American politician><Barack Obama, first African American president> < Barack Obama, born in, Hawaii><Barack Obama, 44th president of the United States>
Example 4: Barack Obama's birthplace and spouse name are USA and Michelle Robinson respectively. *Structured information input:* <Barack Obama, birthplace, USA> < Barack Obama, spouse name, Michelle Robinson>

The birthplace and spouse name are referred to as attributes. This information is imported through the following structure. We found some individual features of concept from this sentence of corresponding examples.

4.4.3. OMRKBC with Information

We design a program to build the RSI in OMRKBC. Individual features or characteristics of concepts and general information on concepts associated with attributes are embedded in RSI. Therefore, rich structured individual features and general information with attributes are built in OMRKBC. After importing the short abstract or the description, the ontology is enriched with new concepts, relations, or attributes.

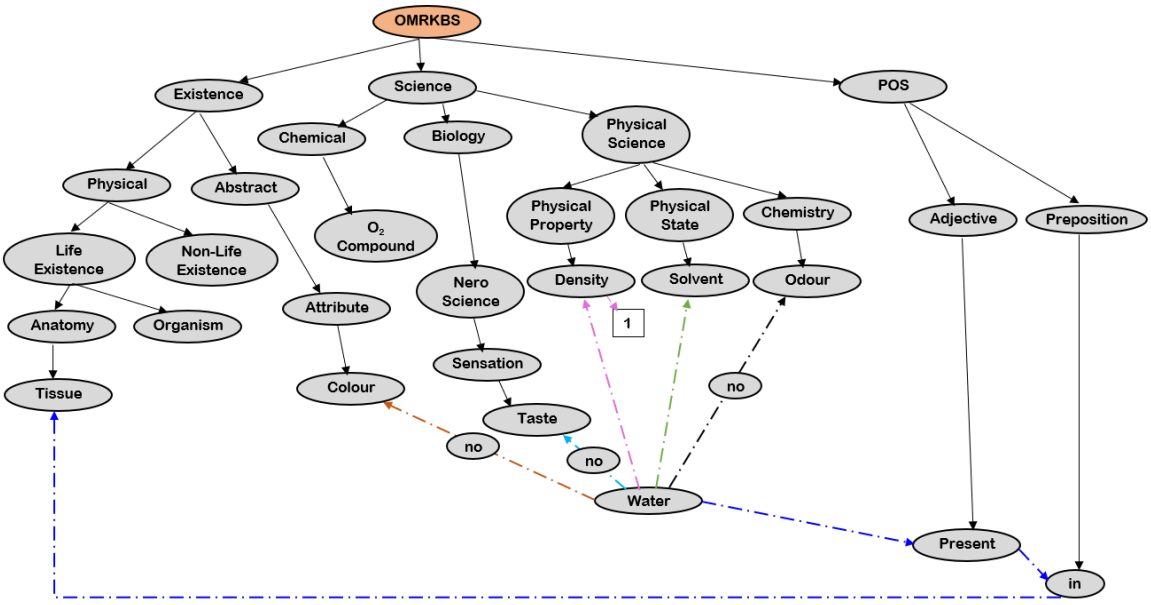


Figure 4.2: An example of a class 'water' defined by the proposed ontology OMRKBC.

The words in the grey circles are concepts and the root is denoted by the orange circle. The different colored dashed arrows indicate the relationship between concepts according to the features of the concept 'water' i.e., <water, no, taste><water, no, color> while the solid arrows indicate the subclasses.

4.5. The OMRKBC System

We describe how OMRKBC is built. Firstly, we construct the base ontology manually. Then, we introduce methods to discover the concepts and relations. Next, we develop a standard procedure to define the concepts through RSI.

4.5.1. Constructing Base Ontology

Concepts are classified and stored in a hierarchical structure in an ontology. Three majors domains: 'existence', 'science', and 'part of speech (POS)' are the top of the structure in OMRKBC. These top three domains will be the basic class labels in the ontology, which means all the concepts will be assigned under these three classes. These basic class labels are built in the ontology manually. We illustrate these three class labels as follows. 'existence' is one root class of the hierarchical structure which is divided into 'physicalExistence', 'abstractExistence', 'entity'. 'physicalExistence' can be 'lifeExistence' and 'nonLifeExistence'. 'entity' is something that exists apart from the

other things, having neither an abstract or physical existence, having its own independent existence (e.g. ‘weather’).

Figure 4.3: Segment of science and existence domains.

Concepts marked with the symbol ⊖ are subclasses of the concepts marked with the symbol ⊕. The concepts denoted by the color orange are discovered by OMRKBS.

The ‘attribute’ and ‘relation’ class are added in the ‘abstractExistence’ class. Important phrases (e.g., ‘perform in’, ‘capable of’) are added in the relation class and important attributes (e.g., color, size) are added in the attributes class. The ‘generalAttributes’ class, which is a subclass of attributes, contains general properties of the class and the corresponding instances. Another top class is ‘science’ which has eight domains.

Figure 4.4: Segment of relation and attribute domains

These domains are related to eight major science disciplines: ‘behavior’ or ‘social science’, ‘biology’, ‘chemical’, ‘physical’, ‘food’, ‘medicine’, ‘diseases’, ‘technology’ that are mostly imported from CRISP [48]. These domains contain concept which are related to their topics. ‘parts of speech’ is one more top class, and some general words are added here such as verbs, prepositions, adjectives, adverbs and articles. An ontology with these basic classes is called the base ontology. Some important domains are extracted from BioPortal [47], EVS [107] and DBpedia [49] repository. For example, various types of important attributes (e.g., ‘shape’, ‘depth’, ‘speed’) from the ‘attributes’ domain of a thesaurus ontology in EVS and the ‘organization’, ‘place’, ‘creative work’, ‘entity’ and ‘action’ domains from the ontologies (e.g., schema, entity) in BioPortal and DBpedia are extracted and then these domains are placed under the base ontology. We take the example of water which are given as definition 1in Table 4.1. Figure 4.2 shows how “water’ can be defined by the proposed ontology OMRKBC. Figure 4.3 shows segments of the ‘existence’ and ‘science’ domains. We extend the base ontology to enrich the domain so that various types of concepts can be assigned under the base ontology. Figure 4.4 shows the segments of relation and attributes that are discovered by OMRKBC system.

4.5.2. Discovering Concepts

In OMRKBC, each word is assigned as a concept and each concept can be defined by its relationships with other concepts. This section explains how each word is discovered or allocated as a concept in an ontology. First, the existence of the word is checked in the OMRKBC system. A concept is discovered when the word exists as a concept in the ontology. Otherwise, the word should be assigned into the ontology. Assigning a word as a concept is as follows:

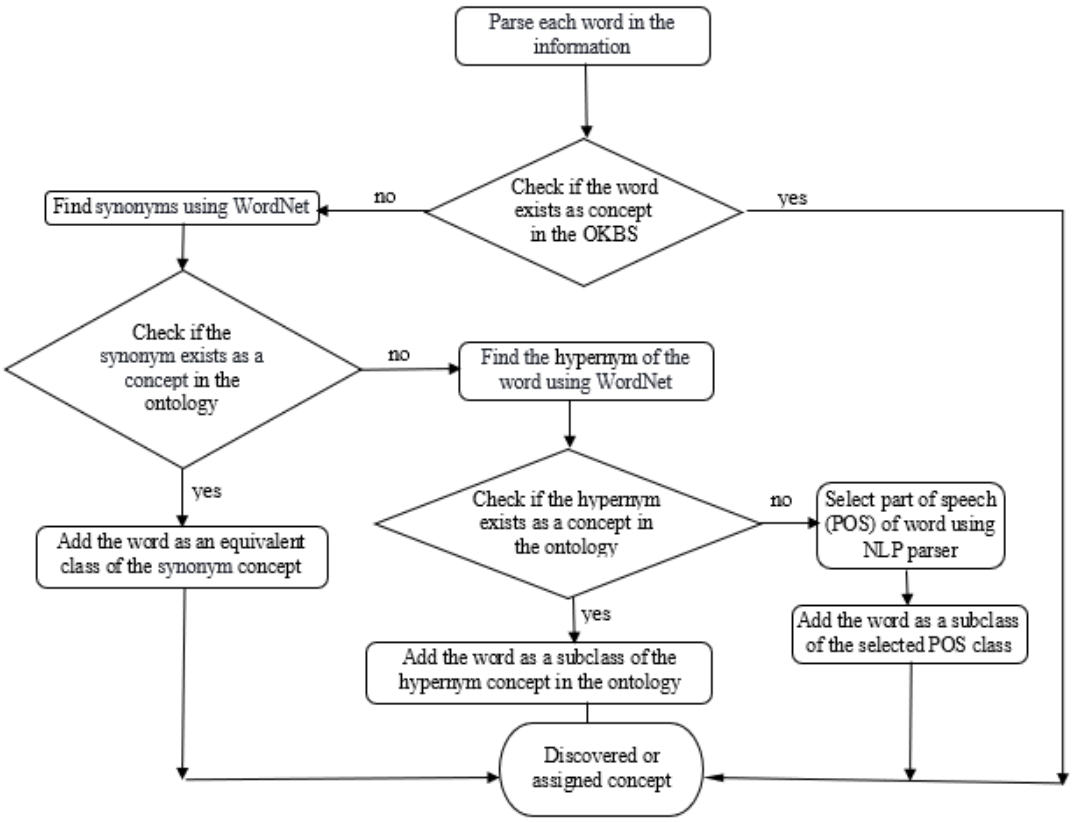


Figure 4.5: Flowchart to discover a concept in OMRKBC.

First, the word's synonyms are found using WordNet. When the synonyms are found in the ontology, the word is added as same class as the synonym class. For example, the synonym of '*undertaken*' using WordNet is '*take*' which exists in our ontology under the class of '*action*' as a verb. '*undertaken*' is added to the '*action*' class. If a synonym cannot be found in our ontology, the ontology should be checked to find the hypernym of the word. If a hypernym is found, the word is added under the hypernym class. For instance, when '*action*' is a hypernym of '*perform*' in WordNet and '*action*' exists in OMRKBC,

'*perform*' is added under the '*action*' class. However, when a word does not exist in the ontology or it cannot be related to a concept using WordNet, then this word needs to be tagged as a part of speech and this word is added as an axiom under the part of speech class. For instance, '*fresh*' cannot be found in our ontology. So, '*fresh*' is tagged as an adjective and this word is added under the '*adjective*' class in our ontology. Verbs which are considered as actions are assigned to the '*action*' class. We used the Stanford NLP parser [108] to tag the POS. Figure 4.5 shows the flowchart to discover a concept. We use this method to discover concepts from various sources later.

Table 4.2: Set of rules to discover relations from document.

Rule 1	When a word which is a verb (V) or abstract noun (ABSN) or common noun (CN) or an adjective (ADJ) is followed by a preposition (P) in the information, the two consecutive words are counted as a relation. Example: power through (V, P), leader of (CN, P), good for (ADJ, P), respect for (ABSN, P)
Rule 2	When a word is a verb but is acting as an adjective (VADJ) and is followed by a noun in the information, the two consecutive words are counted as a relation. Example: washing machine (VADJ, N).
Rule 3	When a word is an adjective and is followed by a verb in the information, the two consecutive words are counted as relation. Example: dry cleaning (ADJ, V).

4.5.3. Discovering Relations

There are particular groups of words which are used often in sentences. These phrase words are discovered as concepts in an ontology. We call these groups of words '*relation*' since they can be used to link words in sentences. The relation is governed by a few rules when parsing the information to discover the concept. Table 4.2 shows the rule to discover the relation from the sentences. Finally, these two consecutive words are joined together as one word (e.g., '*powerThrough, leaderOf, goodFor, respectFor*') and are assigned as a concept in the subclass under the class 'relation'. Instantly, each word in the relation is discovered or assigned as a concept using the method outlined in Discovering concepts. The reason for calling the relation word a concept is because the concept can be defined with other concepts. Next, when a concept is discovered as a subclass of a class in the '*attributes*' domain, the attribute is added to the concept. For example, the '*American*' concept is subclass of the '*nationality*' concept in '*attribute*'. When '*American*' is

discovered in the structure, the '*American*' concept is added to the '*nationality*' class in the structure using ':' (e.g., *nationality: American*).

4.5.4. OMRKBC with Instances from DBpedia

Importing instances from the spreadsheet data of DBpedia in OMRKBS (IISDBS) is one part of the proposed OMRKBC process. We provide the background on the functional procedure of IISDBS. We also discuss the challenges of the functional procedure which motivate us to design a program with algorithms and techniques for IISDBS.

Extracting Instances

The largest DBpedia KBS which is extracted from the English edition of Wikipedia consists of over 400 domains. Each domain has various properties known as attributes. The core DBpedia data in tabular form are available in CSV format in http://web.informatik.uni-mannheim.de/DBpediaAsTables/. Each CSV file contains instances of one concept and corresponding instances of properties or attributes.

Example 5: As shown in Figure 4.6, the first column in the CSV file of the '*president*' domain contains the name of presidents as instances. In this file, '*president*' has more than 90 properties i.e. '*birthdate*', '*birthplace*', '*spouse*'... etc. and these property fields contain instances corresponding to each president's name.

Functional Procedure

A functional procedure from the Protégé project available at https://github.com/protegeproject/cellfie-plugin imports spreadsheet data into the ontology in three steps using Protégé [96]. Firstly, the contents of the Excel file are loaded using the Cellfile plugin. Cellfile is a plugin which supports the creation of OWL ontologies from spreadsheets through a flexible

	A	B	C	D	E	F	G	H	I	J	K
1	**President**	**birthDate**	**birthPlace**	**spouse**	**age**	**country**	**children**	**politicalParty**	**serviceStartYear**	**serviceEndYear**	**Occupation**
2	Barack Obama	August 4, 1961	Honolulu, USA	Michelle Robi	57	USA	2	Democratic	January 20, 2009	January 20, 2017	
3	Donald Trump	June 14, 1946	New York City	Ivana Zelnickc	72	USA	5	Republican	January 20, 2017		Politician | businessman
4	Justin Trudeau	December 25, 19	Ottawa, Canada	Sophie Grégoi	47	Canada		Liberal	November 4, 2015		Teacher | politician
5	Moon Jae-in	24 January 1953	Geoje, South Korea	Kim Jung-sook	66	South Kore	2	Democratic	May 10, 2017		

Figure 4.6: Example of instances of '*president*' domain in DBpedia CSV format

mapping expression which maps spreadsheet content to OWL ontologies. Next, a simple mapping rule or expression for the class declaration axiom is created. Finally, axioms are

imported into the ontology. We adopt the functional procedure to import instances from DBpedia data into our ontology.

Algorithm 1: Pre-process CSV data

```
input  : String FileName: The location of CSV file and file name
output: Excel file generation with formatted data
initialization
eachRow: contain the row of csv file ;
begin
    while eachRow in FileName do
        A[]← contain cell content of each row;
        for i ← 0 to A.length do
            B← A[i];
            B← removeAll_invalid_char(B)
            B← replaceAll_null(B)
            A[i] ← B;
        end
    end
end
```

Addressing the Challenges of IISDBS

There are several challenges when implementing this procedure. We focus on three challenges in building the IISDBS. The first challenge is pre-processing the extracted CSV data so that these data can be imported into OMRKBS efficiently. The next challenge is to discover or allocate each word in the data as a concept in the ontology. The last challenge is mapping the data to embed instances in OMRKBS. We develop algorithms to address the three challenges which are discussed in the following:

a) *Pre-processing the data:* Protégé [96] or OWL API [109] only support Excel files with .xlsx extensions when importing spreadsheet data into an ontology. A CSV file contains many invalid characters which are not supported when being imported into an ontology, which results in many NULL values which consume a lot of space in an ontology. Also, large files take a long time to process and sometimes the process is terminated, which is another challenge. A lot of work has to be done manually before CSV data can be imported into an ontology. Therefore, CSV data should be pre-processed before being imported into our ontology. For this reason, we propose an algorithm for pre-processing the data from a CSV file. This algorithm converts all CSV files into Excel files and the file size is reduced by almost 68.4% for each file. All invalid characters are removed

from the CSV files and all null values are replaced with empty values. After this, the file size reduced by 93%. Excel files are split after each 3000 rows, resulting in thousands of Excel files which are only 716 KB in size for each file. The overviews of the algorithm for preprocessing the spreadsheet data are shown in Algorithm 1. Now, the system can load and process each file and embed the instances in OMRKBS.

b) *Discovering Concepts:* The names of all attributes or properties are concepts in our ontology. The reason for adding attributes as concepts instead of object properties is because an attribute as a concept can be related to other concepts in the proposed ontology. Discovering each property's name as a concept is one of the important points in the IISDBS procedure. This stage confirms that the names of all the properties in the CSV file are discovered or allocated as concepts. The names of the properties in a domain which are presented in the first row of the excel file are imported as concepts into the ontology. The properties of the concepts i.e., '*birthdate*', '*birthplace*' are imported as concepts under the '*generalAttributes*' class. When importing the data into the ontology, it is important to check whether this concept already exists or not. Suppose the property name i.e., '*spouse*' already exists in the ontology, this property name will not be imported into the ontology twice. Now each word in the property names is discovered or assigned using the method described in "Discovering concepts" section. For example, '*birthdate*' contains two words birth and date. These words are assigned or discovered.

c) *Mapping:* Mapping spreadsheet content to OWL ontologies is a great challenge in the process of IISDBS. There are more than 400 domains in DBpedia. Writing mapping expressions for each domain with properties is time consuming. Furthermore, some domains contain more than 700 properties. Manually writing mapping rules for a large number of properties in a domain is a tedious task. We notice that the mapping expressions [110] are the same for all domains except the properties and column names. We consider that these property names and columns are variables. Example 5 shows a part of the CSV file of the president domain where the president names exist in the first column '*A*' i.e., '*Barak Obama'* and the other columns i.e., '*B*', '*C*' and '*D*' contain data on the corresponding properties i.e., '*birthdate*', '*birthplace*', '*spouse*'.

The mapping expression can be written automatically for each domain with only the property name P_1, P_2, P_3 (‘*birthdate*’, ‘*birthplace*’, ‘*spouse*’) ...corresponding to the column name (*‘A’, ‘B’, ‘C’*...) needing to be changed. Otherwise, all terms in the expression are the same. Since all column names corresponding to the property names are listed in the file, we can write a fact expression programmatically. Therefore, a pseudocode is developed to create a mapping expression to map the spreadsheet data to the ontology. Firstly, the property names of a domain which are presented in the first row are extracted from the Excel file. After this, an array list is used to store the property names P_1, P_2, P_3 . . . (i.e., ‘*birthdate*’, ‘*birthplace*’, ‘*spouse*’). Also, a function ‘*getNameFromNumber*’ is devised to generate the column names (i.e., *‘A’, ‘B’, ‘C’*) which correspond to the instances of a property or domain and return the column name in colname in Algorithm 2. For instance, ‘*birthdate*’ is a property of the president domain and instances of ‘*birthdate*’ are listed in column ‘*B* (see Figure. 4.5). The function will return the column name (e.g., ‘*B*’) of the property name (‘*birthdate*’). If properties exist as object properties in an ontology, we can write the mapping rule using the fact expression using a variable of the property name and the corresponding column name as follows. Here, instances of concept are imported from ‘A’ and corresponding properties P_1, P_2 . . . information is imported from ‘B’,’C’... using the mapping rule as follows.

Individual: @ A∗
Types: C_{oncept}
Facts: P_1*@B∗,* P_2*@C∗,* P_3*@D∗, . . .*

However, properties exist as concepts not as object properties in our ontology. So, before using a fact expression in the mapping rule, we create individuals or instances for properties or concepts using the mapping rule as follows.

$mapping\_rule_1$ *= Individual: @B∗ (*$N_m = P_1$*#)*
Types: P_1
Individual: @C∗ ($N_m = P_2$*#)*
Types: P_2

For example, ‘*birthdate*’, ‘*birthplace*’ are concepts in the ontology and the *‘B*’ and ‘*C*’ columns in the CSV file contain all the instances of ‘*birthdate*’, ‘*birthplace*’ respectively. The instances of these properties are imported using the following

mapping expression mapping_rule$_1$. The instances of properties or attributes P_1, P_2 are imported from column *'B','C'*.... Nm contains the string 'mm:namespace'. We used the Nm variable which allows the ontology to have specific reference to properties. We used this reference to identify the instances of the attributes. After all the individuals corresponding to the concepts in the ontology are added, we relate the instances or individual concepts as properties to the domain or concept in the second phase. For instance, '*birthdate*', '*birthplace*' contain data on the corresponding president's name in the CSV file. Instances of properties are related to the concept i.e., '*president*' using the following mapping expression mapping_rule$_2$.

mapping_rule$_2$ = Individual: @A∗
Types: C$_{oncept}$
Facts: hasPropertyValue@B∗(Mp(P1#")), hasPropertyValue @C∗(Mp(P2#)) . . .

Here, column '*A*' contains all the instances of the concept (i.e. president names) and '*hasPropertyValue*' is an object property which is used only to relate the instances of the other concepts or attributes P_1, P_2 . . . with instances of the main concept $C_{concept}$ i.e. the president's name. Concept variables contain the name of main concept ('*president*'). The instances of P_1, P_2 . . . lie in the CSV data in columns '*B*', '*C*'...respectively. M_p contains 'mm.prepend' which is used to prepend the properties' names with each instance.

In cases where the property name already exists as a concept, then this property name is declared to be an equivalent class as the existing concept. For example, "*occupation*' attribute is an equivalent class to "*occupation*'. Now, when the instances of a class already exist as a concept in the OMRKBS, the existing classes are added as types of instances through the 'types' properties. For instance, businessman is an instance of "*occupation*' and "*lawyer*' also exists as a class under the "*occupation*' class. So, the concept "*lawyer*' is added as type to the "*lawyer*' instances through the 'types' of property. Also, we see in Example 5 that some instances (e.g., the "*occupation*' field) contain multiple values separated by '|' where each value is another instance. We split the instances or axiom by '|' and consider each split value (e.g., lawyer, politician) as the instance corresponding to the concept (e.g. "*occupation*'). We add each split value separately corresponding to the

concept in OMRKBC. After that, we relate these instances of concepts to the main concept to be defined (e.g., "*president*"). Algorithm 2 shows the pseudocode for creating the mapping rules automatically.

Algorithm 3: Import Spreadsheet Data

input : String FileName: The Excel file name and domainNm: domain Name
output: generate axiom from excelfile and import into ontology

Initialization
ontology: contain an OWL ontology;
ontologySource: contain an ontology source
spreadsheetSource: contain spreadsheet data source
owlRendering: contain set of OWL axioms
mappingRule: contain mapping expression as string
mmExpression: contain node representing the expression
columnNumber: contain number of column in file
rowNumber: contain number of row in file
mapping_rule: contain mapping expression as string ;
begin
    ontologySource ← createOntologysource(ontology)
    spreadsheetSource ← createOntologysource(FileName)
    mapping_rule ← CreateMappingExpression(FileName, domainNm) /*call create mapping expression algorithm */
    mmExpression ← CreateMappingMasterExpression(mappingrule)
    parser: Mapping Master parser
    parser ← CreateMappingMasterParser(mmExpression)
    mmExpressionNode ← ParseNodeForExpression(parser)
end

4.5.5. OMRKBC Program

In this section, we explore how the IISDBS process is executed into a program efficiently. This is the main program where CSV content is imported into the ontology. Mapping master [44] is a source library which can be used to transform the content of spreadsheets to OWL ontologies. We use this library with OWL API in Java to convert the spreadsheet into ontologies [109]. The three algorithms proposed to address the challenges in IISDBS are called by the program in order. Primarily, CSV data are pre-processed, and large files are split into multiple files after being pre-processed. This code executes tasks for each file through a loop. First, each Excel file is loaded into the program. Next, the domain properties are discovered as concepts in the ontology. After this, the mapping expression algorithm is called, and the mapped master expression is returned to the node which

represents the expression. Then, the data is looped as specified by the Mapping Master expression. Finally, the OWL axioms rendered by the Mapping Master expression are added to the source ontology. Algorithm 3 shows the pseudocode for importing Excel data into the ontology using OWL API. In line 8, we can see that a Mapping Master expression is rendered over a range of cells in a sheet. A Mapping Master parser is created for the expression in line 10. The parser parses and returns a node representing the expression in line 11. In line 12, the cells are looped as specified by the Mapping Master expression. Line 14 shows that a Mapping Master expression is rendered in the context of a location in a spreadsheet. The OWL axioms are added which are rendered by the Mapping Master expression in line 20. The system takes an average of 20.1 min to embed the instances of each file of concepts after resolving the challenges. On the contrary, the large file could not be loaded and mapped into the system before resolving the challenges. In "Experiments and a comparison of the results" section, we discuss the details of the space reduction and the time consumed for this program.

4.5.6. OMRKBC with ConceptNet Data

This section explores how ConceptNet data are built in the OMRKBS. We discuss the challenges involved in importing data from ConceptNet into OMRKBC and discuss the solution. Then, we present a program to build the ConceptNet data in OMRKBC.

Extracting the Data

ConceptNet provides seven large CSV files as datasets which can be downloaded from https://github.com/commonsense/conceptnet5/wiki/Downloads. The important fields or columns in the CSV files are 'relation', 'node at the start' and 'node at the end'. To better understand the ConceptNet dataset in CSV format, Example 6 is given 'relation', 'node at the start' and 'node at the end' are expressed by the edge: '*/r/CapableOf*', '*/c/en/ president*' and '*/c/en/govern_a_nation respectively*'.

Example 6: As an example, the 'president is capable of governing a nation' appears in the ConceptNet dataset as follows: */r/ CapableOf /c/en/president /c/en/govern_a_nation /ctx/all "weight": 1.0*

Addressing the Challenges of Building RSI from ConceptNet

We concentrate on importing the features of the concept associated with the relations from ConceptNet. The challenges in converting the data into RSI are discussed and resolved.

a) *Pre-processing the data:* The ConceptNet CSV files are too large to open. Therefore, all the CSV files are imported into the MySQL database where relation, start node and end node' are three fields in the table of the ConceptNet database. The data of 'start node' associate the 'relation' with 'end node'. We can see from Example 6 that 'start node' contains '*president*', 'relation' contains '*capableOf*' and 'end node' contains '*govern a nation*' for the sentence '*president is capable of governing a nation*'. The information on each concept is queried with each relation where the concept is matched with the 'start node' or 'end node' field in the dataset. For instance, data are queried about a concept i.e., '*president* where the 'start node' or 'end node' field is like '%president%' and 'relation' is like '*capableOf*'. The query will return all data lying between '*capableOf*' and '*president*' which means the query will return all things a president is capable of (i.e., *governing a nation*).

b) *Discovering concepts and relations:* This stage confirms that all the relations of ConceptNet are built in OMRKBC. Each concept is discovered here using the method described in "Discovering concepts" section. First, ConceptNet uses some important relations to represent concepts and these relations appear in the relation field of the dataset. They are '*capable of*', '*used for*' etc. All relations are assigned under the '*relation*' class. After this, each word in the relation is discovered or assigned if the relation contains more than one word. Next, we split all the words in the statement or description (i.e., '*governing a nation*') which results from the query and each word in the statement is discovered or assigned as a concept. In cases where any word group in a statement is identified as a relation according to the rule given in "Discovering relations" section, these word groups are assigned as concepts under the '*relation*' class.

c) *Mapping:* The associations between statements and relations corresponding to a domain (i.e., '*president*') are mapped in the OMRKBS. A mapping expression is given to build relations with the domain as shown in Example 6.

(govern and nation) and *capableOf* is a superclass of president

A synonym is also a relation in ConceptNet. The synonym of a word is expressed the same as other relations in OMRKBC. For example, in ConceptNet, the synonyms of '*president*' are '*head of state*'. This is shown by the following expression.

(head and of and state) and *synonym* is a superclass of president

OMRKBC Program

We design a program to import the features of the concepts associated with the relations from ConceptNet. We use the procedures proposed to resolve the three challenges in converting the data into RSI. Then, the RSI is built in the OMRKBS using the proposed program.

4.5.7. OMRKBC with a Description of Concepts

Defining a concept with a description is an important part of the proposed OMRKBC process. We identify the challenges in converting a description to RSI and provide the formula to address the challenges. In this section, we explore how concepts are defined by a short abstract or description through RSI. Also, we define the instances with a description and the relation with meanings.

Extracting the Description

The DBpedia dataset provides a short abstract for each article and can be downloaded from (http://wiki.dbpedia.org/data-set-36). This abstract can be used as a definition or description of the concept. A short abstract from DBpedia is shown in Examples 7–9 from Table 4.3. Next, we extract the meaning of the concept as text from WordNet. Then, concepts are imported with a description or annotation from the existing ontologies (i.e., CRISP, Schema) while constructing the base ontology. The meaning of the relation is retrieved from the Oxford Dictionary [111] using API, which is available at https://developer.oxforddictionaries.com and the relation is defined with the meaning. We take three examples from Examples 7 to 9 to illustrate the OMRKBC with definition and they are

'politician' domain, concept 'president' which is subclass of 'politician and 'Barak Obama' which is instances of 'president'.

Table 4.3: Examples of short abstract which are used to describe to transform into RSI.

Example 7: For instance, a short abstract of the "*politician*' domain is written in the dataset as http://dbpedia.org/resource/politician "A politician is a person active in party politics, or a person holding or seeking office in government. In democratic countries, politicians seek elective positions within a government through elections. In non-democratic countries, they employ other means of reaching power through appointment, bribery, revolutions and intrigues. Politicians propose, support and create laws or policies that govern the land and, by extension, its people. Broadly speaking, a politician can be anyone who seeks to achieve political power in any bureaucratic institution".
Example 8: A short abstract of the 'president' concept is given as "A president is the leader of a country or a division or part of a country, typically a republic, a democracy, or a dictatorship. Among other things, President today is a common title for the heads of state of most republics, whether presidential republics, semi-presidential republics or parliamentary republics".
Example 9: Take for example instance 'Barak Obama of 'president'. The short abstract of 'Barak Obama' is Barack Obama is an American politician who is the 44th President of the United States. He an American politician, author, and retired attorney. He is the first African American president born in Honolulu, Hawaii. He worked as a community organizer in Chicago. Obama signed landmark bills, the Affordable Care Act , the Dodd–Frank Wall Street Reform and Consumer Protection Act; and the Don't Ask, Don't Tell Repeal Act of 2010 bills.

Addressing Challenges in Building a Description in OMRKBC

Our challenge is to learn how to process the text in the description into RSI. As discussed, there are three types of challenges which must be addressed: pre-processing content, mapping information and embedding information.

a) *Preprocessing:* the content short abstracts or descriptions must be pre-processed because sentences in the short abstract may be too complex or too long to relate words in the sentence with concepts. As shown in Examples 7–9, the sentence is so complex that it is difficult to relate the words in the sentences directly with the concepts in the ontology. Therefore, the text in an abstract is reformed into RSI in three steps. Firstly, each complex or long sentence is split into several simple clauses. Open information extraction (Open IE) [106] which is part of the Stanford NLP parser [108] extracts simple clauses from sentences.

Table 4.4: Example of how sentences are formatted after split.

Politician	*President*
1. <politician, create; laws><create policies> *2.* <politician, propose, laws><propose, policies> *3.* <politician, seek elective positions within; a government> *4.* <politician, holding office in; a government> *5.* <politician, seek political power; to achieve, in any bureaucratic institution> *6.* <politician, reaching power through, bribery> *7.* <politician, reaching power through, revolutions> *8.* <politician, reaching power through, intrigues> *9.* <politician, seek elective positions at; times> *10.* <politician, is active person, in party politics> *11.* <politician, is, person active>	*1.* <president, is the leader of, a country> *2.* <president, common title for, the heads of state of most republics> *3.* <president, is, common title> *4.* <president, is the leader of, division or part of a country> *5.* <president, is, leader> *6.* <president, is leader of, republic> *7.* <president, is leader of, democracy> *8.* <president, is leader of, dictatorship>
Barak Obama *1.* <Barack Obama, American politician> *2.* <Barack Obama, first African American president> *3.* < Barack Obama, born in, Hawaii> *4.* <Barack Obama, 44th president of the United States> *5.* < Barack Obama, worked as, community organizer> *6.* <Barack Obama, is, Author> *7.* <Barack Obama, is, retired attorney> *8.* <Barack Obama, signed, landmark bill>	*9.* <Barack Obama, signed, the Affordable Care Act bill> *10.* <Barack Obama, signed, the Dodd–Frank Wall Street Reform and Consumer Protection Act bill> *11.* <Barack Obama, signed, the Don't Ask, Don't Tell Repeal Act of 2010 bill>

Each simple sentence appears to be an individual feature of the concept and is presented as (subject; property; object). Table 4.4 shows examples of how sentences which are taken from Examples 7–9 are formatted after splitting. Secondly, we remove some sentences from a list of simple sentences before converting the structured input to reduce redundancy. First, we only take one sentence which contains the subject, object and predicate related to the concept to be defined. If a synonym or an equivalent of the concept exists as a subject or object, we consider the sentence also. Next, if there is more than one sentence which looks similar or almost similar, the most complete sentence is included in the structure. For example, between two statements: *<politician, active person, in party politics><politician, is, person active>*, the complete statement

is <*politician, active person, in party politics*>. We remove the other similar statement. Finally, the rest of the simple sentences are converted into structured information. Subject is the concept to be defined, and (predicate, object) are the characterization of the concept. The structures of the simple sentences are constituted from (predicate, object). Each predicate and object are parsed from each sentence and are turned into the structured input using a few rules. The structured input is presented with a series of arguments and each argument is separated with ‘,’. We present the rules to transform the definition into structured information input in Table 4.5. The features which were generated from the sentences in Examples 7–9 by splitting are structured with the rules shown in Table 4.6. Each sentence in the description has been formatted into structural information so that the structure of the sentences can be mapped easily to build RSI into the ontology.

Table 4.5: Set of rules to transform documents to structure information input.

Rule 1	The verb to be in the predicate acts as simple present and will not be included in the structure.
Rule 2	When a clause in the predicate or object contains ‘of’ or ‘*by*’, the clause is split into three parts: the first part is the words following ‘*by*’/’*of*’, the next part is ‘*of*’/’*by*’ itself, and the last part is the words preceding ‘*of*’/’*by*’. Each part is an argument in the structure.
Rule 3	When a clause in the predicate or object can be declared as a relation using the method in “Discovering relations”, the relation word is an argument.
Rule 4	if a verb is not included as a relation as in Rule 3, the verb is included in the structure in base form.
Rule 5	Adverbs located before verbs are removed in the structure.
Rule *6*	Adjectives of the subject or object are removed in the structure.
Rule 7	When the structure contains only the object with no prepositions (e.g., ‘*of*’) and a concept (e.g., ‘Author’) in the structure is discovered as a subclass of the attributes class (e.g., ‘*occupation*’), the attribute class is included as an argument in the structure also.

b) *Discovering concepts and relations:* Here, each word is discovered using the method discussed in “Discovering concepts” section. After extracting the structure of sentences in the abstract, each word in the structure is discovered as a concept. Instantly, possible relations and attributes in the structure are identified according to the rule given in “Discovering relations” section.

Table 4.6: Examples of how rules structure the sentences about concepts.

Politician	
<create; laws> <create, policies> [1]	No rule
<propose, laws><propose, policies> [2]	No rule
<seek elective, positionsWithin, a government> [3]	Rule 3
<reach, powerThrough, bribery> [6] <reach, powerThrough, revolutions> [7] <reach, powerThrough, intrigues> [8]	Rule 4
<seek political power to, achieveIn, bureaucratic institution>	Rule 3 and Rule 6
<active person, in party politics> [10]	Rule 1
<hold office, in government> [4]	Rule 4
President	
<leaderOf, a country> [1]	Rule 1 and Rule 3
<title for, heads of state, of, republics> [2]	Rule 1 and Rule 2 and Rule 6
<leaderOf, division or part, of, a country> [4]	Rule 1 and Rule 3 and Rule 2
<leader> [5]	Rule 1
<leaderOf, republic> [6] <leaderOf, democracy> [7] <leaderOf, dictatorship> [8]	Rule 1 and Rule 3
Barak Obama	
<American politician> [1] <president> [2] <first, African, President> [2] < 44th president> [3]	Rule 1
< workedAs, community organizer>	Rule 1 and Rule 3
<occupation, retired attorney > [5] <occupation, Author > [5] <nationality, American> [5]	Rule 1 and Rule 7
<bornin, Hawaii> [6]	Rule 3
<sign, landmark bill> <sign, the Affordable Care Act bill > < sign, Dodd–Frank Wall Street Reform and Consumer Protection Act bill> <sign, the Don't Ask, Don't Tell Repeal Act of 2010 bill>	Rule 1 and Rule 2

Bold face font are the concepts which define with examples. The features of concept are taken from Table 4.4. All these examples have moved to another concept, as indicated by the dark grey color Number in the bracket indicate the number of the feature, we mentioned in the Table 4.4 after split.

c) *Mapping:* Each sentence in the description about a concept or instance are built into the ontology according to the structure of the sentences. For this, the relationship among the arguments in the structure are mapped using the following two phases. First, the relationships among the words in the arguments and the relationships among the arguments in the structure are made by joining the words with an 'and' expression. Then, this relation is declared as a superclass of concept or types of instances.

Example 10: As an illustration, we take some sentence structures about '*politician*' from Table 4.6 and map them using the following expression.

(*create* and *law*) is a superclass of *president*

(reach and *powerThrough)* and *bribery* is superclass of *president*

Example 11: As another example, '*Barak Obama'* is an instance in OMRKBC and the expression for mapping is as follows:

(bornIn and *Hawaii)* is types of *Barak Obama*

(nationality and *American)* is types of *Barak Obama*

The reason for declaring features as super-classes is because these features or characteristics are inherited by the concept and the subclasses of the concept. The features of the instances will be derived through property type.

OMRKBC Program

We designed a program to define the concept with a description through RSI. In this program, the text in the description or abstract is processed into structural information by resolving the challenges and the structural information is built in the OMRKBS. In conclusion, when any new word is added as a concept in the ontology, this new concept can be defined with the description and the data and instances and synonyms from DBpedia and ConceptNet. Thus, we can develop an independent ontology based KBS.

4.6. System Output

In this section, we show how information about a word is queried and represented in the OMRKBS. We use the SPARQL [71] language for the query and format the proposed ontology in the RDF format. We introduce three types of searches in the system and represent the information according to the search. In the following, we describe these three types of searches: concept search, instance search and process queries. Generally, when a word is searched in the system to retrieve information on this word, we call the word a '*query word*'. We declared a few PREFIXs to reference IRIs where a nsf prefix is a source of OMRKBC.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schem a#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX nsf: http://www.seman ticweb.org/shirinkhan /2017/

4.6.1. Concept Search

A concept search is executed when a '*query word*' exists as a concept in an ontology. First, we introduce the queries to return the information about the concept. After this, the resulting data for a concept is represented. There are two types of queries by which to extract information about the concept.

Table 4.7: Queries for the concept search of 'president'

No	Query
1	select ?superclass where nsf:politician rdfs:subClassOf ?superclass.
2	select ?entity ? type WHERE {?entity rdf:type ?type. ?type rdfs:subClassOf* nsf:president.}

These queries extract features or characteristics, instances, and general attributes. Table 4.7 shows the queries for the '*president*' concept and the result of the queries. The query on the first row retrieved the features of the concept and the second row retrieved the instances of the concept. In this section, we explore how information is presented with these queries.

Feature Representation

Since features are considered to be a superclass of a concept, the extraction of a feature becomes easier. A simple SPARQL query syntax is written to retrieve all the super classes of a concept which represent the characteristics or features of a concept in an ontology. The query syntax and results of a query are in the first row of Table 4.7. All the features of a concept are combined from the resulting data and are then represented as the definition of the concept. The presentation of the resulting information replaces the 'and'/'or' expression' with ',' in the original result.

Example 12: We give examples of the representation of features as the definition of concept '*president*' as follows:

President	
Leader Title for heads of state of republics Leader of: Country Division of a country Part of a country Republic, democracy, dictatorship	*Capable of*: Give a speech Duck the issue Govern a population

Because each feature appears individually, and the feature may be associated with attributes and relations in the result, the presentation of the results becomes more specific and meaningful. In Example 12, we understand from the result: who is the leader of country. Also, some important relations such as '*leaderOf*' helps to separate some characteristics which specify the result. We consider a relation to be important if the relation has a connection with more than two features.

Example 13: For example, '*president*' concept returns the list of president names in the following format.

List of presidents
Barack Obama
Barak Obama
Justin Trudeau
Moon Jae-in
..
..

Instance Representations

The query in the second row of Table 4.7 returns the instances of the concept. Also, if the instances of a subclasses of a concept exist in an ontology, this query returns the instances of the subclasses with the subclasses name. We place ',' between the names of the instances and the subclasses in the representation of the resulting data. Appending instances of the concept makes the output more informative. In conclusion, when a concept search is performed, the results of these two types of queries are represented in a bind. Example 13 is given to illustrate the instance representation.

Table 4.8: Query for the instance search (i.e., 'Barak Obama')

Query
select distinct ?value ?generalAttributes ?Gtype ?GClass where { nsf: 'Barak Obama' ?property ?value. nsf:'Barak Obama' rdf:type ?GClass. FILTER(?GClass != owl:Class&& ?GClass != owl:NamedIndividual) ?value rdf:type ?GeneralAttributes. OPTIONAL ?value rdfs:seeAlso ?label. ?label rdf:type ? Gtype. FILTER(? Gtype!= owl:Class&& ? Gtype!= owl:NamedIndividual) filter (?property not in (rdf:type)) FILTER(? GeneralAttributes!= owl:Class&& ? GeneralAttributes!= owl:NamedIndividual)}

4.6.2. Instance Search

When the '*query word*' exists as an instance of a concept in OMRKBC, the system starts the instance search. This search returns the concept of the instance and the instances of the attributes corresponding to the instance. Table 4.8 shows the query for this search and Example 14 shows how the information on the instance '*Barak Obama*' is represented.

Example 14: For example, '*Barak Obama*' is queried in the system and the query returns information about '*Barak Obama*' as shown in Table 4.8. The information means that '*Barak Obama*' is an instance of the 'president' class. This president's birthplace is the USA, and the USA is also a concept 'country' which is mentioned through 'type'." The results for the instances '*Barak Obama*' are presented as follows. Also, the instance will have the same feature of concept. For example, '*Barak Obama*' is an instance of the '*president*' concept. Therefore, '*Barak Obama*' has all the characteristics of a president. The features of a president which are inherited by instances are retrieved using a concept search. Thus, it is possible to vary the search scope in OMRKBS.

Barak Obama	
President	Birth Date: August 4, 1961
First African President	Birthplace: USA
American politician	Spouse: Michelle Robinson
Author	Nationality: American
Retired Attorney	Occupation: Author, lawyer
44th President	Start Year: 1976
Born in Hawaii	Political Party: Democratic
Sign bill	Age: 72
Landmark bill The Affordable Care Act bill Dodd–Frank Wall Street Reform and Consumer Protection Act bill Don't Ask, Don't Tell Repeal Act of 2010 bill	Start Year: January 20, 2009

4.6.3. Process Queries

Process queries allow the user to ask a question to find information about a word which has a specific property or attribute. This question is asked with two arguments where the first argument usually represents the main word that the user wants to know, and the second argument represents the property or attribute of a main word.

Each argument is separated by ',' in the question. The result of this search shows the relationship between the two arguments. Table 4.9 shows the query for a question ('*politician*', "*policies*') with the result. First, the main word can be the concept in the question. Also, the property or attribute can be a concept which describes the main concept with other concepts. This search returns all the features of the main concept related to the property or attribute. It also filters all the inherited features of the main concept associated with the property. We have given Example 15 and 16 in Table 4.10 to explain for this query. Secondly, suppose a user wants information on an instance associated with a specific property. In this case, the main word is an instance and the property is a concept in the question. This query returns instances of a property associated with a main word instance. We give example 17 to explain this query.

Table 4.9: The query for an example question ('politician', 'policies')

Query

SELECT ?subClassOf WHERE {
nsf:politician (rdfs:subClassOf| owl:equivalentClass) * ?subClassOf.
?subClassOf owl:intersectionOf | owl:unionOf ?subClassOf_name.
?subClassOf_name rdf:rest*/rdf:first* ?name.
optional{
?name owl:intersectionOf | owl:unionOf ?subsubClassOf_name.
?subsubClassOf_name rdf:rest*/rdf:first* ?subname.}
filter(nsf:policies in (?name) | | nsf: policies in (?subname))}

In short, a concept search finds the definition of a concept or word and an instance search returns information about an instance. Finally, a question can be asked about a word associated with a property, and the system will return data related to the word and the property.

Table 4.10: Example we have used to explain process queries.

Example 15	For example, '*politician*' is the main concept and '*policies*' is one property concept of '*policies'*. When the question ('*politician*', '*policies*') is asked in the system, the results show all the features of the politician related to policies (e.g., politician create policies).
Example 16	For example, '*president*' is a subclass of '*politician*' and '*politician*' has been defined in the ontology as <*politician create policies*>. When ('*president*', '*policies*') is queried, the results show <*president create policies*> as '*president*' inherits the superclass features of '*politician*'.
Example 17	For example, ('*Barak Obama*', '*occupation*') or ('*Barak Obama*', '*birthdate*') is queried in the system to retrieve information on the occupation or birthdate of '*Barak Obama*', hence the query will return the result (Barak Obama's occupation: author and birthdate: August 4, 1961).

4.7. Characteristics Comparison of OMRKBS with other KBSs

There are several fundamental qualities that facilitate efficient searches in the OMRKBS. First, every concept is defined by relating other concepts in the OMRKBS rather than by using annotation. The importance of the object and data properties is not significant, since each property is declared as a concept in OMRKBS. Then, OMRKBC supports various relations and attributes. Next, individual features are richly structured with relations and attributes and are called super classes of a concept in OMRKBS. The features are inherited by the concept and subclasses of the concept. Also, OMRKBS provides general information about concepts or instances (e.g., '*birthdate*', '*spouse*'). These fundamental qualities enable different types of searches in OMRKBS and assist in returning specific, meaningful, and logical information for the search.

Table 4.11: Comparison of the characteristics of OMRKBS with the existing KBSs

Characteristic	DBPedia	WordNet	ConceptNet	OMRKBC
Individual features are presented	X	X	√	√
Each word is a concept	X	X	X	√
No specific data property	X	X	X	√
Limited object properties	X	X	X	√
Provides general information on an instance or concept	√	X	X	√
Inherits the superclass features	X	√	X	√
Individual feature is presented with attributes	X	X	X	√
Individual feature is presented with relations	X	X	√	√
Allows a question to be asked about a concept with a property	√	X	X	√

In addition, as we see from Examples 15 to 17, OMRKBS enables questions to be asked with two arguments to understand the relation between these two arguments. The DBpedia system supports instance searches (e.g., '*Barak Obama*', '*occupation*') but not concept searches ('*politician*', '*policies*'). But other KBSs such as WordNet or ConceptNet do not support this type of query. Also, the individual features as defined will be inherited by the subclasses in response to the question. WordNet and our system have this functionality. '*president'* inherits all the features of '*politician*' and answer the question ('*president*', '*policies*') that <*president, create, policies*>. Therefore, the system can answer logical questions. Table 4.11 compares the characteristics of OMRKBS with the other KBSs.

4.8. Experiments and a Comparison of the Results

4.8.1. Implementation Setup

We propose a standard implementation of our framework in Java for Windows, version 10 and the experiments are performed on a PC with quad-core CPU (4 GHz) and 16 GB of RAM. The resources that were used in the development of OMRKBC are DBpedia, ConcpetNet 5 and WordNet. We used OWL API to import axioms as a subclass, superclass, or class in OMRKBC. The program uses the JWNL, which is an interface to the WordNet dataset. The synonymous terms and the considered hypernyms are retrieved from WordNet using this interface. We use the Mapping master source library to transform the content of the spreadsheets to OWL ontologies. We used Open IE and Stanford NLP parser library [108] sources to split the sentences. We connected the MySQL database with the MySQL JDBC driver to retrieve the ConceptNet data. We used the SPARQL query language to retrieve the data from our KBS.

4.8.2. Datasets

To evaluate OMRKBC, we created a KBS in English for the domain '*agent*' ('*person, artist, athlete, journalist*, etc.), *place* (*city, country, region, country*), *work, game* (*soccer, cricket, golf*, etc.), *organization*, *animal*, *educational institution* (*university*), *event, album, chemical*. We grouped the results by dataset and analyzed the outcome of the structural information from different sources. The datasets are ConceptNet 5, WordNet 3.0 and DBpedia. To compare the results with our KBS, we used the abstract and instances with the properties from DBpedia, and the relation statement '*capable of*', *'used for*' and '*type of*' from ConceptNet, and the meaning, hypernyms, and synonyms from WordNet.

4.8.3. Evaluation Methods

An evaluation team of 15 PhD students, all experienced in the field of information retrieval, was formed to assess the accuracy of the process queries, relation discovery and overall accuracy of OMRKBS.

$$\text{Accuracy} = \frac{|\{\text{relevant results}\} \cap \{\text{retrieved results}\}|}{|\{\text{retrieved results}\}|} \qquad 4.1$$

We calculate overall accuracy of OMRKBS by the mean accuracy of these two searches using the following equation.

$$Overall\ Accuracy = \frac{Concept\ search\ accuracy + \ Instance\ search\ accuracy}{2} \qquad 4.2$$

Table 4.12: The file size reduction for each domain after preprocessing algorithm step by step.

Domain	Original size	Size after excel conversion	Size after null and invalid values removed	File size after split (KB)
Agent	13 GB	5 GB (60%)	260 MB (98%)	800
Animal	250 MB	61 MB (75%)	23 MB (91%)	820
Chemical	15 MB	3.1 MB (79%)	1.7 MB (89%)	705
Event	134 MB	44 MB (68%)	8.1 MB (94%)	718
University	32 MB	7.5 MB (77%)	3.4 MB (89%)	690
Game	1 MB	188 KB (82%)	132 KB (87%)	660
Album	327 MB	116 MB (65%)	13 MB (96%)	650
Organization	1.32 GB	564 MB (57%)	46 MB (96%)	680
Place	4.43 GB	1.8 GB (60%)	125 MB (97%)	720
Work	1.7 GB	663 MB (61%)	27 MB (98%)	710
		68.4%	93.5%	716

The second column shows the original size of the file. Columns 3, 4 and 5 show the file size reduction after the program converts the file to Excel, removes the null and invalid values, and splits the file. The percentage of file reduction is given in parentheses. The bottom line indicates the average percentage of the file reduction of each step.

4.8.4. Results Analysis

First, we conduct an experiment to execute the program for IISDBS over the existing datasets. As part of the experiment for this program, we evaluate the space reduction of the file over the pre-processing algorithm and time consumption of the program of IISDBS. We can see from Table 4.12 that the average file reduction is 68.4% from the actual size after Excel conversion and 93.5% after the null and invalid values are removed over the pre-processing algorithm. The Excel files size is only 716 KB after each file is split. We embedded instances of one file for each domain. The execution times of the IISDBS program for each domain are shown in Figure 4.7. We can see from Figure 4.7 that the average execution time of each file is 20.14 min. This implies that the system can process the small size file and embed the data in the system after pre-processing the large file and creating an automatic mapping expression whereas the large file could not be loaded and mapped into the system before resolving these issues.

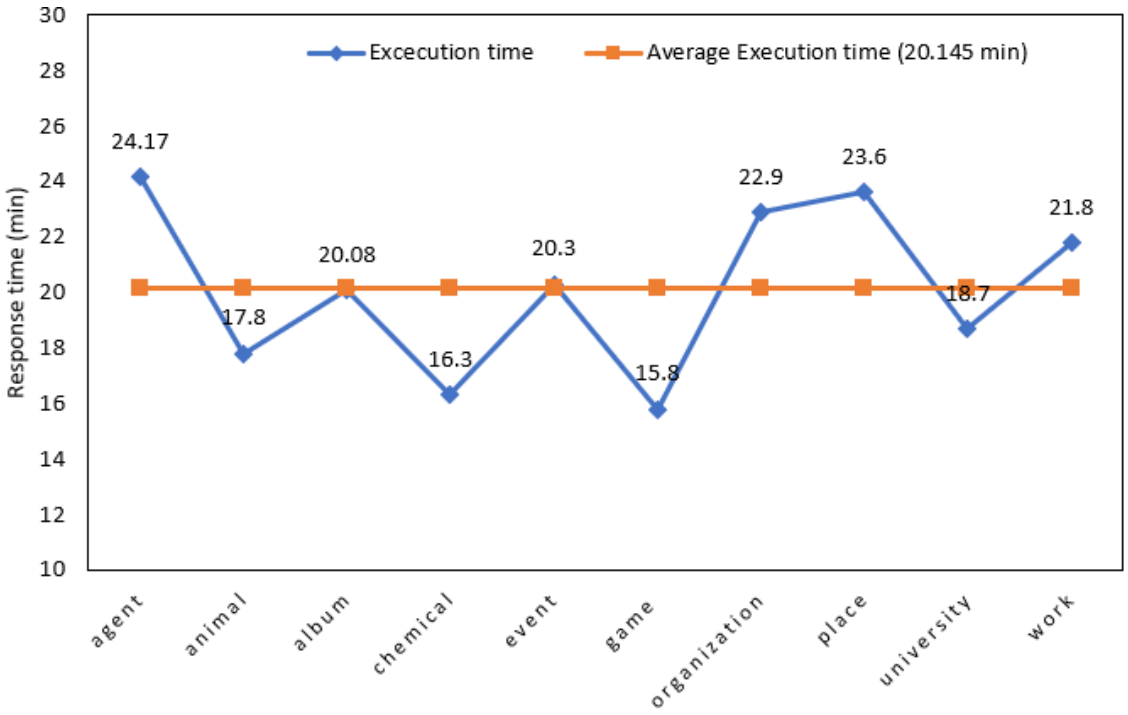


Figure 4.7: Time consumption to execute the IISDBS program.

Next, we evaluate the accuracy of the process queries, relation discovery and overall accuracy of OMRKBS over the selected datasets. We can see from Table 4.13 that the mean accuracy of the process queries is 93%. The accuracy of relation discovery is measured by the number of relations which are discovered correctly in system and the results shows a 100% accuracy. Finally, the evaluation team was asked to determine how well each statement in the results is expressed when a word is queried in the system. When a statement presents ambiguous, unrelated or unclear individual features, this information is regarded as being poorly expressed. On the other hand, when a statement presents relevant and correct individual features with attributes and relations, this information is regarded as being well expressed. The accuracy of the word query is calculated using the number of well-expressed features among the results. The evaluation team calculated the accuracy of the concept search and the instance search using equation (4.1). The results from each domain were averaged for the two searches as shown in Figure 4.8. We calculate the overall accuracy of OMRKBS by the mean accuracy of these two searches using equation (4.2) and present the overall accuracy in Figure 4.8. Observe that the accuracy of the instance search is slightly lower than the concept search. Interestingly the event and

organization domains have a higher overall accuracy whereas the chemical domain has the lowest accuracy.

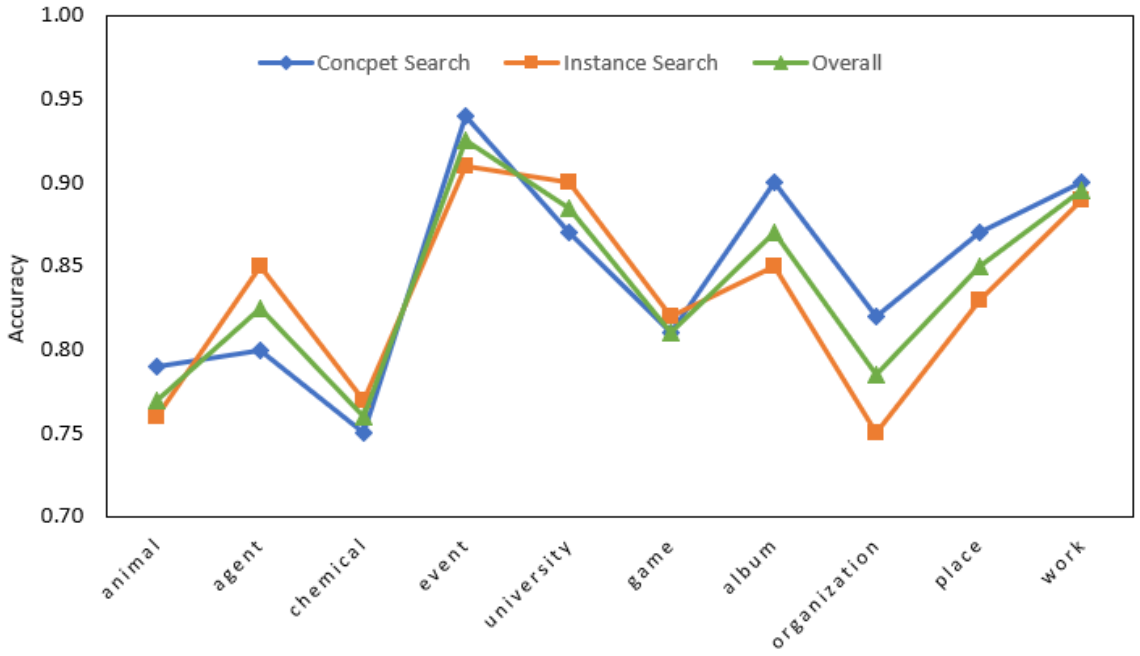


Figure 4.8: Evaluation of accuracy of concept search and instance search of important domain in OMRKBS.

The green line shows the overall accuracy of OMRKBC.

Table 4.13: Evaluation of the accuracy of the process queries and relation discovery

Domain	Process Queries	Accuracy
Agent	89%	100%
Animal	93%	100%
Chemical	96%	100%
Event	97%	100%
University	93%	100%
Game	94%	100%
Album	94%	100%
Organization	91%	100%
Place	90%	100%
Work	94%	100%
	93%	**100%**

Now the accuracy of the KBSs: DBpedia, ConceptNet, WordNet and OMRKBS is evaluated using the same dataset. Figure 4.9 gives a breakdown of the results. In OMRKBS, each word is a concept, and each concept is defined with other concepts rather than with statement or a description. Therefore, it is feasible to represent the features of the concepts with attributes, relations, and related concepts in specific ways. OMRKBS

supports many relations (i.e., *'born in*', '*capable of*') and attributes ('*nationality*', '*occupation*'). In DBpedia and WordNet, the definitions (e.g., abstracts, meanings) are descriptive and presented in text format whereas ConceptNet and OMRKBC provide more specific or individual features with relations and attributes. ConceptNet has a higher accuracy of 77% in terms of the appearance of individual features than DBpedia and WordNet as ConceptNet gives specific information with relations.

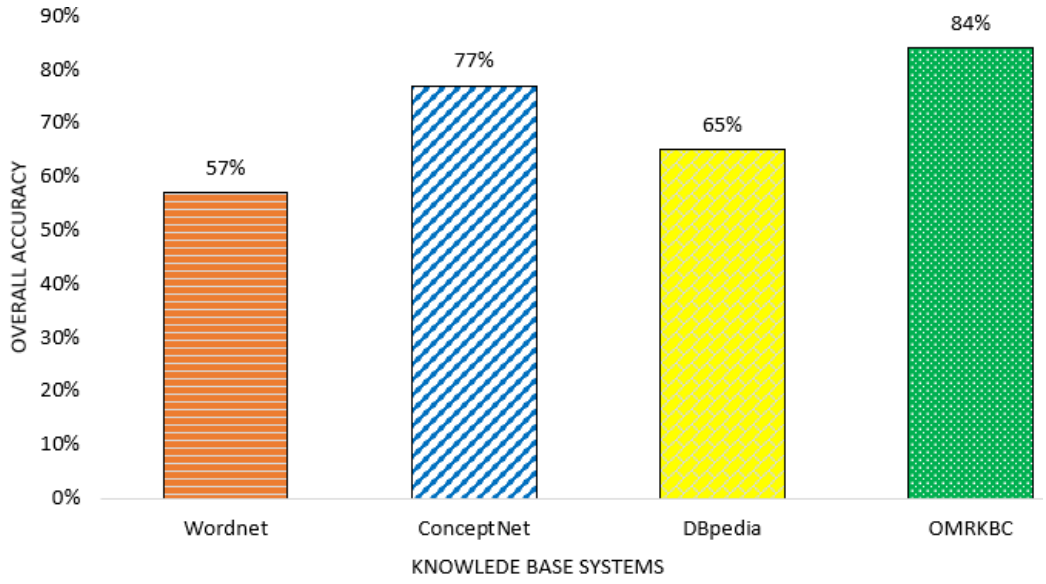


Figure 4.9: Comparison of the accuracy of OMRKBC with the existing KBSs over the same dataset.

However, OMRKBC is one repository where concepts are interconnected with relations and attributes in various ways. Therefore, individual features are informative and meaningful. For example, when an instance such as president name ('*Barak Obama*') is queried, OMRKBC shows the president's general information such as '*birthdate*', '*birthplace*' etc., whereas ConceptNet mostly does not provide general information. Although DBpedia provides general information with attributes, it does not present individual features with relations (e.g., <*Barak Obama, president of, USA*>). Some properties of an object are lost or invisible when defining an object in ConceptNet. In contrast, concepts or instances can be represented with attributes in OMRKBC. For instance, *Barak Obama* is businessman who is represented in our ontology <*Barak Obama, occupation, businessman*>. ConceptNet may mention '*Barak Obama*' is author, but the property of businessman is hidden. In this sense, the representation of data in OMRKBS is

more meaningful and tangible. OMRKBS has higher accuracy (84%) than the other KBSs for individual features. This result suggests that the individual features of concepts are well-structured in OMRKBS.

4.9. Summary

This chapter discussed the process of building an OMRKBS over the last few years and several issues relating to OMRKBC. We described the NLIKR scheme in which each concept, denoted by an English word or phrase, is defined by its relations with other concepts and its position in the concept space. We identified a key challenge, this being to convert the data into RSI using a classical technique to map information into the structure. However, the classical techniques are not effective on large datasets, hence, we used the NLIKR scheme to translate the information in OMRKBS. We applied rules, algorithms, and techniques to transform the data into RSI. As a result, the information is well structured with attributes and relations. This improves the effectiveness of the query results and has higher accuracy compared to the other KBSs. Though OMRKBC is not a fully independent KBS, it is partially developed and focuses on a specific domain. However, it is a proposed method, where the development of a complete large knowledge base repository is possible.

Chapter 5.

Concept-based Topic Attention for a Convolutional Sequence Text Summarization Model

Neural network-based text summarization often suffers from the problem of summarizing irrelevant topic content regarding the main idea. One of the main reasons leading to this problem is a lack of human commonsense knowledge which generates facts that are not decipherable. We propose a text summarization framework called CSN based Text Summarization with Concept-based Topic Triple Attention (TEXSCTTA). The framework incorporates concept-based topic information into a convolutional sequence text summarization model. We propose a concept-based topic model (CTM) to generate semantic topic information using conceptual information or knowledge which is retrieved from a knowledge base. We introduce a triple attention mechanism (TAM) to not only measure the importance of each topic concept and source element to the output elements but also the importance of the topic concept to the source element. TAM presents contextual information from three aspects and then combines them using a SoftMax activation to acquire the final probability distribution to enable the model to produce coherent and meaningful summaries with a wide range of rich vocabulary. The experimental evaluations which are conducted over the Gigaword and CNN/Daily Mail datasets reveal that TEXSCTTA surpasses the various widely recognized state of-the-art models (WSOTA) such as Seq2Seq, PGEN, CSM and TopicCSM. TEXSCTTA achieves competitive results by generating coherent and informative summaries.

5.1. Introduction

Text summarization is a mechanism to generate a brief statement for an original document that preserves the genuine meaning of the content. This is an important step in overcoming this task of summarization to understand natural language. Moreover, a concise and meaningful summary of documents assist humans to comprehend the document well in a short time. Text summarization is widely classified into two parts: extractive and abstractive based on the earlier research. Extractive summarization trims the important

chunk of the document to generate summaries and integrate them to produce a coherent summary. Abstractive summarization produces qualitatively more similar to human-written summaries from scratch which does not comes from the phrases of the original text directly. Extractive methods were firstly introduced to reproduce semantic information from the original document and summarize it.

Table 5.1: An example of generated summary of our model.

DOCUMENT: Barak Obama on Wednesday announced the closure of government schools with immediate effect as a military campaign against religious separatists escalated in the north of the country.
SUMMARY: Barak Obama shutdown school because war escalated in the north of the country.

Blue content" war" shows generated output element appeared from topic concepts. The topic concept" war" is the latent knowledge of the related words which are marked in red in the document.

More recently, deep learning models have attracted great interest from the research community in relation to text summarization approaches as they are able to achieve good summary results [36][42][112]. Sequence-to-sequence models have been proposed to map an input sequence into another output sequence in the abstractive summarization approach. There are useful models for abstractive summarization based on sequence-to-sequence RNNs such as long short-term memory (LSTM) with the encoder-decoder model. This model works well for machine translation where input and output length do not vary much. However, the length of the summary should be short compared to the length of document. Therefore, the one of the main problems in text summarization is to shorten the original document so that the main concept in the original document is preserved. The interrelation between words and documents are captured on a large scale in CNN [80] compared to RNN [36] since input sequences are represented hierarchically in multi-layered structures. A sequence-to-sequence model based on CNN called the Convolutional Sequence Model [80] (CSM) has been used in the text summarization approach. However, due to a lack of background knowledge, these models have a tendency to include unnecessary and grammatically incorrect information in the summaries that originate from the source document. Furthermore, the main topic might be overlooked because of this when generating the summary. This may lead to unconcise summaries that focus on irrelevant topics. Incorporating latent topic information into a text summarization model can ensure the relevant theme is discovered from the source document which is useful for generating a meaningful summary.

A traditional Latent Dirichlet Allocation (LDA) topic model [28] has been proven to achieve high accuracy as a text summarization model to uncover latent topics from documents [3] [113]. However, obtaining novel topics with only statistical models is clearly not sufficient. Conceptual information is a kind of external knowledge generated from a concept-based knowledge base which captures the latent semantic information of the text and provides contextual information. Using conceptual information in topic models is a potential solution to enrich the novelty of topics. Currently, text summarization research based on conceptual information and deep learning is limited. One of the important problems is handling rare words. The importance of a rare word is undermined since the researcher observes the rare word by counting the occurrences of the word. This is because they have no knowledge about that word which results in the failure to determine the importance of that word. Another disadvantage of these models is the ability of obtaining the syntactic structure in topic keywords. A good summary always has a strong association with the topic, document and summary, therefore, determining the associations among them is very important. To resolve these challenges, in contrast to most of the summarization models based on the RNN Seq2Seq model with the attention mechanism, we focus on obtaining topics based on conceptual information, called topic concepts, with a strong association among topic, document and summary, and summarize the document by incorporating topic concepts into the CSM-based summarization model.

In this chapter, we present a text summarization framework based on CSM with concept-based topic attention called Convolutional Sequence Text summarization with Concept-based Topic Triple Attention (TEXSCTTA). First, we generate topic concepts by incorporating conceptual information into the statistical topic models. Then, we incorporate topic concepts into CSM to map the salient knowledge and its alignment information by introducing TAM through contextual and topic concept attention. An example of the generated summary of a document by the TEXSCTTA model is shown in Table 5.1.

Research in this chapter has achieved the following:

- We propose a concept-based topic model (CTM) by integrating conceptual information into the LDA statistical topic models. This model generates salient

topic concepts efficiently which are more semantically relevant and informative for the input elements.

- We propose a CSM-based text summarization framework with concept-based topic attention (TEXSCTTA). This framework incorporates topic concepts which are retrieved using CTM into CSM. This model produces concise and rich summaries with salient information for a document.
- We introduce the triple attention mechanism (TAM) to measure the importance of each topic concept and source element to the output elements and the importance of each topic concept to the source elements which presents contextual information from three aspects and combines them using SoftMax activation to acquire the final probability distribution to assist the model to produce coherent and meaningful summaries with a wide range of rich vocabulary.
- We evaluate our proposed framework on datasets and compare them with various WSOTA such as Seq2Seq [100], PGEN [42], CSM-ATS [114] and TopicCSM [3]. The experiment results show that our model achieves a consistently better performance than WSOTA.

5.2. Related Work

Text summarization has been widely investigated from extractive [52] to abstractive summarization using deep learning. Extractive summarization has a drawback in respect to coherence, readability and providing concise information. Deep learning-based approaches such as NLP, data mining, image processing or video streaming have been proven to be very useful techniques in various research studies [50]. Abstractive summarization is the part of NLP research where deep learning-based text summarization achieves better results compared to extractive methods. Rush [40] first introduced neural networks to the text summarization approach. Bahdanau [78] proposed an approach for contextual alignment for machine translation which is applied to the text summarization approach. Two datasets, Gigaword and the CNN/Daily Mail datasets, have been used extensively to evaluate the summarization model and have achieved good results. Nallapati [36] proposed an RNN-based ses2seq+attention baseline text summarization model and

built the model based on the CNN/Dail Mail dataset which has been used as a base model of many summarization models to improve the existing models. Paulus et al. [45] improved the text summarization model by introducing an improved attention mechanism and by training the model with a reinforcement learning approach. Pasunuru [115] proposed a reinforcement learning approach into the base seq2seq model and utilized the multiple reward methods to improve the performance of the model. The base Seq2Seq RNN uses the soft attention mechanism to focus on the position of the important and relevant information segment of the document while generating summaries. The abstraction summarization model (ATM) generates semantically well-formed and human readable summaries. Most ATM models are built on the sequence-to-sequence framework (Seq2Seq) which maps the sequence of inputs to the sequence of outputs using RNN [36] [42] [112] [116]. Usually, this soft attention captures salient and relevant information from the document for each of the decoders in the generated summaries. In these types of models, the contextual information is captured at a syntactical level where the semantic-level information has been ignored. This affects the performance of the summarization output due to the lack of semantic information. The main concept or original meaning of the document should not be overlooked while generating abstractive summaries. Therefore, incorporating topic information into the ATS model can be effective in terms of providing good summary results. This is because the dependence of words and key information within the document are captured while generating the summaries. Only a few studies have been conducted which incorporate topic information in the ATS model [3] [113].

Gehring [5] first introduced CNN to the text summarization approach which has been successful in providing better results. Lin [117] utilized CNN to introduce a gated convolution unit to retrieve global information and reduce the chance of duplication. This model enables the system to compute the operation and locate contextual information in document in parallel. Convolutional neural networks tackle the challenges of training speed by leveraging parallel computing and achieving document-level inference, abstraction and paraphrasing by capturing long dependencies between words. However, using only a convolutional sequence model (CSM) in the text summarization approach [36] has proven less effective while generating incoherent summaries due to the failure to identify a novel topic. This is because only word-level attention is considered but the high-level semantic

structure of the input elements such as the topic and knowledge background of the document is not considered. There are several models such as text summarization with pre-trained encoders [43], BART [118], ProphetNet [119] and PEGASUS [120] based on pretrained objectives for abstractive text summarization and to evaluate datasets, however topic-based summarization where topics are retrieved using conceptual information has not been sufficiently investigated.

However, most of these models do not utilize the concepts in the document which can provide meaningful and informative information. Limited research has been conducted to utilize the meaningful information as concepts in the summarization model and scant research has been conducted to justify the performance of the convolutional architecture in the summarization approach. Higher-level attention in terms of conceptual information or knowledge could facilitate a model to generate effective results such as Context-Relevant Knowledge which is introduced into CNNs for text classification [121]. So, we introduce the topic model to provide topic information based on concepts and incorporate this information into the ATS model to resolve this problem. The CTM provides topic information which has a string association with the documents. This ATS model considers the relevance of the summaries, the topic information, and the document jointly while generating summaries. Therefore, we incorporate topic concepts in CSM through high-level attention.

5.3. Base Model

We use a base model Topic-CSM [3] to improve our TEXSCTTA model. In this section, we describe the mechanism of the base model Topic-CSM. This model is shown in Figure 5.1. It uses word embedding with topic information, a multi-layer convolution structure and multi-hop attention to incorporate topic information into the CSN-based ATS model. Multi-layer convolution constructs a representation of the elements in a hierarchical order. Then, this model passes the output of the last layer to a SoftMax classifier to predict a probability distribution over the target elements in the summary. This model uses the LDA topic model to obtain and pass topic information from LDA to CSN as additional input.

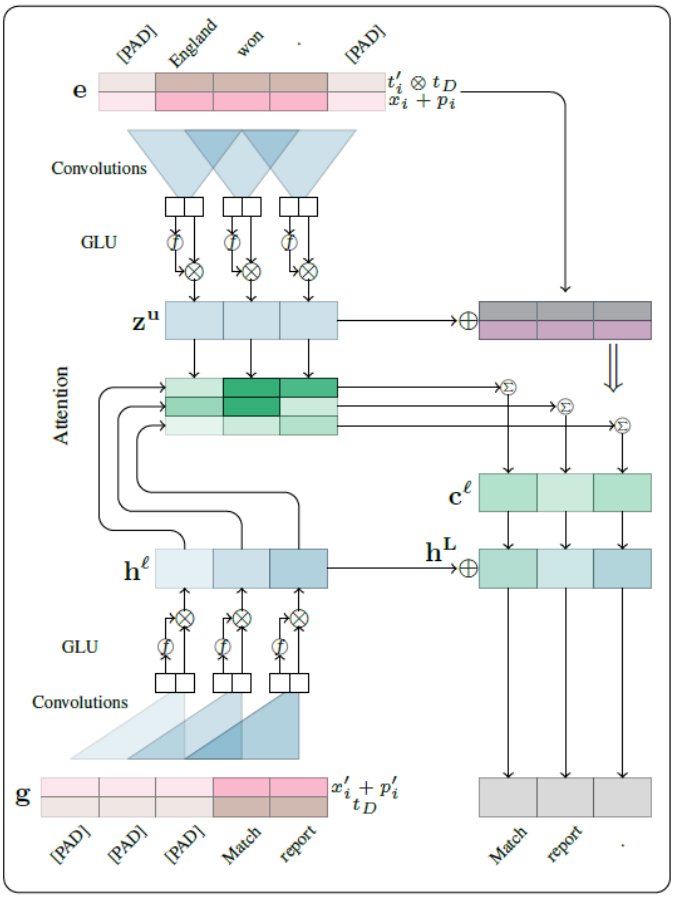


Figure 5.1: Convolutional model for the topic summarization model.

5.3.1. Words with Topic Embedding

Document d is embedded into distributional space, $x = (x_1, \ldots, x_m)$ along with their absolute word positions in document $p = (p_1, \ldots, p_m)$ where $(w_1, \ldots, w_m)$ are the sequence of words in the document. The topic distribution of document d is denoted as $t_d \in R^f$ and the topic distributions over words in the document is denoted as $t' = t'_1, \ldots, t'_m$.

This model incorporates topic information with the embedding of the word with its position via a representation $e = \{e_1, e_2, \ldots, e_n\}$ where e_i is as follows:

$$e_i = [(x_i + p_i); (t'_i \otimes t_D)] \in \mathbb{R}^{f+f'} \quad 5.1$$

where $\otimes$ is the point-wise multiplication. $\mathbb{R}^{f+f'}$ is the total (f+f') real number which is the dimension of the embedding matrix. The representation of the decoder for the prediction

of the output element, $g = (g_1, \ldots, g_n)$ for the earlier predicted target elements ($y'_1, y'_2, \ldots,$) and g_i is computed as follows:

$$g_i = \left[\left(x_i^{'} + p_i^{'}\right); t_D\right] \in \mathbb{R}^{f+f^{'}} \quad 5.2$$

where p' and x' are the embeddings of the word and position of the previously predicted element y_i and t_d denotes the topic distribution of document d.

5.3.2. Multi-Layer Structure

The k adjacent elements of the input are encoded to each convolution block and the concentration of the blocks represent the input $x \in \mathbb{R}^{k \times d}$ which is embedded in a d dimensional space and an output element of each block is denoted as $y \in \mathbb{R}^{2d}$. Gated linear units are used on the output of the convolution to transform them to y. The k output elements of the previous layer are operated through the successive layers and connect via residual connections which enable in depth hierarchical representation. The output of the ℓ-th layer is denoted as $\mathrm{h}^{\ell} = \left(h_1^{\ell}, \ldots, h_n^{\ell}\right)$ and $\mathbf{z}^{\ell} = \left(z_1^{\ell}, \ldots, z_m^{\ell}\right)$ for the decoder and the encoder network, respectively.

5.3.3. Multi-hop Attention

Multi-hop attention for each state captures the important context of each block by attending to the representation at the decoder and passing the output to the next upper block which assists the model to remember the histories of the attention of words. The attention a_{ij}^{ℓ} of state i and source element j is computed as:

$$a_{ij}^{\ell} = \frac{\exp\left(d_i^{\ell} \cdot z_j^{u}\right)}{\sum_{t=1}^{m} \exp\left(d_i^{\ell} \cdot z_t^{u}\right)} \quad 5.2$$

where the decoder state using current decoder state h_i^{ℓ} and the previous output element embedding g_i is as follows: $d_i^{\ell} = W_d^{\ell} h_i^{\ell} + b_i^{\ell} + g_i$. The output from the last encoder layer u is denoted as vector z^{u}. The conditional input c_i^{ℓ} to the current decoder layer is computed as follows:

$$c_i^{\ell} = \sum_{j=1}^{m} a_{ij}^{\ell}\left(z_j^{u} + e_j\right) \tag{5.3}$$

where e_j is the input element embeddings. This model incorporates the topic information through word embedding, does not utilize high-level topic attention and also does not consider concept information. We improve this model by introducing a high-level topic attention and utilize the concepts to capture the topic information while generating summaries.

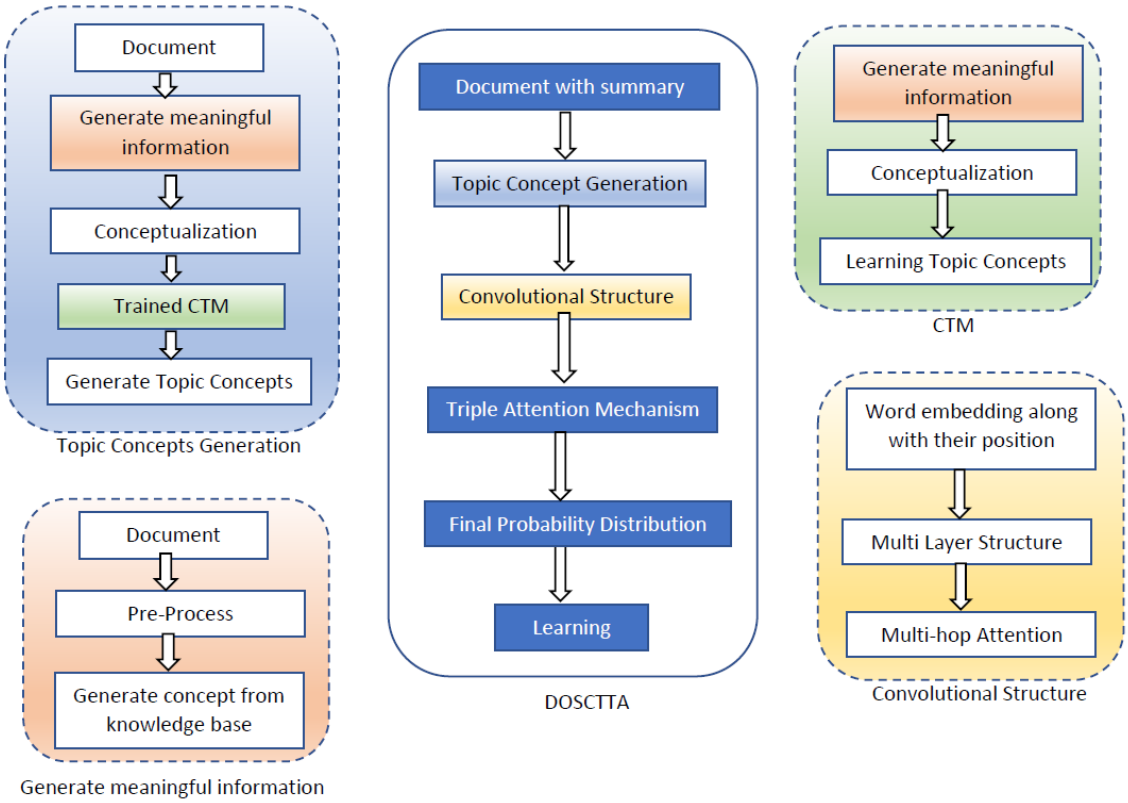


Figure 5.2: Architecture of the proposed TEXSCTTA model. Each subprocess is marked with a dashed line and a different color.

When a subprocess is used in another process, that subprocess is filled marked in the same color.

5.4. Architecture of TEXSCTTA

In this section, we present the architecture of our TEXSCTTA model. Figure 5.2 shows the architecture of TEXSCTTA. This architecture comprises four subprocesses: generate meaningful information, CTM, topic concept generation and convolutional structure. The

subprocess generate meaningful information has two steps: preprocess and generate concepts from the knowledge base. We apply the rule and algorithm used in chapter 4 to transform information into meaningful and structural information. The subprocess generate meaningful information provides the informative features of the document as a concept set to CTM. The concept-based topic model (CTM) consists of two steps: conceptualization and learning concepts. CTM receives the concepts from the meaningful information process, uses the conceptualization algorithm to rank the top-N concepts related to the documents, and incorporates this information into the LDA topic model to obtain topic information based on the concepts in the document (also called learning topic concepts). The topic concept generation process produces topic concepts from the trained CTM model. This uses the subprocesses generate meaningful information and conceptualization to retrieve the top-N relevant concept set for the document and produces the topic concepts from the trained CTM model. The convolution structure is the architecture of the CSN model which consists of three steps: word embedding with their position, multi-layered structure and multi-hop attention. First, the input and output are embedded at the encoder and decoder of CSN with their position, then the multi-layer structure is applied to represent the input and output elements in the hierarchical structure, and finally multi-hop attention computes the state of the encoder and decoder with their attention in the CSN.

We use the aforementioned subprocesses to build our proposed TEXSCTTA model. We generate the topic concept of the document using topic concept generation, we use the convolution structure to encode and decode the input and output elements and measure their attention, we introduce a tri-attention mechanism which utilizes the high-level topic attention to incorporate the topic concept into our model, we generate the final probability distribution to predict the next target element at the decoder of CSN, and finally, we use the reinforcement learning approach to maximize the performance of the model.

5.5. Text summarization Model with Concept-based Topic Triple Attention (TEXSCTTA)

In this section, we propose a text summarization framework, TEXSCTTA. This framework introduces an architecture based on CSM [80] with concept-based topic attention for a text

summarization model. This architecture consists of a convolutional structure, concept-based topic generation and attention, a triple attention mechanism (TAM) and a probability generation and learning procedure. The graphical illustration of text summarization with concept-based topic attention is shown in Figure 5.3.

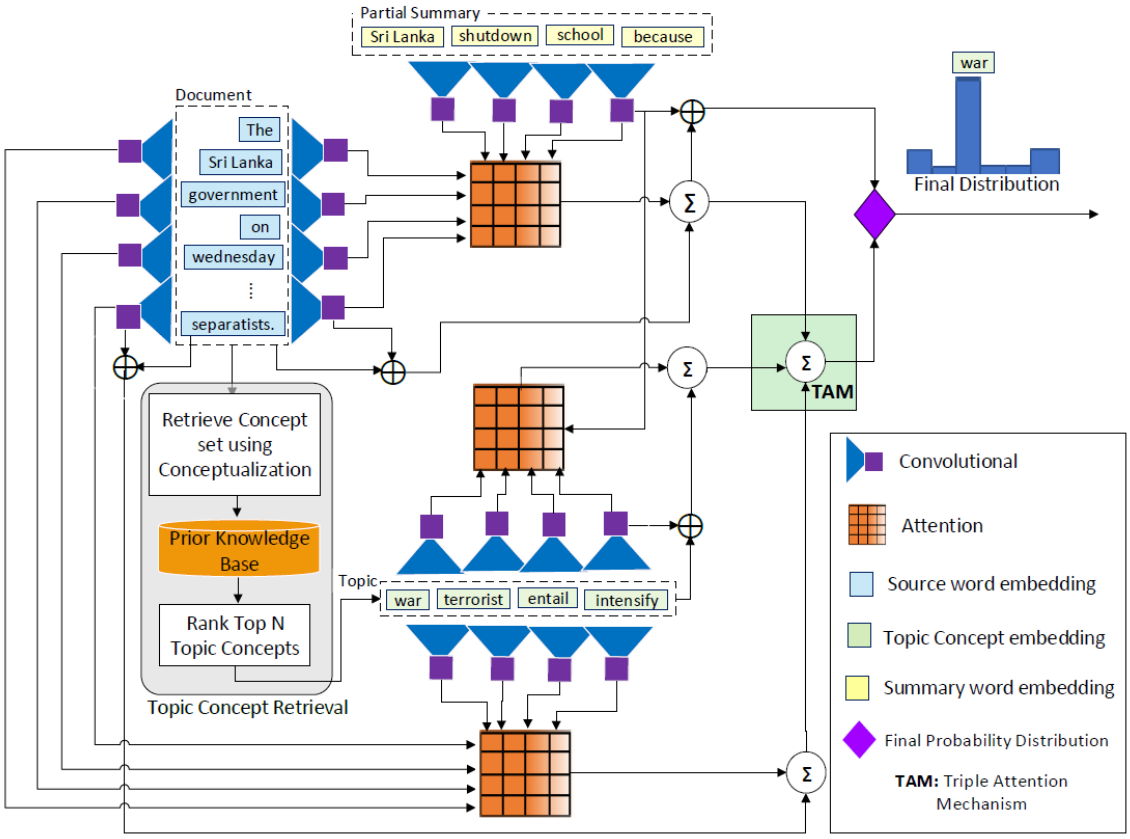


Figure 5.3: Text summarization Model with Concept-based Topic Triple Attention (TEXSCTTA).

The partial summary for the input segment of the source “Barak Obama on Wednesday ... separatists” is” Barak Obama shutdown school because” and the next target element that comes from the topic concepts in the output summary is” war”.

5.5.1. Convolutional Structure

We use the CSM architecture [80] as the base for our framework. We introduce word embedding and concept-based topic embedding and feed these embeddings into three CSM units in the architecture, along with position embedding, multilayer structure and multi-hop attention.

Word Position Embedding

We encode the absolute position of each word in the source document by adding position embeddings with word embeddings in the multilayer CNN architecture. Let document d be represented as a sequence of words d = (w_1, w_2, …, w_n) with a total of n words. We embed the input elements into distributional space $x \in \{x_1, x_2 \ldots, x_m\}$. $M \in R^{V\,xd}$ is an embedding matrix, $x_j \in R^{Vxd}$ is a row in M and vocabulary size is denoted as V. The absolute position p= (p_1, …, p_n) of the input elements in the document is embedded to preserve the order of the sequence. Finally, we represent the input elements along e = (e_1, e_2, …, e_n) by combining word and position embedding. That is e_i = (w_i; p_i) (i = 1, 2, . . ., n). Similarly, the output elements with m words generated by the decoder are represented along q = (y_i; $\bar{p}_i$) (i = 1, 2, …, m) and leads to the next step.

Multi-layered Structure

CNN applies multiple layers as units on top of each other to build a multi-layered hierarchical representation over the document. We call this the multilayered convolutional structure (MCS). MCS is applied in both encoder and decoder networks in a model. The output of the lth layer is defined as $d^{\ell} = (d^{\ell}_1, \ldots, d^{\ell}_n)$ at the decoder, and $e^l = (e^l_1, \ldots, e^l_m)$ at the encoder. In an encoder network, the state of a single block d^l_i contains information over kernel (k) input elements. A concatenation of k adjacent elements in d dimension, $x \in R^{kd}$ are provided in each convolution block using convolutional layers and these are mapped to an output element y $\in R^{2d}$. Y is represented as [A B] $\in R^{2d}$ and Gated Linear Units are applied over output y as

$$g([A; B]) = A \otimes \sigma(B) \qquad 5.4$$

In equation 5.4, the inputs to the non-linearity are denoted as A, $B \in R^d$, the sigmoid function as σ, point-wise multiplication as ⊗, and q ([A B]) $\in R^d$ as the output. A residual connection is applied to compute convolution unit i on the l-th layer and an example for the decoder is given as follows.

$$h_i^l = g\left(W^l\left[h_{\frac{i-k}{2}}^{l-1}, \ldots, h_{\frac{i+k}{2}}^{l-1}\right] + b_w^l\right) + h_i^{l-1} \qquad 5.5$$

Here the learnable parameters, kernel matrix and bias term are denoted as $W^l \in R^{(2dxkd)}$ respectively and $d^l_i \in R$. Finally, the last decoder output d^L_i is used to compute the next target elements y_{i+1} for K possible outputs using a SoftMax classifier:

$$p(y_{i+1}|y_1, \ldots, y_i x) = softmax(W_Y h_i^L + b_Y) \in R^T \quad 5.6$$

Multi-hop Attention

First, we measure the attention of the source elements towards the output elements. We use a multiple hop attention per time step to access previously attended words in the encoder and decoder for the output summary. We compute the summary of decoder state z^l_i to measure the attention as

$$z_i^l = W_d^l h_i^l + b_d^l + g_i \quad 5.7$$

where the current decoder state is denoted as d^l_i and the previous target element embedding is denoted as $g_i \in R^d$. Weight matrix $W^l_d \in R^{dxd}$ and bias $b^l_d \in R^d$ are the learnable parameters. Let v^{u0}_j be the output from the last encoder layer u_0 and Δ^l_{ij} is the attention of state *i* and input element *j*. The attention is measured as

$$\Delta_{ij}^l = \frac{\exp\,(z_i^l .\ v_j^{u_o})}{\sum_{t=1}^{m} \exp\,(z_i^l .\ v_t^{u_o})} \quad 5.8$$

The conditional input $c^l_i \in R^d$ to the present decoder layer is obtained as the sum of the output from the last encoder v^{u0}_j and the input element embedding e_j.

$$c_i^l = \sum_{j=1}^{m} \Delta_{ij}^l\ (v_j^{u_0} + e_j) \quad 5.9$$

Finally, we add c^l_i, to the output of the corresponding decoder layer d^l_i, delivered as part of the input to the next state d^{l+1}_i in the decoder.

5.5.2. Generating Meaningful Information with a Concept Set

In this section, we retrieve meaningful information about a term as a concept set from knowledge bases such as our OMRKBS or ConceptNet to understand the document. We take the example in Table 5.1 “The Barak Obama on Wednesday ... country”. First, we

tokenize the document and identify the terms or word list defined as $w = w_1, w_2, w_3, \ldots$, such as (Barak Obama, announce, closure, military campaign, escalated, …) from the document. We extract information for each term from knowledge bases such as our OMRKBS or ConceptNet. We discussed in chapter 4 how information about terms is built in OMRKBS. We use the same approach to transform the information into informative and meaningful features. First, we extract the information about a term from DBpedia, split the sentences, apply the rules, and build information about the term using the mapping algorithm. For example, the information about 'military campaign' is extracted from DBpedia as follows:

"A military campaign is a long-duration significant military strategy plans incorporating a series of interrelated military operations or battles forming a distinct part of a larger conflict often called a war."

Similarly, the information about 'escalate' is extracted from ConceptNet as follows:

"intensify", "war", "extent", scale, nuclear, and so on.

After splitting the sentences and applying rules, the features of the military campaign are represented in OMRKBS using the mapping expression algorithm detailed in 4.4.7. This helps us to understand the document since this provides unique features, and structural and concept information. We retrieve information about the concepts 'military campaign' and 'escalate' using the aforementioned steps as follows:

Military campaign
< military strategy, plans>
<incorporate, military, operations >
< incorporate, battles>
<form, conflict>
<war>
Escalate
<intensify>
<war>
<extent>
<scale>
<nuclear>

We generate a list of concepts $c = (c_1, c_2, c_3, \ldots, c_n)$ for each word w_i in the document using the approach detailed in 5.3.2 from knowledge bases such as Dbpedia and ConceptNet. Figure 5.4 shows how the 'military campaign' concept is defined in OMRKBS. The generated concepts for each word using the approach in OMRKBS are as follows: (1)

Barak Obama (w_1): country (c_1), state (c_2), ..., (2) announce (w_2): declare (c_1), advertise (c_2), …, (4) closure (w_4): shutdown, blockage, … (5) school (w_5): education, organization, …, (6) military campaign (w_6): war, campaign, entail war, ..., (7) against (w_7): protest, opposition, …, (8) Tamil (w_8): terrorist, human, .. (9) escalated (w_9): intensify (c_1), war (c_2), …, and so on.

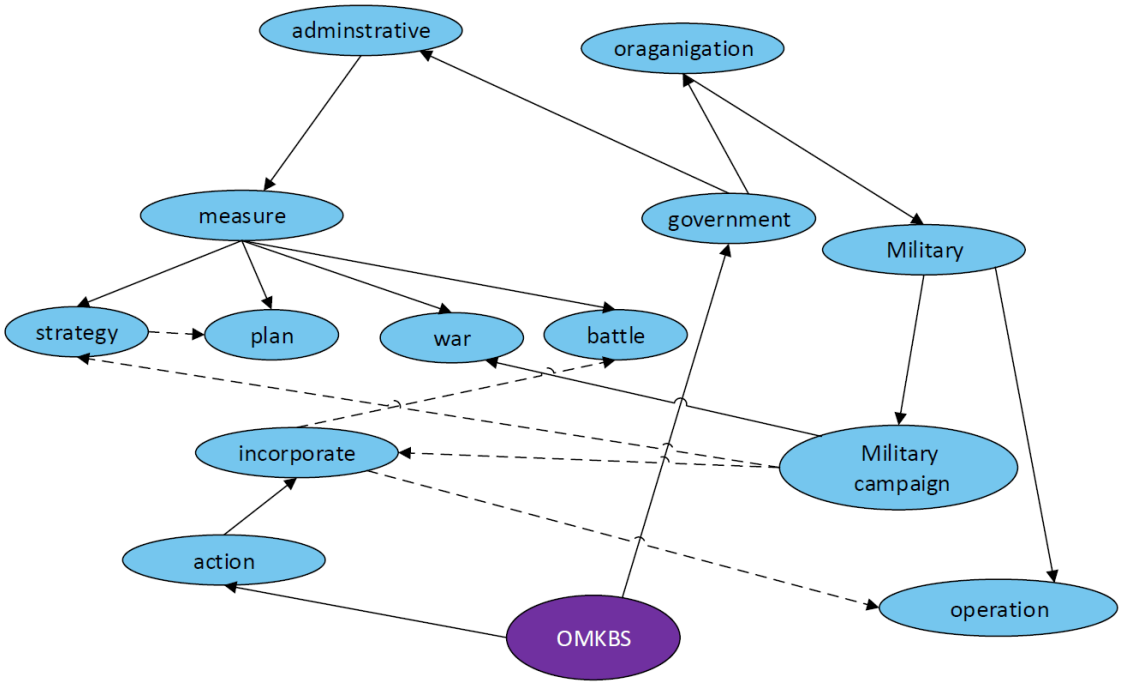


Figure 5.4: An example of a concept ('*military campaign*') defined by the proposed OMRKBS.

The words in the blue circles are concepts and the root is in the purple circle. The dashed arrows indicate the relationship between concepts according to the definition of a concept ('*military campaign*) while the solid arrows indicate subclasses.

5.5.3. Concept-based Topic Generation and Embedding

In this section, we propose a concept-based topic model (CTM) to obtain topic concepts based on conceptual information. Then, we embed the topic concepts.

Concept-based Topic Model (CTM)

This model uses conceptual information to discover the salient topics of documents. This model consists of preprocessing which retrieves information from the knowledge base and converts it into informative concepts of the document, and conceptualization which retrieves conceptual information from the knowledge base and learns the topic concepts

using this conceptual information based on the LDA topic model to produce the relevant topic concepts. For example, this model generates the topic concept "war" which is the latent knowledge of related words such as 'military campaign', 'separatists' and 'escalated' in the document from the example in Table 5.1.

Algorithm 1 Text conceptualization

Input: A novel document S
Output: List of concepts related to each word w in S
where count(w,c) is the number of cooccurrences of word w and concept c.

Obtain each sentence from novel S, denoted by $S_c = \{s_c\}$
for each $s_c \in S_c$ **do**
    Preprocess s_c sentence by the tokenization, removing stopword, lemmatized and stemming.
    draw each word from s_c, denoted by W={w}
    for each $w \in W$ **do**
        retrieve related the concepts $c_{wxr} \in C, R$ of w from Probase
        for each $c \in C$ **do**
            Calculate the probability of concepts $c \in C$ under the corresponding entity e in s_c, $P(c \mid w)$
        end for each
        Rank the top N probability for concept c_n from C for w
    end for each
    for each $w \in W$ **do**
        for $i \leftarrow 1$ to N **do**
            Calculate weight w_{c_i} of the concept c_i in s_c
    end for each
    Rank the top k weighted concept $c_k \in s_c$
    Obtain the list of top k concept with probability denoted $(c_k, p_k) \in w$ in s_c
end for each
return the list (c_k, p_k)

a) *Conceptualization:* We propose a conceptualization algorithm to derive the conceptual distribution for each word in the document using a conceptual knowledge base such as Probase [102], ConceptNet or DBpedia. Algorithm 1 shows the procedure of the proposed conceptualization to retrieve the conceptual information corresponding to each word in the document and to compute their distribution. We describe here how the algorithm works. We use the inverted indexing technique [141] to map the words into corresponding weighted sequences. First, we calculate the probability of word w in the document corresponding to the concepts, denoted by P(c|w) using equation (5.10).

$$P(c|e) = \frac{count(e,c)}{\sum_{c \epsilon C} count(e,c)} \qquad 5.10$$

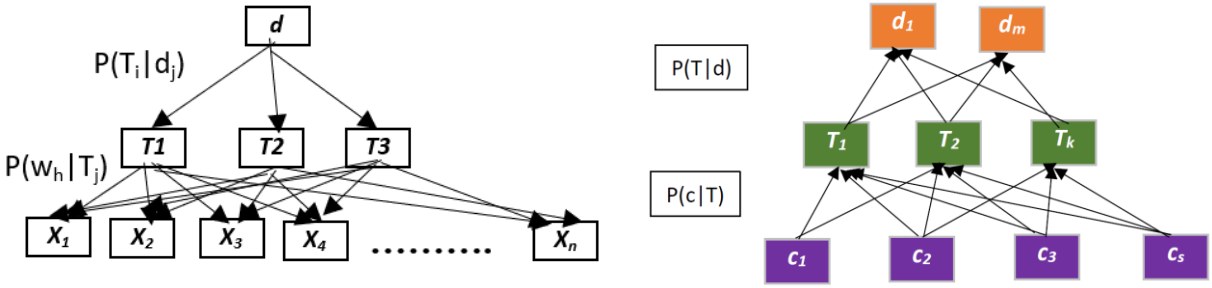
Then, the word vector range in the document is transformed to the concept range through weight vector w_c which represents the original document in the conceptualized range. Let $w_c = (w_{c_1}, w_{c_2}, .., w_{c_k})$, where each w_{c_i} represents the weight of concept c_i in the document, specifying the degree of the association of the concept and the document. This weight is measured using the following equation (5.11).

$$w_{c_i} = log \sum_{e_i \in T_i, Q_i \in Q_{c_i}} Q_i \times iDocf_c(e_i) \times iConf(c_i) \quad 5.11$$

$$iDocf_c(e_i) = log \frac{C_n}{N(e_i) + 1} \quad 5.12$$

$$iConf(c_i) = log \frac{C_n}{N(c_i) + 1} \quad 5.13$$

where Q_i is the probability of being mapped to word w_i by concept c_i; in other words, it is a set of the probability of relevant concepts which represent a word. The inverse document frequency $iDocf_c$ expresses the passive existence of the word in the concept and the inverse concept frequency *iConf* expresses the attention of the concept in the concept set, and we compute these using equations (5.12) and (5.13) respectively. We choose the N-highest concepts with a weight score for each document. We can see that 'military campaign', 'escalated', and 'terrorist' are related to the concept 'war'. The war concept is one topic information for this document.



5.5(a): Classic LDA-based topic model

5.5(b): Concept-based topic model

Figure 5.5: Comparison of the LDA and Concept-based topic model

b) *Learn Topic Concepts:* After conceptualization, we train the topic concepts from the conceptualization information using the LDA technique [28]. The statistical classic

LDA method obtains topics based on the source word of the document. We propose the CTM method to obtain topic-based concepts of the source documents. Figures 5.5(a) and 5.5(b) compare the classic LDA and our CTM model. We see from the figure that LDA determines the topics using word distribution per topic whereas our CTM determines the topics using concept distribution per topic. We retrieve these concepts using our conceptualization algorithm. LDA and CTM follow the same procedure to determine the topic distribution per document.

We used the Gibbs sampling technique to train our CTM model.

$$p(z_d = \mathrm{t} \mid \vec{z}_{-d}, \mathrm{c}, \alpha, \lambda) \; \alpha \frac{n_{d,t} + \alpha_k}{\sum_i^T n_{d,i} + \alpha_i} \frac{s_{t,\mathrm{c}} + \beta_{\mathrm{c}}}{\sum_i s_{t,i} + \beta_i} \qquad 5.14$$

where $n_{(d,t)}$ denotes the number of times topic t is used to present document *d*, and $s_{(t,c)}$ denotes the number of times concept c is used to represent topic *t*. This equation is used to find the best match for the concept set for each topic t and topic set for each document jointly. The first part of the equation finds the association between each topic in a document and the second part finds the association between each concept in a topic jointly. These topic concepts are used as prior topic knowledge and incorporated through a topic attention channel into CSM for our framework.

Dataflow of CTM

In this section, we describe how our CTM model works. Figure 5.6 shows the dataflow of the CTM model. The classic LDA model obtains vocabulary such as “Barak Obama”, “Wednesday”, “country”. Assume we have z topics; m documents and each topic contain k words. First, words from the vocabulary are assigned to each topic. The association between words in the topic and the association between topics in the document are measured using the first and second part of equation 2.2. After N iterations, the LDA topic is able to find the best match of word set for each topic and topic set for each document. However, our CTM model uses a more relevant concept set of the document rather than the word set when determining topics. For the example detailed in Table 5.1, CTM first retrieves the top n concept set using the conceptualization algorithm: war, shutdown, campaign, entail war, protest, opposition, terrorist, intensify, country, declare, and so on

and obtains the concept vocabulary. Assume we have z topics; m documents and each topic contain k concepts. Then, we assign k concepts to each topic to represent a topic from the concept vocabulary. Next, we find the association between the concepts and the topic using the second part of equation 5.14 and the association between the topic and the document using the first part of equation 5.14 and maximize the association jointly using equation 5.14. A concept such as 'war' which is more relevant to the words in the document is one of the topic concepts.

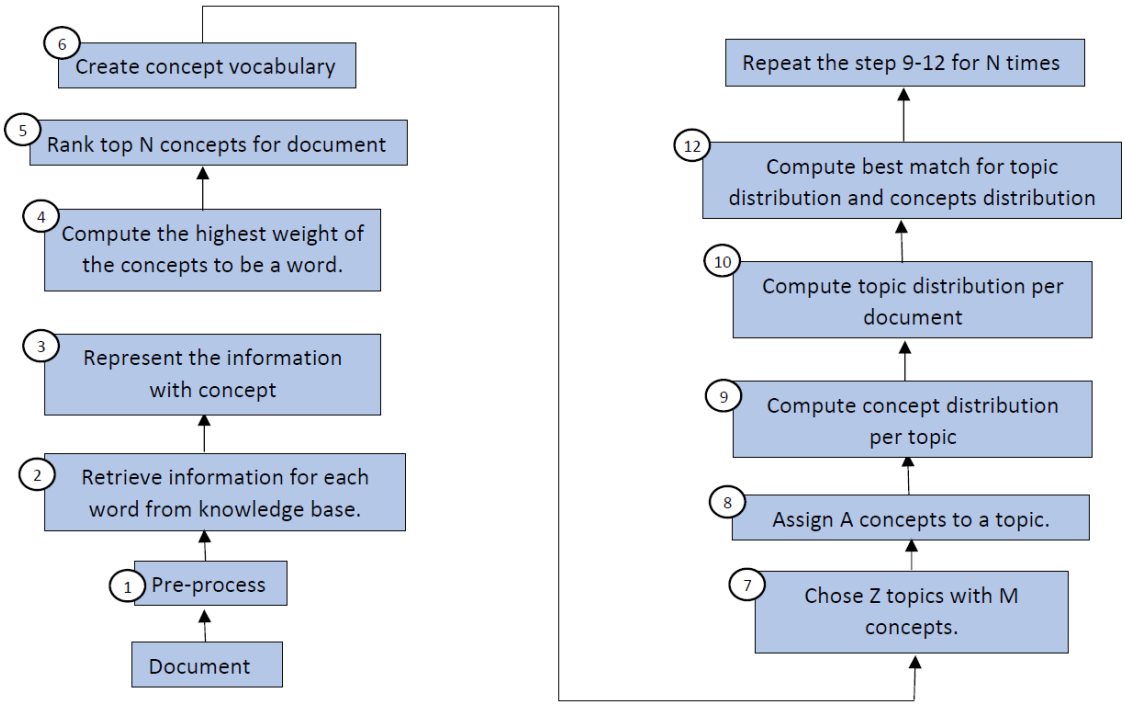


Figure 5.6: Dataflow of the CTM model.

Topic Concept Embedding

The conceptual information is retrieved from the knowledge bases for the given input elements. We obtain the topic concepts from this conceptual information using prior topic knowledge. The top *m* concepts are selected for each topic based on the highest probability words in the topic and these are embedded into the topic concept in the topic vocabulary. Given a concept set *c* of size m denoted as ($c_1, \ldots, c_m$) where c_i is the i-th concept vector, we aim to produce its vector representation topic embedding matrix P_{topic}. *V* is the vocabulary of the document and *T* is the vocabulary of the prior topic knowledge. A topic concept $c_i \in Rd$ is embedded as a row in the topic embedding matrix $P_{topic} \in R^{Kxd}$ when c_i

$\in T$. We embed the position of topic concepts and join these with the topic concepts elements at encoder n_t and decoder t_t to obtain a topic embedding respectively. After obtaining word and topic concept embedding, we compute the attention of these over the summary elements.

5.5.4. Triple Attention Mechanism (TAM)

We introduce a triple attention mechanism (TAM) to decide how much attention to pay to words and the corresponding topic concept of documents jointly at each decoder step. TAM jointly computes attention from three aspects: input words over summary elements, input words over topic concepts and summary elements over topic elements. Similar to 5.4, we compute convolution unit *i* on the lth layer in the decoder at the topic level for the output summary. The current decoder state, $a^l{}_i$ of the topic concept measure is embedded as $\hat{z}_i^l$.

$$\hat{z}_i^l = W_d^l a_i^l + \hat{b}_a^l + t_i \tag{5.15}$$

where $t_i \in$ Rd denotes the topic concept embedding of the previous decoded element. W_d^l and $\hat{b}_a^l$ are the learning parameters. Then, we use a similar method to measure $\hat{\Delta}_{ij}^l$ which is the attention of the topic concept *j* to the output decoder state *i* for a *l* layer using the multi-hop approach in equation 5.8.

$$\hat{\Delta}_{ij}^l = \frac{\exp\left(\hat{z}_i^l . \hat{v}_j^{u_t}\right)}{\sum_{t=1}^{m} \exp\left(\hat{z}_t^l . \hat{v}_t^{u_t}\right)} \tag{5.16}$$

where $\hat{v}_j^{u_t}$ is the output from the last encoder layer u_t and $\hat{z}_i^l$ is the embedding of the current decoder state, $a^l{}_i$ of the topic concept. We obtain the topic concept embedding k_t at the decoder state. $\bar{z}_i^l$ denotes the current decoder state of the topic concept measure and is embedded as

$$\bar{z}_i^l = W_h^l h_i^l + \bar{b}_h^l + k_i \tag{5.17}$$

where $k_i \in \mathrm{R}^{\mathrm{d}}$ denotes the topic concept embedding of the previous decoded element. Weight matrix $W^l{}_h$ and bias $\bar{b}_h^l \in \mathrm{R}^{\mathrm{d}}$ are learnable parameters. First, we incorporate topic concepts into CSM using the dual attention mechanism. The dual attention, $\overline{\Delta}_{ij}^l$ denotes the

weight of attention from i_{th} source elements and topic concepts to the summary j element at l-th state and is measured by the following equation.

$$\overline{\Delta}_{ij}^{l} = \frac{\exp\left(\bar{z}_i^l \cdot \bar{v}_j^{u_s} + \bar{z}_i^l \cdot \bar{v}_j^{u_0}\right)}{\sum_{t=1}^{m} \exp\left(\bar{z}_i^l \cdot \bar{v}_t^{u_s} + \bar{z}_i^l \cdot \bar{v}_j^{u_0}\right)} \quad 5.18$$

We call this CSM model with topic knowledge dual attention DTopicCSM where we incorporate topic concept information into CSM. This model attends to the topic concepts and input elements jointly over the output target elements to present more relevant topics for document summarization. However, this dual attention does not consider the relevance of the topics over the input elements. Therefore, we improved our TEXSCTTA model by introducing TAM. TAM adds one more attention to choose more relevant topics for the input elements. In TEXSCTTA, topic concepts are chosen over the output elements and the input elements while in DTopicCSM, topic concepts are chosen over the output elements only.

For this, instead of measuring dual attention $\overline{\Delta}_{ij}^{l}$ using equation 5.18, we measure the attention of the topic concepts over the source elements. We denote this attention as $\overline{\Delta}_{ij}^{l}$ which is the weight of attention from the i-th topic concepts to the j input elements at lth state. We compute attention weights $\overline{\Delta}_{ij}^{l}$ which measure a weighted sum of the concept vectors to the source elements and derive a semantic vector that represents the concepts. A larger $\overline{\Delta}_{ij}^{l}$ indicates the i-th concept is more semantically like the document.

$$\overline{\Delta}_{ij}^{l} = \frac{\exp\left(\bar{z}_i^l \cdot \bar{v}_j^{u_s}\right)}{\sum_{t=1}^{m} \exp\left(\bar{z}_i^l \cdot \bar{v}_t^{u_s}\right)} \quad 5.19$$

$$\bar{c}_i^l = \sum_{j=1}^{m} \overline{\Delta}_{ij}^{l}\left(\bar{z}_j^{u_s} + r_j\right) \quad 5.20$$

$\bar{v}_j^{u_s}$ denotes the output from the last encoder layer u_s for topic concepts. r_j denotes the embedding of the source elements to the topic concepts. $\bar{c}_i^l$ indicates the conditional topic concepts at the current decoder state. n_j is topic concept embedding. The topic concepts are incorporated into the model through a TAM. The triple attention weight is computed by

$$\rho = \frac{\exp\left(\alpha\Delta_i^l + \beta\bar{\Delta}_i + \gamma\hat{\Delta}_i\right)}{\sum_{t=1}^{m} \exp\left(\alpha\Delta_i^l + \beta\bar{\Delta}_i + \gamma\hat{\Delta}_i\right)} \quad 5.21$$

The embedding matrix P_{topic} is normalized from the final attention weights for each topic concept c:

$$P = \sum_{i=1}^{M} \bar{\rho}_i c_i \quad 5.22$$

The conditional input is computed as

$$\hat{c}_i^l = \sum_{j=1}^{m} \rho_{ij}^l \left(v_j^{u_t} + n_j\right) \quad 5.23$$

Finally, the three conditional inputs $c_i^l, \bar{c}_i^l$ and $\hat{c}_i^l$ are joined to the output of the corresponding decoder layer a^l and are fed back as input to a^{l+1}

5.5.5. Final Probability Generation

To derive the final probability distribution, first we transform the outputs of the top decoder for word portion, d^{L_0} and topic portion, a^{L_t} via a linear $\omega(.)$; which is computed by

$$\omega(d) = \varphi(W_0 d + b_0) \quad 5.24$$

where $W_0 \in R^{\text{Txd}}$ and $b_0 \in R^{\text{T}}$ are the learning parameters. Then, we generate the final distribution using the following equation.

$$p_{\text{final}}(\tilde{y}) = \frac{1}{Z}\left[\exp\left(\omega\left(d_i^{L_0}\right)\right) + \exp\left(\omega\left(a_i^{L_t}\right)\right) \otimes G_{\{w \in K\}}\right] \quad 5.25$$

where Z denotes the normalizer, the outputs of the word and topic at the *i*-th top decoder are defined as d^{L_0} and a^{L_t}, respectively, and *G* denotes the indicator vector of each candidate word or concept in y_{i+1}.

5.5.6. Learning

We train the TEXSCTTA with respect to mixed training [45] which associates the original maximum likelihood L_{ml} with policy learning L_{rl}. The mixed learning is computed using the parameter $\varphi \in [0,1]$ as follows:

$$L_{mix} = \varphi L_{rl} + (1 - \varphi) L_{ml} \quad 5.26$$

We train α, β and γ using a neural network. We use the following formula to calculate α, β and γ using the sigmoid function σ:

$$[\alpha, \beta, \gamma] = \sigma\left(W^t\left[\Delta, \overline{\Delta}_i, \widehat{\Delta}\right] + b\right) \quad 5.27$$

5.5.7. Dataflow of the Model

In this section, we describe the dataflow of the TEXSCTTA model. Figure 5.7 shows the dataflow of the model. We define three subprocesses to describe the dataflow: topic concept generation, multi-layer structures and multi-hop attention. Topic concept generation produces the topic concept from our trained CTM model. First, this model retrieves the related concept set for each word in the preprocessed document from the knowledge base, then ranks the top-N concepts from the concept set with the highest probabilities which are more relevant to the document, and then obtains the topic concepts from the learned CTM model with their probability information. Multi-layer structures embed the input and output elements in a hierarchical structure to represent the whole input elements of the encoder or decoder by stacking several blocks. First, each unit represents k elements of the input, then the GLU structure is applied to represent k elements of each unit as the single output. Finally, we stack all the units to represent all input elements. The process multi-hop attention is used to encode the input and output at the encoder and decoder state and measure the attention of the CSM. First, we compute each state of the encoder and decoder, measure the attention between the input and output elements using the state of the decoder and finally compute the conditional input of the encoder state.

We describe how the source document is streamed through the process to train our model. First, we obtain the embedding of the elements of the source document which are defined as $x = \{x_1, x_2, \ldots, x_n\}$ and the summary elements which are defined as $y = \{y_1, y_2, \ldots, y_m\}$. Next, we retrieve the topic concept using the topic concept generation process and obtain the embedding of the topic concepts of the document which are defined as $\{t_1, t_2, \ldots, t_m\}$. After this, we embed the source elements $\{x_1, x_2, \ldots, x_n\}$ and summary elements $\{y_1, y_2, \ldots, y_m\}$ at the encoder and decoder of a CSM, embed the source elements $\{x_1, x_2, \ldots, x_n\}$ and topic concepts elements $\{t_1, t_2, \ldots, t_m\}$ at the encoder and decoder of the second CSM, and embed the topic concepts elements $\{t_1, t_2, \ldots, t_m\}$ and summary elements $\{y_1\, y_2, \ldots, y_m\}$ at

the encoder and decoder of the last CSM. Then, we apply the multi-layer structure to represent the embedding elements of each CSM in the hierarchical structure.

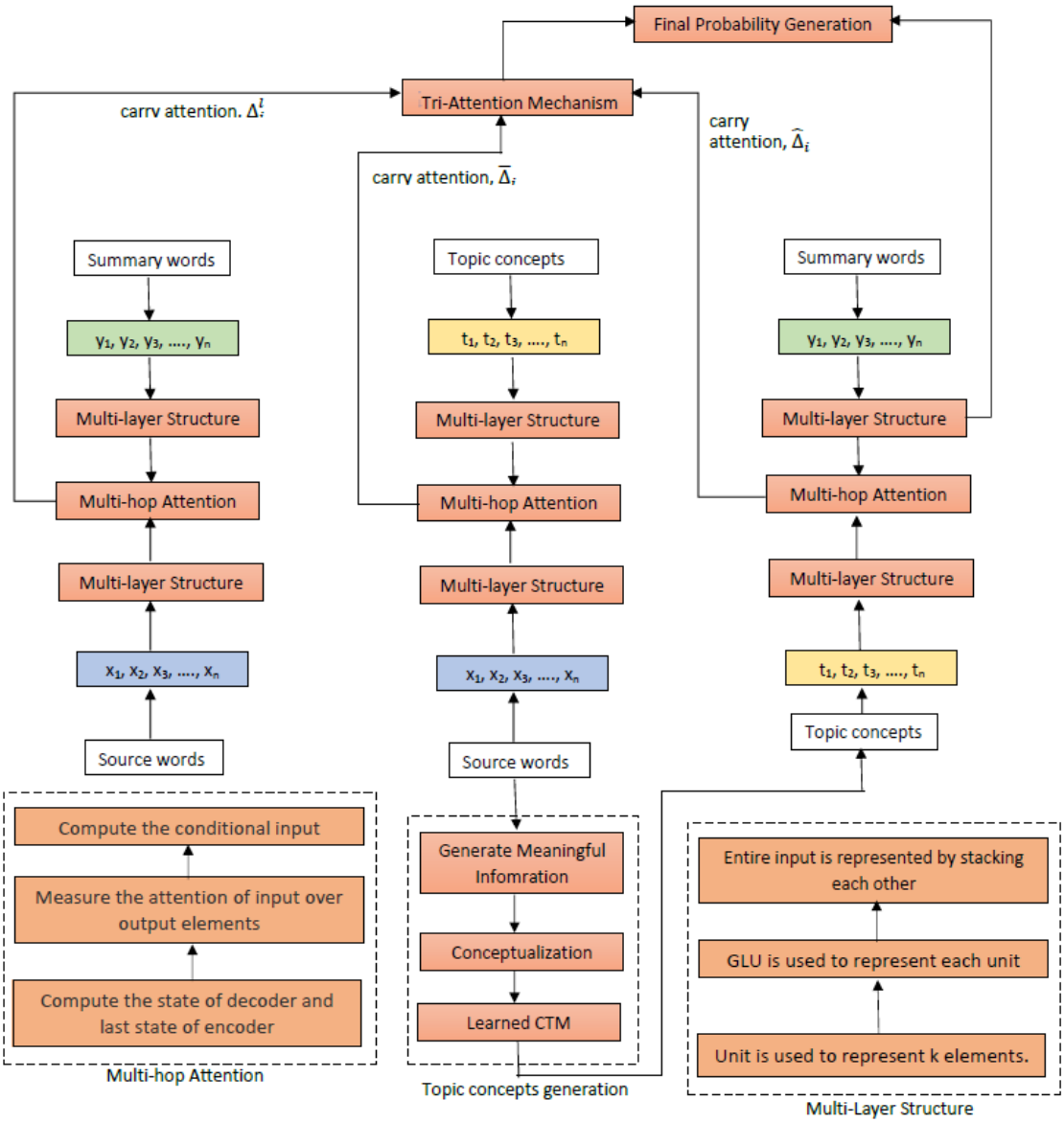


Figure 5.7: Dataflow of our TEXSCTTA model.

x_i, y_i, t_i are the embedding of the source, topic knowledge and summary elements.

Then, we measure the attention for a layer l of each CSM: Δ_i^l, attention of the source elements of the document $\{x_1, x_2, ..., x_n\}$ over the summary elements $\{y_1, y_2, ..., y_m\}$, $\overline{\Delta}_i$, attention of the source elements of the document $\{x_1, x_2, ..., x_n\}$ over the topic concepts elements $\{t_1, t_2, ..., t_m\}$ and $\widehat{\Delta}_i$, attention of the topic concepts elements $\{t_1, t_2, ..., t_m\}$ over the summary elements $\{y_1, y_2, ..., y_m\}$. The tri-attention mechanism combines all three

attentions Δ_i^l, $\overline{\Delta}_i$ and $\widehat{\Delta}_i$ for each layer and normalizes the attention using the SoftMax approach. The conditional input of each encoder state is computed for each CSM. Finally, the probability generation produces the probability distribution of the output elements at the decoder state of CSM to predict the next target element in the summary output.

5.6. Experiment

In this section, we describe the experiments on two datasets to assess the performance of TEXSCTTA. We describe the experiment setup including the datasets, the comparison model, parameters, and optimization, and compare our approach CTM and TEXSCTTA with the other methods LDA and WSOTA over the datasets.

5.6.1. Datasets

In the experiment, we use the Gigaword [123] and CNN/DM [42] datasets to evaluate TEXSCTTA and the various existing models in relation to the text summarization task. The Gigaword corpus is an English text summarization dataset where the summaries which are used for training are expressed along with the headline and the first sentence of the articles. We randomly split the Gigaword datasets into 95% training (380,0000), 4.95% validation (190,000) and .05 % (1951) test examples for assessment. The CNN/DM dataset comprises news stories from the CNN and Daily Mail websites and human written summaries, comprising more than 312,000 texts and corresponding summaries. This dataset is split into 90% training (280000), 9.5% validation (29650) and .5% (1560) test samples.

5.6.2. Comparison Model

We compare our proposed TEXSCTTA with the following WSOTA abstractive model: Seq2Seq [92] a neural sequence-to-sequence model with attention; and PGEN [42] a hybrid pointer generator model which can copy words from the source document. We compare our model step by step in the following order: CSM [80]: text summarization using a sequence model based on only a CSM [114]; TopicCSM [3]: a text summarization model

where topic information is incorporated into a CSM. The topic information in TopicCSM is obtained from the source document rather than conceptual information using the LDA technique. CSM+ Dual Attention (DTopicCSM): We proposed this DTopicCSM text summarization model in Section 5.5.4 which incorporates topic concepts through dual attention. CSM+ Triple Attention (TEXSCTTA): Our improved TEXSCTTA model measures attention jointly from three aspects: input elements and topic concepts over output elements, and topic concepts over input elements while DTopicCSM computes attention from two aspects: the attention of input elements and topic concepts over output elements.

Table 5.2: Example of the topics learned by CTM.

No	Topic Words
1	color, text, contrast, brightness, screen, sharp, resolution, image, picture
2.	majority, victims, foreign family, politicians, committee, tactics
3.	reason, people, follow, belief, moral, simply, proof, sources
4.	teams, looked, biased, member, biggest win, worst, losing scorers
5.	batteries, noticed outlets, detecting, screw, wires, existing, fail

5.6.3. Evaluation Method

We use topic coherence to evaluate topic results and measure topic coherence using the semantic coherence method introduced by Mimno [124]. ROUGE [125] is an evaluation method which is widely applied to summarization evaluation. This method measures the quality of the generated summary by comparing the summary against the reference summary. The comparison is performed by counting the number of overlapping unigrams, bi-grams and the longest common subsequences which are called ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) respectively. In our experiment, we use different ROUGE metrics to evaluate our model using the pyrouge package.

5.6.4. Parameter and Optimization

To evaluate LDA and CTM, the number of iterations for each Gibbs sampler algorithm is set to 1000, and both the initial hyperparameters α and β to 0.01. We set the batch size to 56. For Seq2Seq and PGEN, we set the dimensions of the hidden states and word

embeddings to 128 and the learning rate to 512 and .20 respectively. For our models, DTopicCSM and TopicCSM, the number of layers of the convolutional network is set to six for both the decoder and encoder and the dimensional word and position embeddings are set to 256. We set the learning rate to 0.25 and used the 256 dimensionalities for layer mapping between the hidden and embedding states. When the validation ROUGE score does not change or increase after each epoch, the learning rate is reduced by a decay rate of 0.1 for a more accurate score until the learning rate declines to 10^{-5}. We set the scaling factor for mixed leaning λ to .9. All models are implemented with Python and trained on a GPU cluster. The concept based LDA topic model was trained on Gigaword datasets documents. A probability distribution over the topic concepts for each word is measured and 512 topics with the best results are obtained.

Table 5.3: Topic coherence for a different number of words in topics.

	CNN/Daily Mail Datasets			Gigaword Datasets		
N	5	10	15	5	10	15
LDA [28]	-210.75	-960.69	-2488.65	-260.68	-1298.43	-3352.56
CTM	**-172.52**	**-903.12**	**-2295.73**	**-201.23**	**-1148.30**	**-3174.63**

Higher scores are marked in bold. N represents the number of words in the topic.

5.6.5. Topic Results on Datasets

We measure the strength of the semantic similarity between words in the topic which is topic coherence. We compare the results of the topic coherence of our model with LDA. Table 5.3 shows the topic coherence score for the CNN/Daily mail and Gigaword datasets. We can see from the topic coherence results, our model CTM achieves a higher score than LDA which demonstrates the concepts or words in the topic are more semantically coherent than the tradition statistical LDA [28] topic. Table 5.2 shows an example of the topics learned by CTM.

5.6.6. Summarization Results on Datasets

In our experiments, we evaluated our model with WSOTA over two dataset and measured the R1, R2, and RL metrics. We systematically investigated the performance of our TEXSCTTA model step by step. Table 5.4 and Table 5.5 show the experiment results for the R1, R2, and R3 metrics over the CNN/DM and Gigaword datasets. First, we tested the

Table 5.4: R1, R2, and RL scores on the CNN/Daily datasets for various models and TEXSCTTA.

Models	ROUGE1	ROUGE2	ROUGEL
Seq2Seq [36]	32.60	15.31	30.62
PTGEN [42]	35.45	16.55	32.73
ATS-CSM [114]	35.82	17. 51	33.26
Topic-CSM [3]	36.42	17.61	33.35
CSM+ Dual Attention (Our)	36.93 (1.4% ↑)	18.45 (7.7% ↑)	34.33 (1.3% ↑)
CSM+ Triple Attention (Our imp.)	**37.56** (**1.3**% ↑)	**18.92** (**2**% ↑)	**35.02** (**2.1**% ↑)

The best scores are marked in bold.

Table 5.5: R1, R2, and RL scores on the Gigaword datasets for various models and TEXSCTTA.

Models	ROUGE1	ROUGE2	ROUGEL
Seq2Seq [36]	35.39	13.35	32.62
PTGEN [42]	39.49	17.31	36.31
ConvS2S [114]	39.80	17.23	36.62
Topic-CSM [3]	40.29	17.61	37.12
CSM+ Dual Attention (Our)	40.87 (1.7% ↑)	18.97 (4.5% ↑)	37.62 (3% ↑)
CSM+ Triple Attention (Our imp.)	**41.39** (**1.7**% ↑)	**19.34** (**2.5**% ↑)	**38.43** (**2**% ↑)

The best scores are marked in bold.

base CSM structure and TopicCSM [52]. We can see from Table 5.4 and Table 5.5 that the TopicCSM method achieves better results for the R metrics score than CSM since TopicCSM incorporates topic information in the CSM model. However, TopicCSM does not consider the conceptual information to incorporate topics in the model. Then, we tested our model DTopicCSM (CSM + Dual Attention) which improves the results since DTopicCSM utilizes conceptual information while incorporating topics. This demonstrates that topics based on conceptual information produce better results since conceptual information provides latent knowledge about source documents which cannot be found in source documents. Lastly, we evaluated the performance of the proposed model TEXSCTTA (CSM+ Triple Attention) which uses TAM to incorporate more relevant topic concepts into the input elements. It is clear from Table 5.4 and Table 5.5 that our improved TEXSCTTA model achieves better results than DTopicCSM since DTopicCSM does not consider the relevancy of topic concepts over the input elements.

We can see from these tables that TopicCSM which incorporates topic information only in the base CSM model improves the score of the R metrics more than CSM. Incorporating topic concepts in DTopicCSM achieves better results than TopicCSM. This demonstrates

that topics based on conceptual information produce better results since conceptual information provides latent knowledge about source documents which cannot be found in the source document. DTopicCSM joins the two multi-hop attentions which pay attention to the input elements and topic concepts at the encoder over the output elements at the decoder while TEXSCTTA has one more attention than DTopicCSM which enables it to generate more relevant topic concepts in terms of the input elements. We see from Table 5.4 and Table 5.5 that the TEXSCTTA model achieves better scores for R1, R2 and RL than DTopicCSM since the DTopicCSM model does not consider the relevancy of topic concepts over input elements. This proves that incorporating more relevant topics in terms of source elements improves the performance of the summary results based on R metrics.

We further evaluated the various summarization models against our proposed model. Table 5.4 and Table 5.5 show the R1, R2 and RL scores of the PGEN, CSM, base TopicCSM and DTopicCSM over Gigawird and CNN/Daily Mail dataset. The results show that the concept-based topic attention and the mixed learning procedure improve the quality of text summarization in terms of accuracy. Moreover, the TEXSCTTA model achieves the best scores for R1, R2 and RL metrics. We compare the generated summaries with the reference summaries using CSM, TopicCSM and DTopicCSM. Examples are shown in Table 5.10. We can see that after the concept- based topic model is merged, some topic concepts which are not in the reference summaries, or the source document correctly appeared in the generated summaries. So, we can say that topic concepts using the triple attention passes more informative knowledge and increases the distinction and coherency of the summarization.

5.6.7. Effect of Documents of Different Lengths

We tested the performance of TEXSCTTA on the CNN/DM dataset for source documents of different lengths. We divided the test dataset into three groups in terms of the number of words in the document: short (<250 words), medium (251-400 words) and long (>400 words). We can see from Figure 5.8 that TEXSCTTA achieves the best performance for documents which are short in length. This model achieves a better performance for short- and medium-length documents compared to long documents.

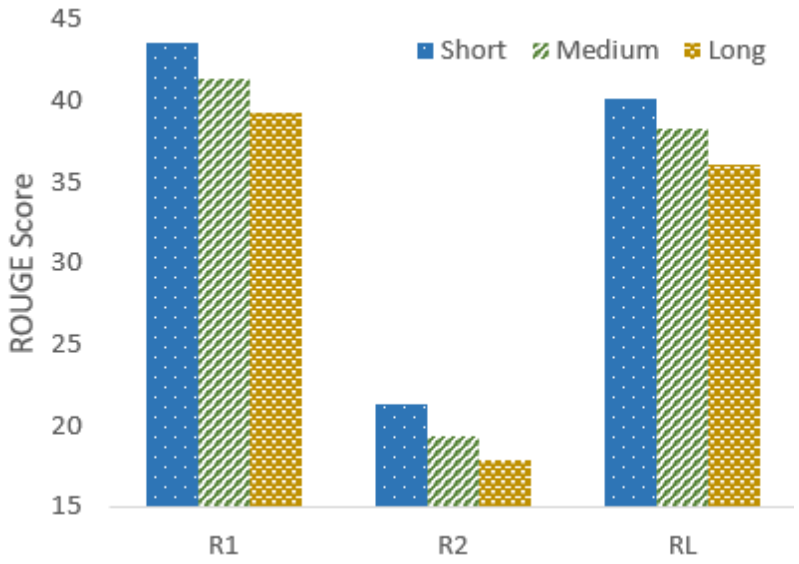


Figure 5.8: R1, R2 and RL scores of TEXSCTTA on CNN/DM datasets for documents of different lengths

5.6.8. Effect of the Attention Mechanism

TopicCSM incorporates topics in CSM however it does not utilize high-level attention whereas our DTopicCSM (CSM+Dual attention) model uses dual attention to integrate topic concepts which jointly pay attention to input elements and topic concepts at the encoder for the output elements at the decoder. We observe from Tables 4.4 and 4.5 that our model improves the results of the R1, R2 and RL score by 1.3/%, 7.7% and 1.3 % respectively compared to TopicCSM on the CNN/DM datasets. This implies that incorporating topic concepts using high-level attention is effective. Our improved model TEXSCTTA (CSM+Triple Attention) uses one more attention than the DTopicCSM model to produce more relevant topic concepts to the input elements. We can see that our TEXSCTTA improved the R1, R2 and RL scores by 1.7%, 2.5% and 2% respectively. This proves that incorporating more relevant topics in terms of source elements through TAM improves the performance of the summary results.

5.6.9. Human Evaluation

Evaluating the model only with an automatic method using the ROUGE score may provide a misleading justification of the model's performance because this automatic method

assesses the summaries only in terms of the informative features, not the semantic features. Therefore, it is important to evaluate the summary output using human judgment. We first randomly selected 50 articles from the test dataset and asked five participants, all of whom are university students with expertise in computer science technology, to compare the summaries generated from RNN-S2S, PTGEN, ATS-CSM, and the human-written summaries referred to as gold summaries in relation to two aspects: I and F. I is the evaluation of the informativeness summaries (when the summary captures topic information from the document) and F is the evaluation of the fluency summaries (when the summary is coherent with a good syntactic structure).

Table 5.6: Results of human evaluation over Gigaword datasets.

Model	I	F
Seq2Seq [36]	2.51	2.2
PTGEN [42]	2.93	3.12
ATS-CSM [114]	3.02	2.98
Topic-CSM [3]	3.33	3.15
DTopicCSM	3.55	3.40
TEXSCTTA	3.72	3.65

'I' indicates the score from the informative aspect and F indicates the score from the fluency aspect.

Table 5.7 Results of human evaluation over CNN/DailyMail datasets.

Model	I	F
Seq2Seq [36]	2.75	2.82
PTGEN [42]	3.11	3.22
ATS-CSM [114]	3.15	3.25
Topic-CSM [3]	3.42	3.39
DTopicCSM	3.62	3.52
TEXSCTTA	3.82	3.74

'I' indicates the score from the informative aspect and F indicates the score from the fluency aspect.

In relation to the I evaluation, the participants are asked to read the entire document and its summary carefully, understand the document, and highlight the important and salient information from the document. Then, we asked the participants to give a higher score to the summary output which contains similar salient information as the document. In relation to the F evaluation of the summaries, we asked the participants to give a lower score to the output summaries which are not readable or contain grammatical mistakes. The participants are asked to assign a mark to each summary for both informativeness and fluency ranging from 1 to 5: 1 (unsatisfactory), 2 (limited), 3 (satisfactory), 4 (good), and

5 (better). Tables 5.6 and 5.7 show the results of the human evaluation for the summarized document for WSOTA and our model. We can see from the results that TEXSCTTA outperforms WSOTA in relation to both the I and F aspects, which shows that our model is able to capture salient and meaningful information from the document and produce high-quality human-like summaries.

5.7. Discussion

In this section, we discuss the characteristics of the model and then compare the characteristics with the WSOTA. We utilize reinforcement approaches to maximize our learning, fine-tuning technique and the application of our model.

5.7.1. Characteristics of Our Model

We describe the individual attributes of our TEXSCTTA model. Table 5.8 shows the characteristics of our model. The first characteristic of our model named "A" is that this model retrieves meaningful and informative features as concepts from the knowledge base. The next characteristic B is that our model uses a conceptualization algorithm to produce more relevant concepts related to the document to provide this information in the topic model. Characteristic C generates topic information based on concepts by incorporating the concepts from characteristic B in the model to provide latent and important information in the topic information. Next, we use the convolution network in the summarization model instead of the RNN model to take the advantage of parallel computation and the large-scale representation of the input. Characteristic E is that our model incorporates the topic information in the summarization model to generate relevant summaries to the source document. Characteristic G enables our TEXSCTTA model to utilize the informative and meaningful information as a concept in the summarization model while identifying the topics in the generated summaries. Then, we use high-level topic attention to pay attention not only to the source word but also to the topic information. This characteristic helps the model to focus on topic elements while generating summaries. Our tri-attention channel combines the attention from three aspects to more closely associate the source and the topic elements to the summary. The concept-based topics are obtained using meaningful

information and knowledge and a conceptualization algorithm which assists the model to generate rich and meaningful relevant information in the summary. Finally, the combination of all the characteristics enables the model to produce coherent, meaningful, and human-like summaries.

Table 5.8: Characteristics of the TEXSCTTA model.

Name	Characteristics of our model
A	Retrieve meaningful features from knowledge base
B	Produce more relevant concepts
C	Use this conceptual information in the topic model
D	Utilize CSM in the summarization model.
E	Incorporate topic information in the summarization model
F	Utilize conceptual information through the CTM model in the summarization model.
G	Use high-level attention
H	Introduce a tri-attention mechanism
I	Produce rich and meaningful relevant information in the generated summaries.
J	Generate coherent, meaningful, and human-like summaries.

Each characteristic is referred to by the corresponding letter in the first column.

5.7.2. Comparison of Our Model's Characteristics with WSOTA

In this section, we compare the characteristics of our model with WSOTA, as shown in Table 5.9. We can see from the table that Seq2Seq does not support any individual characteristics of our model. Seq2Seq, which is based on RNN, does not utilize the conclusion structure, topic information, conceptual information from the knowledge base, or high-level topic attention which results in this model generating irrelevant and incoherent summaries. PTGEN utilizes characteristic G, high-level attention in terms of copying the word from the source document but does not have characteristics such as incorporating knowledge information in the topic information, taking advantage of the CSM and so on.

ATS-CSM is a CSM-based summarization model which utilizes characteristic D to train the parallel model and capture the large-scale input range. However, this model does not

Table 5.9: Comparison of the characteristics of our model with WSOTA

Model	A	B	C	D	E	F	G	H	I	J
Seq2Seq [36]	×	×	×	×	×	×	×	×	×	×
PTGEN [42]	×	×	×	×	×	×	✓	×	×	×
ATS-CSM [114]	×	×	×	✓	×	×	×	×	×	×
Topic-CSM [3]	×	×	×	✓	✓	×	×	×	×	×
Our model										
DTopicCSM	✓	✓	✓	✓	✓	✓	✓	×	×	✓
DOCSTTA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓ indicates that the model has the corresponding feature and × indicates that the model does not contain the corresponding feature. The letters show the characteristics of our model as explained in section 5.5.1.

incorporate topic information nor does it utilize high-level attention. Topic-CSM has D and E characteristics: it uses the CSM architecture and incorporates the topic model. However, this model only has these two characteristics, and it does not have other important characteristics such as utilizing high-level attention and concepts, nor does it associate the document, source and topic concepts well. DTopicCSM is our proposed model to incorporate topic information based on concept information with high-level topic attention. We can see from Table 5.9 that DTopicCSM has most of the characteristics of our model which helps the model to generate relevant summaries with rich information. This model also includes important relevant topic information in the summaries. However, this model does not utilize the tri-attention mechanism which associates the topic with the source well. As a result, this may sometimes result in failure to determine the most closely associated topic information with the document in the summaries. So, we improved our model by introducing the tri-attention channel which results in the model generating more relevant summaries for the document with large word diversity and which are close to human-written summaries.

5.7.3. Application of the Text Summarization Model

Our summarization model can be applied in the field of data mining and analytics applications such as the retrieval and extraction of information or user queries etc. The TEXSCTTA model can be effectively utilized in the search engines while retrieving information techniques to improve the performance of the results of the user query. There are various types of documents such as articles, books, news, blogs, emails, opinion

Table 5.10: Examples of the summaries generated by the CSM and TEXSCTTA models over the Gigaword datasets.

Source Text: At least 100 people were killed when a Nigerian airways airliner crashed on landing Monday at Kaduna airport in the north of the country, airport officials said. **Reference Summary:** Nigerian plane crashes on landing killing at least hundred. **TopicCSM:** A number of people have been killed in a car crash in the republic of Ireland, officials say **TEXSCTTA:** Hundred people have been killed in a plane crash in northern Nigeria, officials have said
Source Text: Greece said Thursday that it was monitoring the escalating hostilities between Israelis and Palestinians with particular concern and feared it could worsen. **Reference Summary:** Greece worried over escalation of Mideast hostilities **TopicCSM:** Greece extremely concerned for the future between Israelis and Palestinian **TEXSCTTA:** Greece worried as war escalation between Israelis and Palestinian
Source Text: A 26-year-old man sustained facial injuries during the incident on a number 9A bus travelling on Paisley Road, Tradeston , towards Penilee. The disturbance happened at about 20:00 on 20 December. Police described the man they are looking for as white, in his 50s , of heavy build. He has blotchy skin and a shaved head. He was wearing a hooded black body-warmer with a blue zip-top underneath. Officers have asked anyone who recognizes the man or has any further information to contact them. **Reference Summary:** Police have released CCTV images of a man they want to speak to following a racist assault on a Glasgow bus **TopicCSM**: Police have released CCTV images of a man they want to trace in connection with a serious assault in Edinburgh. **TEXSCTTA:** Police have released a cctv image of a man description to find the connection with a serious assault in paisley.

Summary words which are in the same colour as the document words means that these summary words are generated from the topic concept and are retrieved using related document words. They do not come from the source text or the reference summary.

reviews which combine text summarization with medical documents and legal documents, etc. Examples of different text summarization applications are given below. News or Article Summarization: We can apply our summarization model to summarize the news or articles which assist people to find the information they are looking for. Every day, search engines summarize the news and then cluster similar news from several news sites which the user looking at. Sentiment analysis (SA) is the task of detecting, extracting and classifying opinions or emotions on products or events. Our summarization model can be used in sentiment analysis techniques by shortening the content and classifying the sentiment. Our model can be applied to email summarization from unstructured and spam

messages to classify them in terms of what type of email they are. We can apply our method to summarize medical data, legal documents and so on.

5.8. Summary

In this chapter, we dealt with the issue of irrelevant document topics in text summarization. We investigated this issue step by step from a convolutional sequence model to proposing a concept-based topic attention convolution sequence framework which borrows insights from concepts or knowledge for topic identification and incorporates this information through a systematic mechanism by introducing a triple attention among the source element to the topic concept, based on topic information and source/ target elements. Through an extensive set of experiments, we verify that our proposed mechanism advances some high-level semantic information for summarization by capturing information on the local context in a way which surpasses the WSOTA. In this chapter, our focus was acknowledging the concept or knowledge of the words in the document while capturing topics in the abstractive summarization approach. In the future, we will focus on utilizing a pre-trained model such as BERT [43] in our model.

Chapter 6.

Joint Knowledge-based Topic Level Attention to a Convolutional Text summarization Model

Abstractive text summarization (ATS) often fails to capture salient information and preserve the original meaning of the content in the generated summaries due to a lack of background knowledge. In this chapter, we present a method to provide the topic information based on the background knowledge of documents to a deep learning-based summarization model. This method comprises a topic knowledge base (TKB) and convolutional sequence network-based text summarization model with knowledge-powered topic level attention (KTOPAS). TKB employs conceptualization to retrieve the semantic salient knowledge of documents and the knowledge-powered topic model (KPTopicM) to generate coherent and meaningful topic information by utilizing the knowledge that represents the documents well. KTOPAS obtains knowledge-powered topic information (also called topic knowledge) from TKB and incorporates the topic knowledge into the convolutional sequence network through a high-level topic level attention to resolve the existing issues in ATS. KTOPAS introduces a tri-attention channel to jointly learn the attention of the source elements over the summary elements, the source elements over topic knowledge, and topic knowledge over the summary elements to present the contextual alignment information from three aspects and combine them using the SoftMax function to generate the final probability distribution which enables the model to produce coherent, concise, and human-like summaries with word diversity. By conducting experiments on datasets, namely CNN/Daily Mail and Gigaword, the results show that our proposed method consistently outperforms the competing baselines. Moreover, TKB improves the effectiveness of the resulting summaries by providing topic knowledge to KTOPAS and demonstrates the quality of the proposed method.

6.1. Introduction

Automatic text summarization is a process of producing a brief statement on an original text that retains the overall meaning of the content. The key challenges of this

summarization are to properly assess the content, identify the salient information, convey the intended meaning and generate a concise summary. Humans summarize text by reading it in entirety, developing an understanding of the meaning of the content and highlighting the main features. Since machines have a limited ability in terms of human knowledge and their ability to understand language, automatic summarization is very challenging. Generating a summary by interpreting and apprehending the content using background knowledge which is qualitatively close to human-written sentences motivates our interest in this research direction.

Figure 6.1: An example from our summarization result of the KTOPAS model.

DOCUMENT: Barak Obama on Wednesday announced the closure of government schools with immediate effect as a military campaign against religious separatists escalated in the north of the country. **SUMMARY**: America shutdown school because war escalated in the north of the country.
DOCUMENT: A fairly large earthquake measuring a magnitude of 6.7 on the Richter scale rocked wide areas of central and western Japan Sunday, followed by four aftershocks, the meteorological agency said **SUMMARY**: Powerful earthquake shakes the wide area of Japan.

The colored concept 'shake' in the summary comes from the topic words that associate the latent knowledge of the words in the pink color in the document.

Recently, deep learning-based algorithms [77-78] have received a high level of interest which has resulted in significant achievements in generating abstractive summaries. Sequence-based recurrent neural network (RNN) models with an attention mechanism [126] [112] [40] [38] [127] achieve effective results in summarization models. Compared to RNNs, the convolutional sequence network (CSN) [80]-based text summarization model captures the interrelation between words and text on a large scale. Although the recent progress in the neural-based text summarization model is promising, these models have a tendency to include unnecessary and unrelated content that originates from the source text and in doing so, the main theme of the source text is lost in its generated summary. This is because only word-level attention is utilized but high-level attention based on the topic of the document is not utilized in the summarization model. Incorporating latent topic information into summarization models which provide the relevant theme of the document could be effective in generating a meaningful summary.

The traditional latent Dirichlet allocation (LDA) [28] topic model has proven its accuracy in text summarization models to reveal the latent topics from the content [3][102]. However, obtaining novel topics which represent the document explicitly using only the statistical based-topic model is clearly not adequate. This may fail to produce coherent and semantically well-formed, meaningful summaries due to the lack of human-like context knowledge. Conceptual information is a kind of knowledge extracted from a concept-based knowledge base such as Probase [51] and ConceptNet [50] to capture the latent semantic information of the text. Developing a topic model using conceptual information or knowledge is a potential solution to identify and enrich the quality of the topic. The research direction of using a knowledge-powered topic in ATS has received scant attention.

In this chapter, we present a text summarization scheme using knowledge-powered topic information based on a CSN [80]. The scheme consists of two stages. In the first stage, we construct a topic knowledge base (TKB) which contains knowledge-powered topic information which we call topic knowledge (topKs). For this, we first present a conceptualization algorithm to obtain conceptual information. We present a knowledge-powered topic model (KPTopicM) to obtain the topKs using conceptual information. Then, we train datasets using the KPTopicM and use the learning data as prior TKB to provide topKs for the summarization model at a later stage. In the second stage, we propose knowledge-powered topic level attention for a text summarization model (KTOPAS) to incorporate the topKs into the CSN-based summarization model. Figure 6.1 shows an example of a summary generated by KTOPAS.

In KPTopicM, we adopt the LDA three-layer hierarchical topic model: word, topic and document layer respectively using a bottom-up approach. We add an extra latent knowledge or concept layer in the model between the topic and word layer to enrich the background knowledge of the topics. We retrieve concept information and derive the concept distribution over words using a conceptualization algorithm. We use these concept assumptions to integrate the concept layer in KPTopicM to enrich the latent knowledge of the text. We present a KPTopicM algorithm to incorporate conceptual information in the topic model to generate meaningful topics called topKs. We first present a summarization model DTopCSN based on two CSNs: word and topic level CSNs. This model retrieves

the topKs from TKB and incorporates the topKs into DTopCSN. The word and topic level CSN encoders associate each source element and topK of the document with each decoded element in the summary that predicts whether the source word or topK represent the summary element. Here, we measure attention from two aspects: the word and topic level to measure the importance of the source elements and topKs to the output summary. However, this model does not observe the semantic coherence of the topic in relation to the source text. Therefore, we improve the DTopCSN model and propose our summarization model KTOPAS by adding another CSN knowledge level to increase the coherence of the topic in relation to the document and measure the attention from three aspects (word, topic and knowledge level). The knowledge-level CSN encoder associates each word in the source text with decoded topKs to obtain the coherence of topKs in relation to the source text. We incorporate topKs to CSN to map the salient knowledge and its contextual information in our summarization model through an attention channel. We introduce the tri-attention channel which jointly learns the attention of the words over the summary output, topKs over the summary output and words over the topKs in the KTOPAS model. We join three attention weights into one and produce the final attention weight. Then we produce the final probability of the next target element in the output summary at the decoder of the word and topic level CSN. In addition, we use mixed training objective function [45] to maximize our proposed model.

The following has been achieved in the chapter.

• Our conceptualization algorithm retrieves semantically relevant and salient background knowledge of the document. KPTopicM generates coherent and meaningful topic information (topKs) efficiently using latent salient knowledge that represents the document well. We train the KPTopicM model over the Gigaword and CNN/Daily Mail datasets and use this as an independent prior TKB to provide topic information to the summarization model.

• Our proposed summary model KTOPAS incorporates topic information based on the background knowledge of the source text which is retrieved from TKB to provide salient topic knowledge while generating summaries.

• The tri-attention channel computes word, knowledge, and topic level attention jointly to provide contextual information from three observations. Then, a SoftMax activation function combines them to obtain the final probability distribution so that the model can generate semantically well-formed and coherent summaries.

• We conduct the experiment using Gigawords and CNN/Daily Mail datasets to evaluate KPTopicM and KTOPAS. The KPTopicM model provides more semantically relevant topic information compared to the statistic topic model for KTOPAS which improves the accuracy of our summary model KTOPAS. The experiment results show that KTOPAS achieves more competitive results than baselines as it produces meaningful and coherent summaries with a large vocabulary range.

The rest of the chapter is structured as follows. We discuss the recent related works in Section 2. The construction of the TKB is detailed in section 3. Section 4 presents the convolutional text summarization model with the knowledge powered topic level attention model (KTOPAS). TKB which is outlined in Section 3 is used to provide topKs to the summarization model which is detailed in Section 4. Our experiment evaluation is presented in Section 5. Finally, this chapter concludes with suggestions for further research in Section 6.

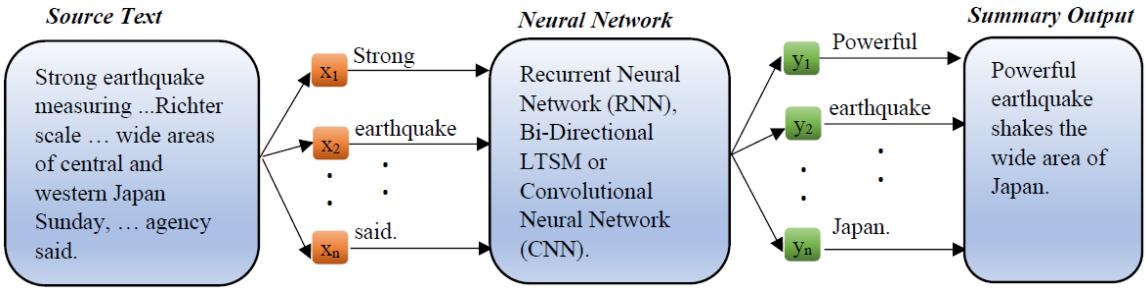


Figure 6.2: Text summarization with a neural network.

The orange box, $x_1, x_2, \ldots, x_n$, indicates the input elements and green box indicates the output elements, $y_1, y_2, \ldots, y_n$.

6.2. Related Work

In general, extractive and abstractive methods are the two methods used for automatic text summarization. Abstractive text summarization (ATS) generates summaries from the corpus with no constraints to use the available words from the original text using a deep

learning model or a neural network (NN) [126] whereas extraction [128] [38] generates summaries by selecting a part of the sentences in the source text. Compared to the extractive methods, the ATS model generates semantically well-formed and human readable summaries [129] [37]. Figure 6.2 illustrates an overview of summarization using NN. The NN model trains a large summarization dataset to predict the sequence of the output elements of the summary based on the sequence of the input element of the source text. During training, the sequence of the input elements is fed into the NN encoder to provide the sequence of the output as a summary to the decoder which is given in the dataset by adjusting the weighted parameter of each state of the encoder corresponding to the decoder in NN. These weighted parameter value predict the output element at decoder for each state of encoder. Once the network is trained based on a large dataset, ATS is able to predict the sequence of the output elements for the sequence of the input elements.

More recently, the RNN-based sequence to sequence framework (RNN-Seq2Seq) [36] [77] has received increased research interest for application in developing ATS approaches because it is able to achieve good summary results. Rush [40] first introduced an attention mechanism to the RNN-Seq2Seq model for ATS. Then, Nallapati [36] learned the hierarchical representations of a document and identified important information from documents by applying attention RNN-Seq2Seq. A pointer-generator model (PTGN) [38] was proposed to determine whether to copy a word or phrase from the source text over a pointer or generate a word from the vocabulary of a dataset while producing summaries. Paulus [37] proposed a deep learning summary model (ML+RL) using bi-directional LTSM [35] for the encoder and decoder, and also a reinforcement learning approach. Keneshloo [45] enhanced reinforcement learning using a multi-reward approach. Recently, Chen and Bansal [130], Gehrmann [44] and Xu [131] proposed a hybrid extractive and abstractive model. Lu learned multitask network for the ATS model MAT [132] based on a bi- directional encoder and decoder shared network.

The summarization model based on the convolutional sequence network (CSN) allows each state to be performed individually and therefore in parallel. The wide-range dependencies between the words in the document are captured and compared to the chain structure modeled by RNN. The ATS approach has been investigated based on CSN [114]

[133]. Recently, work was conducted on pretrained objective based ATS such as text summarization with a pre-trained encoder [43], BART [118], ProphetNet [119], discourse aware summarization model [134], PEGASUS [120], and multi-document based ATS such as hybrid multi-feature fusion [135], feature assessment [136] and post-Pareto analysis [137]. However, summarization models with topic level attention based on background knowledge have received scant investigation. Topic models are used to identify topics that best explain a set of documents. These topics are called latent because they only appear during the process of topic modeling. A traditional LDA topic model [28] has a hierarchical structure with three layers. These layers are used to present the probability distribution of documents over topics, and the probability distribution of topics over words. This LDA topic model achieves effective results in the text summarization model to discover latent topics from documents [3] [102]. However, this model only uses a statistical approach to identify topics, but it does not capture the background knowledge of the document for the topics which might result in the failure to generate meaningful and coherent summaries. Incorporating topKs through higher-level attention in the model could produce effective results such as context-relevant knowledge which is introduced into CNNs for text classification [121]. Therefore, we incorporate topKs in CSN through high level attention.

Humans are able to interpret documents and derive the main idea of the document due to certain background knowledge in the human brain, based on prior learning or past experiences. Referring to the example of the earthquake given in Figure 6.1, humans understand that an earthquake is an event that shakes the ground; it is not a person, place or thing. However, machines are unable to understand this by simply reading the document. A knowledge base (KB) is a kind of repository which provides information about a term [46]. A commonsense KB is a kind of repository which employs taxonomies and the relationships between concepts or knowledge to present information about a term. ConceptNet [50] is a common-sense KB which provides information about a term to help machines understand the meaning of the term similar to a human's understanding. Machines can retrieve and read conceptual information from the ConceptNet KB and relate the document to the main topic using knowledge from conceptual information. For example, a machine retrieves information about the topic term 'earthquake' from

ConceptNet and understands that this is a phenomenon that shakes the ground. Therefore, we incorporate knowledge-powered topic information (topK) in KTOPAS.

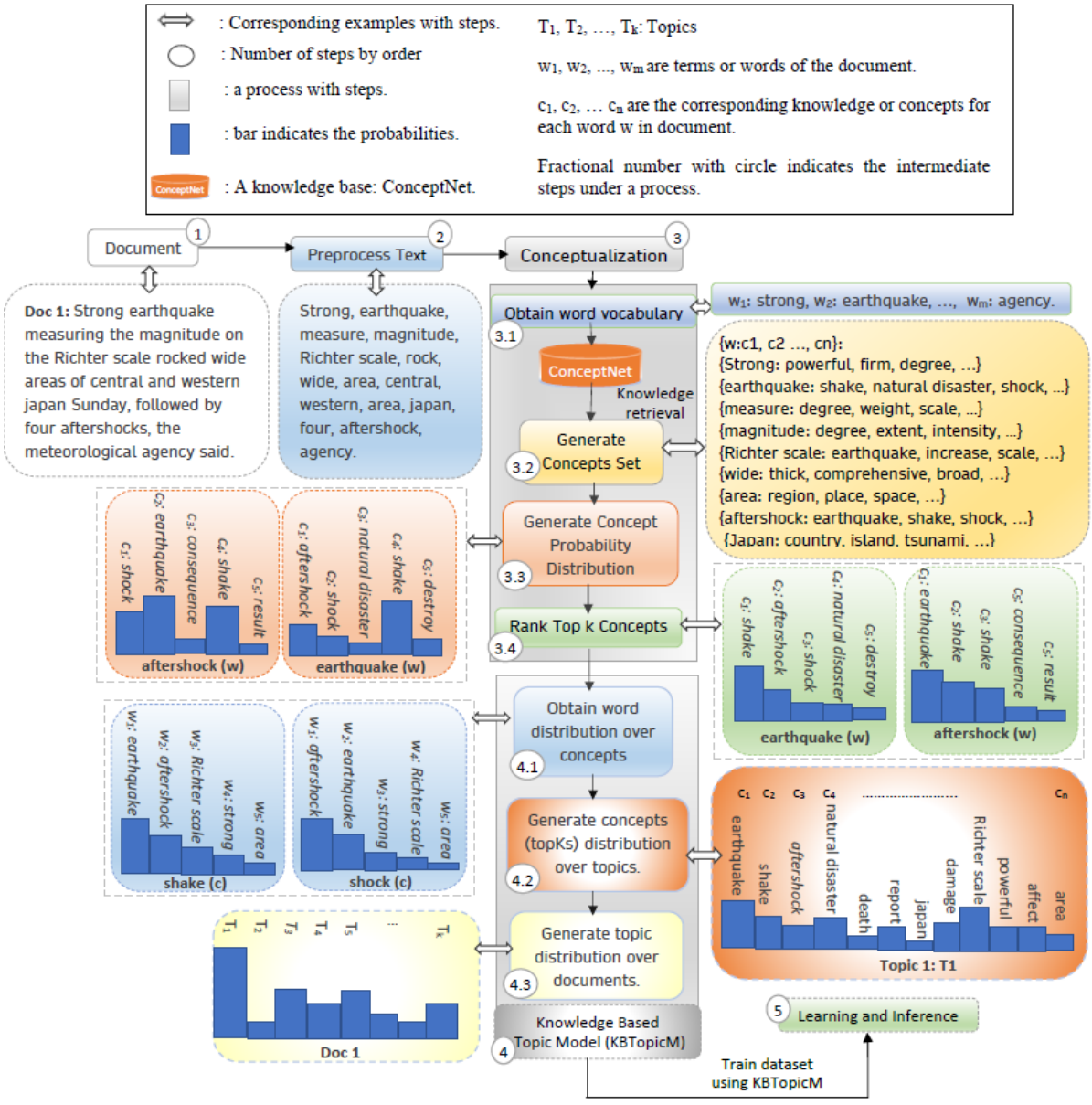


Figure 6.3: Overall scheme of topic knowledge base construction.

The corresponding example of each step is shown with the arrow. Conceptualization retrieves the top *K* concepts with the highest probabilities. The knowledge-powered topic model (KPTopicM) generates topic knowledge (topKs), concept distribution over topKs and topKs distribution over documents. The topic knowledge base is constructed by training the datasets using KPTopicM which contains the topKs with their probability information.

6.3. Topic Knowledge Base Construction

In this section, we propose a scheme to construct a TKB which contains topic information based on the background knowledge of documents. We refer to topic information as topic knowledge (topKs) in this chapter. This TKB is used as a prior knowledge base to provide the topKs in our summarization model. This scheme consists of a preprocessed document, conceptualization to obtain the conceptual information of the document from the KB as background knowledge, the KPTopicM to acquire the topKs using conceptual information, learning and inferring to train the data using the KPTopicM and we use this as prior knowledge for the TKB. For example, TKB produces the topK "shake" associated words in the document such as 'earthquake', 'aftershock', 'Richter scale' and 'Japan' which represent the document well. Figure 6.3 shows the scheme of the topic knowledge base construction.

6.3.1. Preprocessing

We preprocess the document before retrieving knowledge to remove the unnecessary and common text, and thus produce a normalized document. It has the following steps:

- *URL and Email Removal:* We remove URLs and emails from the input text.
- *Lower Case:* We convert the content of the input to lower case.
- *Stop-word Removal:* Stop-words are removed from the dataset.
- *Tokenization*: We tokenize each sentence in the input document. The sentences are transformed into list of words during tokenization.
- *Lemmatization*: Words are reduced to their stems.

We obtain the word vocabulary after preprocessing. To illustrate, the word list that was obtained from example 1 after preprocessing is as follows: 'strong', 'earthquake', 'measure', 'magnitude', 'Richter scale', 'rock', 'wide', 'area', 'Japan', 'four', 'after', 'shock', 'meteorological', 'agency'. We pre-process the text and obtain the vocabulary denoted as W=w_1 (strong), w_2 (earthquake), w_3 (measure), . . ., so on.

6.3.2. Retrieve Informative Knowledge

In this section, we retrieve the informative knowledge about a term from knowledge bases such as our OMRKBS or ConceptNet to understand the document. We take the example in Figure 6.1 “A fairly large earthquake measuring a magnitude …. agency said.”. First, we retrieve the information for each word in the document from knowledge bases such as our OMRKBS or ConceptNet. We transform the definition of the term into individual informative and meaningful features as knowledge using the same approach we used in chapter 4. First, we extract the information about a term from DBpedia, split the text using the NLP technique, and apply a rule to transform information into informative knowledge. For example, the information about ‘earthquake’ is extracted from DBpedia as follows:

“An earthquake is the shaking of the surface of the Earth resulting from a sudden release of energy in the Earth”

Similarly, the information about ‘escalate’ is extracted from ConceptNet as follows.

“earthquake”, “shake”, “shock”, “results”, “consequence”, and so on.

After splitting the sentences and applying rules, the features of the military campaign are represented in OMRKBS using the mapping expression algorithm mentioned in 4.4.7. This helps us to understand the document since this provides unique features and structural and concept information. We retrieve information about the terms ‘earthquake’ and ‘aftershock’ using the aforementioned steps as follows:

Earthquake
<shake, surface, earth>
<results_from, energy, earth >
Aftershock
<earthquake>
<shake>
<shock>
<consequence>
<results>

We take the example from Figure 6.1 to explain that the machine can learn or know about words $w \in W$ in the document such as ‘strong’, ‘earthquake’, and so on using ConceptNet. The generated concept set $c \in C$ for the corresponding term $w \in W$ is given as follows. We

generate a list of concepts c = c_1, c_2, c_3, …, c_n for each word w_i in the document using the approach in 5.3.2 from knowledge bases such as Dbpedia and ConceptNet. Figure 6.4 shows how 'earthquake' concept is defined in OMRKBS. The generated concepts for each word using the OMRKBS approach are as follows: 1. Strong (w_1): powerful (c_1), tough (c_2), degree (c_3), ... 2. earthquake (w_2): shake (c_1), natural disaster (c_2), shock (c_3), ... 3. rock (w_3): stone (c_1), natural disaster (c_2), hard (c_3), 4. Richter scale (w_4): earthquake (c_1), increase (c_2), scale (c_3), ... 5. wide (w_5): thick (c_1), comprehensive (c_2), broad (c_3), ... 6. aftershock (w_6): earthquake (c_1), shake(c_2), shock (c_3), … 7. Japan (w_7): country (c_1), island (c_2), tsunami (c_3), ... and so on.

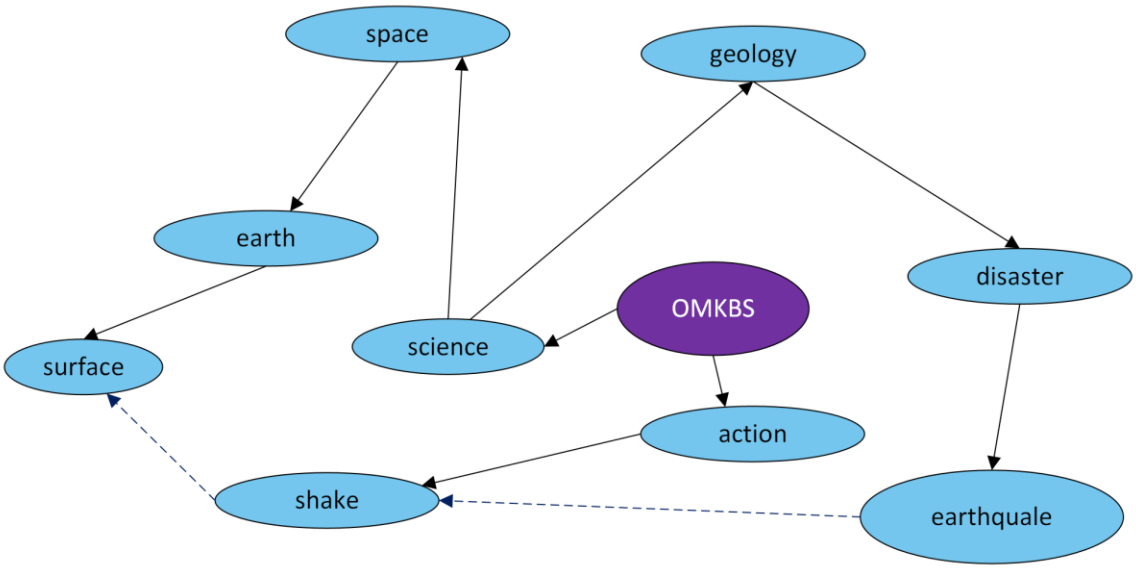


Figure 6.4: An example of a concept ('*earthquake*') defined by the proposed OMRKBS.

'earthquake' is defined as'<earthquake, shake, surface>. The words in the blue circle are concepts and the root is in the purple circle. The dashed arrows indicate the relationship between concepts according to the definition of a concept ('earthquake') while the solid arrows indicate subclasses.

6.3.3. Conceptualization

The appropriate background knowledge of the words in texts can be very informative and can reveal the latent relationships between them. This is important to retrieve relevant knowledge or concepts which are strongly associated with the text without including unnecessary information in the text. Conceptualization is the process of retrieving conceptual information for the document from the knowledge base. We propose a

conceptualization algorithm to obtain conceptual information using the ConceptNet KB which is relevant and well associated with the source document. First, we derive the concept distribution for each $w \in W$ in the document which we retrieve from the knowledge base as the informative knowledge. We compute two types of statistical conceptual information: knowledge or concept distributions which is the probability of a concept set belonging to a word or term in the text and word distributions which is the probability of the word set in the text belonging to a concept or knowledge corresponding to the text. Next, we measure the weight to present the association of each concept to the words in the text. Finally, we rank the top N concepts with highest weight which is strongly associated with the text. In this chapter, knowledge and concept as considered to be similar terms.

Algorithm 6.1: Conceptualization

1. **Input**: Input elements: ($W=w_1, w_2, \ldots, w_n$)
2. **Output**: Top K concepts of input elements, W
3. Begin
4. sample each word w from W
5. for each $w \in W$ do
6. Retrieve related the concepts $C= \{c_1, c_2, \ldots, c_m\}$ from ConceptNet
7. for each $c \in C$, do
8. Compute the probability of w under the concept c, P(c|w)
9. Identify k concepts with highest probabilities from C
10 Compute weight $k_c \in K_w$ of the concept c_i to measure the degree of the association between the concept c_i towards w.
11. end for each
12 end for each
13. Rank the highest K weighted concepts ($c_1, c_2, \ldots, c_K$) associated with W

This algorithm consists of three steps: generate concept distribution over words and word distribution over concepts, measure the weight to associate a concept and word, and rank top k concepts. In the first step, we generate the concept probability distribution for the generated concept sets of the input elements which we retrieve from the knowledge base as informative knowledge. We apply an inverted indexing technique [138] to map the terms $w \in W$ into the weighted set of related concepts. We compute the probability of concept set

$c \in C$ which belongs to the source terms $w \in W$, defined as $P(c|w)$, using the following equation:

$$P(c|w) = \frac{count(w,c)}{\sum_{c \epsilon C} count(w,c)} \quad 6.1$$

where the number of co-occurrences of term w and concept c is denoted as count (w, c). Then, we map the $w \in W$ to the weight vector of the concept set, that is, $K_W = (k_{c1}, k_{c2}, \ldots.)$ which represents the source text T in the conceptualized space. $kc \in K_W$ represents the weight of concept c in text T, indicating the strong association between the concept c and text. We calculate this using the following formula:

$$k_c = log \sum_{w \in W, V_i \in P_c} V_c \times idf_c(w) \times icf(c) \quad 6.2$$

where *Vc* denotes the probability of a term $w \in W$ to be mapped by concept *c*. *idfc* denotes the inverse document frequency which represents the identity of the term in the concept and *icf* denotes the inverse concept frequency which represents the importance of the knowledge in the whole concept set, and we calculate these using the following formula:

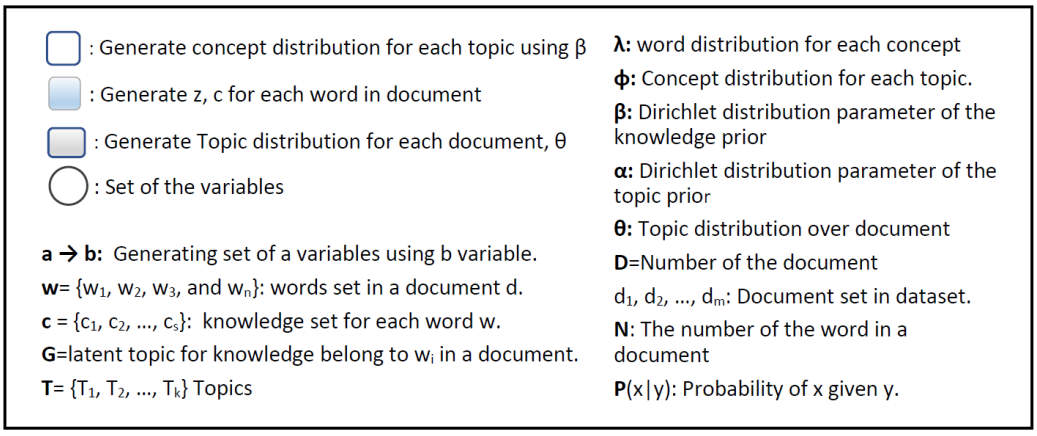
$$idf_c(\mathrm{w}) = log \frac{C_n}{N(w)+1} \quad 6.3$$

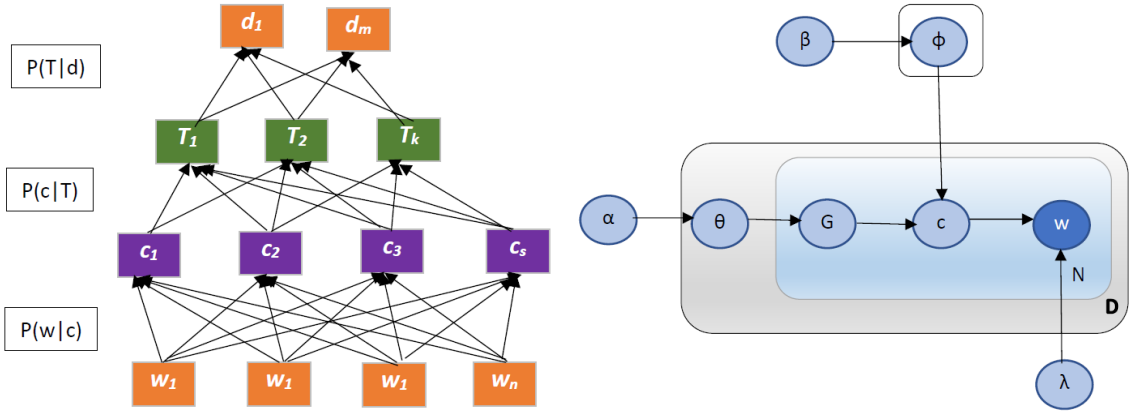
$$icf(c) = log \frac{C_n}{N(\mathrm{c})+1} \quad 6.4$$

where C_n indicates the overall number of concepts in the concept set, $N(w)$ indicates the number of times the term w co-occurs with concept c, and $N(c)$ indicates the number of times concept c_i appears in the entire word mappings. We mapped each sentence in the text or document to the concept set. In the last step, we rank the top *K* concepts' weight in terms of each document. We can see that the words 'earthquake', 'Richter scale', 'aftershock' and 'Japan' are associated with some common concepts such as 'shake', 'natural disaster', 'degree', 'tsunami', 'shock' and so on. These types of latent concepts are the top K concepts for a document.

6.3.4. Knowledge-based Topic Model

Informative and semantic latent knowledge helps to identify and describe the relevant, meaningful and coherent topics in a more extensive way. We propose a knowledge-powered topic model (KPTopicM) which employs informative knowledge to generate meaningful and coherent topics. This is a four-layer topic model that introduces a hidden knowledge layer within the topic and word of the three layers in the LDA topic model to





(a) Four-layer hierarchical knowledge-powered topic modeling.

(b) Graphical model for KPTopicM

Figure 6.5: knowledge-powered topic model mechanism

integrate conceptual information in the statistical topic model. First, we retrieve the top N concepts with their statistical conceptual information which are strongly associated with text using the conceptualization algorithm. The classic LDA deduces the topic distribution per word in the document. Then, apart from LDA, we deduce the topic distribution per concept followed by the concept distributions per word and observe the concepts through

the word distribution to integrate conceptual information and capture the word dependencies of the concept in the topic model. We use the Gibbs sampling technique to predict the concept and topic distribution. Figure 6.5(a) shows the four-layer hierarchical knowledge-powered topic model where we can see the indirect word dependencies through concepts and the direct concept dependencies in the topic distributions. The graphical model and the definition of notations in the KPTopicM model are depicted in Figure 6.5(b).

Algorithm 6.2: Generative process for KPTopicM.

Initial

T: Total number of topics

D: Total number of documents

N: Number of words in d document

1. For each $t \in T$:
2. Produce a word distribution $\varphi_m \sim Dir(\beta)$
3. End for each
4. For each $d \in D$:
5. Produce a topic distribution $\theta_d \sim Dir(\alpha)$
6. For $w \in W_N$ do
7. Produce a topic $G_n \sim Mult(\theta_d)$
8. Produce a concept $cn \sim Mult(\varphi G_n)$
9. Select a word w_n for concept c_n from λ, a probability distribution using ConceptNet.
10. End for each
11. End for each

Let each document $d \in D$ represent a group of words $w \in W$ with a total of N words, the Dirichlet distribution is denoted as *Dir* and the multinomial distribution is denoted as *MultD*. The *Dir* parameter of the topic prior is denoted as α, and the *MultD* of the *d* document over topics is denoted as θ_d (topic distribution for a document). The parameter of the *Dir* of the knowledge prior is denoted as β and the *MultD* of the m-th topic is denoted as φ_m. We denote the knowledge distribution over words as λ which is obtained from conceptualization. w_n is the *n*th word in doc *d*. We use c_n to denote a concept of w_n and G_n to denote the latent topic for c_n. The algorithm shows the generative process of our KPTopicM. First, the concept distribution φ is sampled per topic from the Dir parameter of topic prior α (line 1 and 2). Then, the topic distribution, θ, is sampled per document

using the *Dir* parameter of the knowledge prior, β (lines 3 and 4). Lastly, by selecting a latent topic G_n from the topic distribution, *MultD* of θ_d, concept c_n is generated from the topic distribution of the G topic, *MultD* of φz (lines 5 and 6). In lines 7 and 8, a word is selected from the corresponding word distribution λ for concept c. This model represents each document with various related topics and topKs are the concepts that represent topics. We obtain the top *M* concepts with the highest probabilities for each topic in the vocabulary of topic *K*. For example, KPTopicM produces topKs such as 'shake' which was the background knowledge of the related words in the document such as 'earthquake', 'aftershock', 'Richter scale' and 'Japan'. This topK 'shake' belongs to a topic that the document well that does not come from the source document.

6.3.5. Learning and Inference

The goal of the inference process is to predict parameters $\hat{\varphi}_m$ and $\hat{\theta}_m^d$ which can represent topics and documents well respectively. We predict these parameters using the Gibbs sampling [140] technique. We associate the concepts with words, topics with concepts, topics with documents and find their strongest association while generating topic distribution over documents θ_d and concept distribution over topics, φ_k. First, we mapped each word w_n in n position in document d to a concept set using the concept distribution over words which is obtained from the conceptualization algorithm. The probability of a word w belonging to concept c which is retrieved from ConceptNet is defined as the conditional probability $P(w_n|c_n)$ as follows:

$$P(w|c) = \frac{\mathrm{P(w, c)}}{p(c)} \qquad 6.5$$

where $P(w|c)$ is proportional to the co-occurrence of the word and concepts, and $P(c)$ is approximately proportional to the observed frequency of c. Next, we assign M concepts to each concepts randomly from concept vocabulary and generate the concepts distribution per topics. Then, when a word w_n is mapped to a concept set, we use the conditional probability of a concept c_n belong to*wn* in document d that represent a topic G_n to integrate the concepts information in topic model and obtain the relevant concepts as topics. We compute this probability which is proportional to the weights from three aspect: topic distribution over document, concept distribution over topics and word distribution over

concepts to obtain the association words, concepts and documents to the topics for the given parameters α and β, and the observed words w using the following equation.

$$P\left(G_n = m, c_n = k \middle| w, c_{-(n)}, G_{-(n)}; \alpha, \beta\right) = \frac{\beta_{m,\, c_n} + s^{c_n}_{-(n),m}}{\sum_{x=1}^{E} \beta_{m,x} + s^{(.)}_{-(n),m}} \cdot \frac{\alpha_m + s^{(d)}_{-n,m}}{\sum_{t=1}^{K} \alpha_t + s^{(d)}_{-(n),.}} P(w_n | c_n) \quad 6.6$$

where C is the knowledge vector corresponding to the word. G_{-n} represents the whole topic distribution except topic of w_n. The number of concepts in c is denoted as E. The number of concepts corresponding to topic m in document d except G_n is denoted $s^{(.)}_{-(n),m}$ and the total sum over that dimension is denoted as $s^{d}_{-(n),.}$. The number of terms assigned to topic m is denoted $s^{(d)}_{-(n),m}$ in the document d and $s^{c_n}_{-(n),m}$ is the count of c_n in topic m except G_n. We can see that the equation has weights in relation to three aspects. In the first part, the weight indicates the association that expresses how much each topic is represented by a concept from the vocabulary and in the second part, the weight indicates the association that expresses how much a document is represented by a topic. The last part indicates how much a concept is represented by a word to the chosen topic G_n. We reassign concept c_n to topic G_n which has highest probability. This process is iterated N times and then reaches a convergence state. After Gibbs sampling, we predict the probability of a topic m knowledge given a document d,. $\hat{\theta}^{d}_{m}$ and the probability of a concept given a topic, $\hat{\varphi}_m$ using the sample topic and knowledge. Lastly, we estimated the parameters using the following equation:

$$\hat{\varphi}_{c,m} = \frac{n^{(c)}_{k} + \beta_{m,c}}{\sum_{t=1}^{KE} \beta_{m,t} + \rho^{(c)}_{.,m}} \quad 6.7$$

$$\hat{\theta}^{d}_{m} = \frac{n^{(d)}_{m} + \alpha_m}{\sum_{t=1}^{E} \alpha_t + \rho^{(d)}_{.,m}} \quad 6.8$$

6.3.6. Dataflow of the Topic Knowledge Generation

In this section, we describe the dataflow to generate the topic knowledge using KPTopicM. Figure 6.6 shows the flowchart to describe the process of constructing the topic knowledge base (TKB). This method comprises four steps: preprocessing, conceptualization, KPTopicM and learning. Preprocessing transforms the text into a word vocabulary by

removing unnecessary words or symbols from the text. The conceptualization process retrieves the most relevant background knowledge of the document from the knowledge base. First, this algorithm retrieves the related concept set for each word in the document from a knowledge base such as ConceptNet and computes the concept distribution per word measures the association of each concept to the words in the document. After measuring the weight for the association of concepts of each word in the document, we rank the top N related concepts with the highest weight of the concept associated with the words in the document.

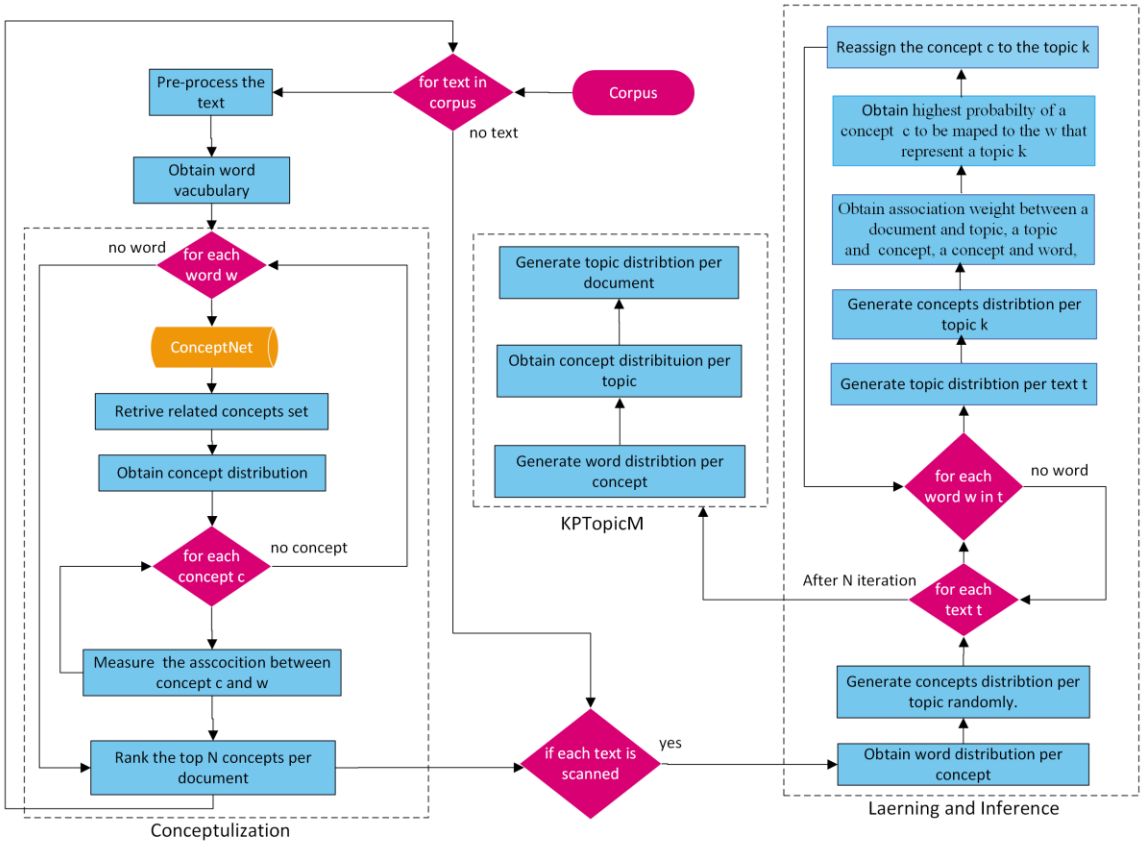


Figure 6.6: A flowchart to show the process of constructing the topic knowledge base (TKB).

KPTopicM provides the concept distribution over topics where a concept has been observed through words and the topic distribution over documents where a topic has been observed through concepts. The Gibbs sampling technique is used to train the KPTopicM and find the best match for the concepts in a topic and the topics in a document with their association from three aspect: association of a concept and topic, a topic and document and

a word which belongs to a concept. We first randomly assign M concepts to each topic. Then, we choose a concept randomly and reassign the other topic which has best association from three aspect: association of a concept and topic, a topic and document and a word which belongs to a concept jointly. After N iterations, topic distribution per document and concept distribution per topic has been learned.

6.4. Convolutional Summarization Model with Knowledge based Topic Level Attention

It is important to capture coherent and informative topic information to focus on relevant and main theme of the document while generating summaries. In this section, we propose the CSN-based summarization model with knowledge-powered topic level attention (KTOPAS) which incorporate topic information based on informative background knowledge to generate coherent, relevant and meaningful summaries with word diversity. We use the background knowledge of the document to bridge the gap between informative knowledge and topic information in capturing coherent, semantic and relevant topics in the generated summaries. This model comprises convolutional sequence architecture, topic knowledge generation and word and position embedding, a tri-attention attention mechanism, final probability generation to predict the target element in the output summary from topic or source elements, and a learning process to train the parameters and maximize the model performance. The graphical illustration of our summarization model KTOPAS is shown in Figure 6.7.

6.4.1. Convolutional Sequence Architecture

To utilize the advantage of the convolutional sequence architecture of CSN [9] which can capture long range dependencies of words in the large text and compute the operation fast, we use CSN in our model. We use convolutional sequence architecture for the text summarization model based on CSN. We add three CSNs: word, knowledge and topic level in the architecture which are paired with the input word and topK embeddings, input word and summary output, and topK and summary output respectively. We describe the word with the position embedding and hierarchical structure for this architecture.

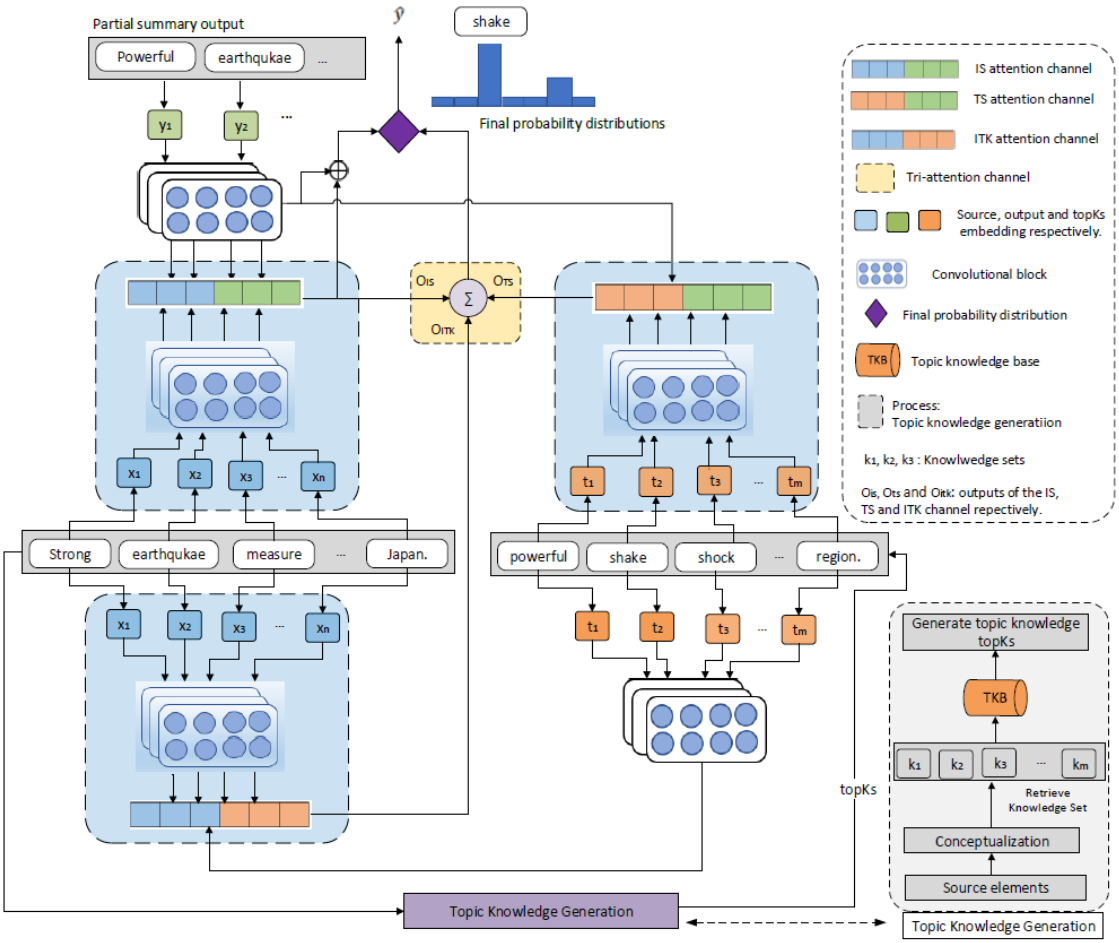


Figure 6.7: Convolutional summarization mode with knowledge-powered topic level attention (KTOPAS).

The Partial Summary is the produced summary elements at a decoder state for a sequence of source elements of documents at the encoder state. For example, given a sequence of input "Strong earthquake measuring on ...aftershock", the partial summary is "Powerful earthquake … "and the next target element for the output summary is "shake". topKs which are the topic know ledge are retrieved using topic knowledge generation to feed itself to the topic level CSN in KTOPAS.

Word and position embedding

We embed the input and output elements with their relative position at the encoder and decoder of the word and knowledge level CSN, respectively. We add each element and its position in the source and encode these for the word and knowledge level CSN. First, we embed the n input elements W= (w_1, w_2, w_3, ..., w_n) of document *d* to vector representation $e = (e_1, e_2, \ldots, e_n)$. $E \in R^{V\,xf}$ is an embedding matrix where the rows are assigned with $e_j \in Rf$ and the vocabulary size is defined as *V*. We then embed the position embedding which

is the absolute position of the input element in the source text, defined as $p = (p_1, \ldots, p_n)$ to keep the order of the sequence of the input. For example, the position embedding for word w_i at position i in the input sequence is p_i. Finally, the input elements are represented as $x = (x_1, x_2, \ldots, x_n)$ by joining the word and position embedding, $x = (w_1, p_1), \ldots, (w_n, p_n)$. Similarly, we represent the output elements with m word at the decoder as $g = \{(\hat{y}_1, \hat{p}_1), \ldots, (\hat{y}_2, \hat{p}_2)\}$ where y is the output and p is the position embedding of the output. We apply a similar embedding at the decoder and encoder for the knowledge level as well.

Hierarchical structure

Multi-layer hierarchical structures are applied to three CSNs: word, knowledge, and topic level. The kernel width is denoted as k and the dimension of the word embedding as d. Let $a^{\ell} = (a^{\ell}_1, \ldots, a^{\ell}_n)$ denote the output of the l-th layer at the decoder, and $h^{\ell} = (h^{\ell}_1, \ldots, h^{\ell}_m)$ at the encoder. a^{ℓ}_i is the layer with kernel width k resulting state at an encoder network which contains information over k input elements. $X \in R^{kxd}$ is fed into each convolution block. Convolution constructs an integration of k input elements in d dimension as $X \in R^{k\,xd}$ by stacking blocks and maps them to a single output element $Y \in R^{2d}$. Gated Linear Units [75] are applied on the output of convolution $Y = [IJ] \in R^{2d}$.

$$g([I; J]) = I \otimes \partial(J) \tag{6.9}$$

where the inputs to the non-linearity are defined as $(I; J) \in R^d$, the sigmoid function is denoted as σ, the point-based multiplication is denoted as $\otimes$, and $g([I; J]) \in R^d$ is denoted as the output. The convolution unit i on the l-th layer is computed by the residual connection as

$$a_i^l = g\left(W^l\left[a_{\frac{i-k}{2}}^{l-1}, \ldots, a_{\frac{i+k}{2}}^{l-1}\right] + b_w^l\right) + a_i^{l-1} \tag{6.10}$$

where W and b are the parameters of each convolutional kernel, and g is the function composition operator. Finally, we compute a distribution over the K possible next target elements y_{i+1} by passing the top decoder output a_i^l via a linear layer with weight parameter W_Y and bias b_y to a SoftMax classifier:

$$p(y_{i+1}|y_1, \ldots, y_i x) = softmax(W_Y h_i^L + b_Y) \in R^T \tag{6.11}$$

6.4.2. Topic Knowledge Generation and Embedding

We introduce process topic knowledge generation which retrieves coherent and relevant topic information based on informative and semantic latent knowledge (also called topKs) from the prior topic knowledge base (TKB) for the given input elements and embed the topKs at the encoder of the topic level CSN in KTOPAS. First, the top N concepts are chosen for each document using the conceptualization method detailed in section 4.1. Let a concept set $C = (c_1, \ldots, c_n)$ be found for the given input words. We obtain the probability of each concept or knowledge $c \in C$ as the topics from TKB and chose M top topic knowledge from these concepts. We refer to this topic knowledge as topKs. We denote $Q_{topic} \in R^{Kxd}$ as the topKs embedding matrix and topic vocabulary of the concepts as $K \in V$ where V is the vocabulary of the document. We produce a vector representation t for the given topKs. It is assumed that $t \in V$. We identify a topK $t_c \in R^d$ for a concept c if $c \in K$ and embed it as a row in the Q_{topic}. Similarly, we present the topKs embedding matrix at the decoder for the knowledge level and at the encoder for the topic level CSN as p_c and r_c respectively.

6.4.3. Tri-attention Mechanism

It is obvious that the strong association between the topic, summary and the source document can make the summary more relevant to the topic and source document. We introduce the tri-attention channel to generate a more relevant summary which has a strong association with the topic and source document. The tri-attention mechanism incorporates this topic information (topKs) which is generated from the topic knowledge generation in our model. The tri-attention mechanism comprises Input-summary (IS), Input-Topic knowledge (ITK), Topic Knowledge-summary (TS) attention channel and a tri attention channel. IS, TS and ITK are used to get attention into our model from three aspects: the word, knowledge, and topic level CSN respectively. IS and TS measure the attention weights of the topics and source elements respectively which are relevant to the summary elements and ITK measures the attention weights of the relevant topic elements to the source elements while generating summaries. We introduce the tri-attention channel which combines the three attentions into one to facilitate the model to generate more relevant and

coherent summaries. We describe each attention channel in the following sub-section. Figure 6.8 shows the mechanism of the tri-attention channels.

IS Attention Channel

We add an individual channel to pay attention to the source words for the summary outputs at the word level CSN to capture the important relevant source words while generating each target element of the summaries. We use the individual attention mechanism for each layer to perform multiple attention ("hop") per time step and access the previously attended words [76]. The current decoder state a^l_i is embedded as v^l_i by joining a^l_i with the previous target element embedding q_i to measure attention:

$$v_i^l = W_h^l h_i^l + b_d^l + q_i \tag{6.12}$$

Let $W^l{}_a \in R^{dxd}$ be a weight matrix and bias $b^l a \in R^d$ is the learning parameter. We measure the attention weight Ois^l_{ij} of the i state and j input element of the source text through dot product between v^l_i and the output u^e_j of the last encoder block e_o as follows:

$$\theta_{ij}^l = \frac{\exp\left(v_i^l .\ u_j^{e_o}\right)}{\sum_{t=1}^{n} \exp\left(v_i^l .\ u_t^{e_o}\right)} \tag{6.13}$$

We compute the conditional input $c^l_i \in R^d$ of the current decoder layer as follows:

$$Ois_i^l = \sum_{j=1}^{n} \alpha_{ij}^l (u_j^{e_0} + a_j) \tag{6.14}$$

where we denote x_j as the input element embedding. After computing c^l_i, this is joined to the output of the corresponding decoder layer al and serves as a part of the input to $a^{l+1}{}_i$.

$$\mathrm{c}_i^l = \sum_{j=1}^{n} \mathrm{Ois}_{ij}^l \left(u_j^{e_0} + x_j\right) \tag{6.15}$$

TS Attention Channel

In contrast to the base CSN, we include a high-level topic attention at the topic level CSN to focus on the important and informative relevant topics while generating summaries. We call this the TS attention channel. First, we embed current decoder state s^l_i of the topic level for convolutional unit i on the l-th layer as v^l_i using the following equation.

$$\ddot{v}_i^l = W_s^l s_i^l + \tilde{b}_s^l + t_i \quad 6.16$$

where r_i is the previous target topic embedding. First, we compute attention for convolution unit i on the l-th layer at the decoder of the topic level over the summary output jointly using the following formula.

$$Ots_{ij}^l = \frac{\exp\left(\ddot{v}_i^l . u_j^{e_0} + \ddot{v}_i^l . u_j^{e_s}\right)}{\sum_{t=1}^{n} \exp\left(\ddot{v}_i^l . u_j^{e_0} + \ddot{v}_i^l . u_t^{e_s}\right)} \quad 6.17$$

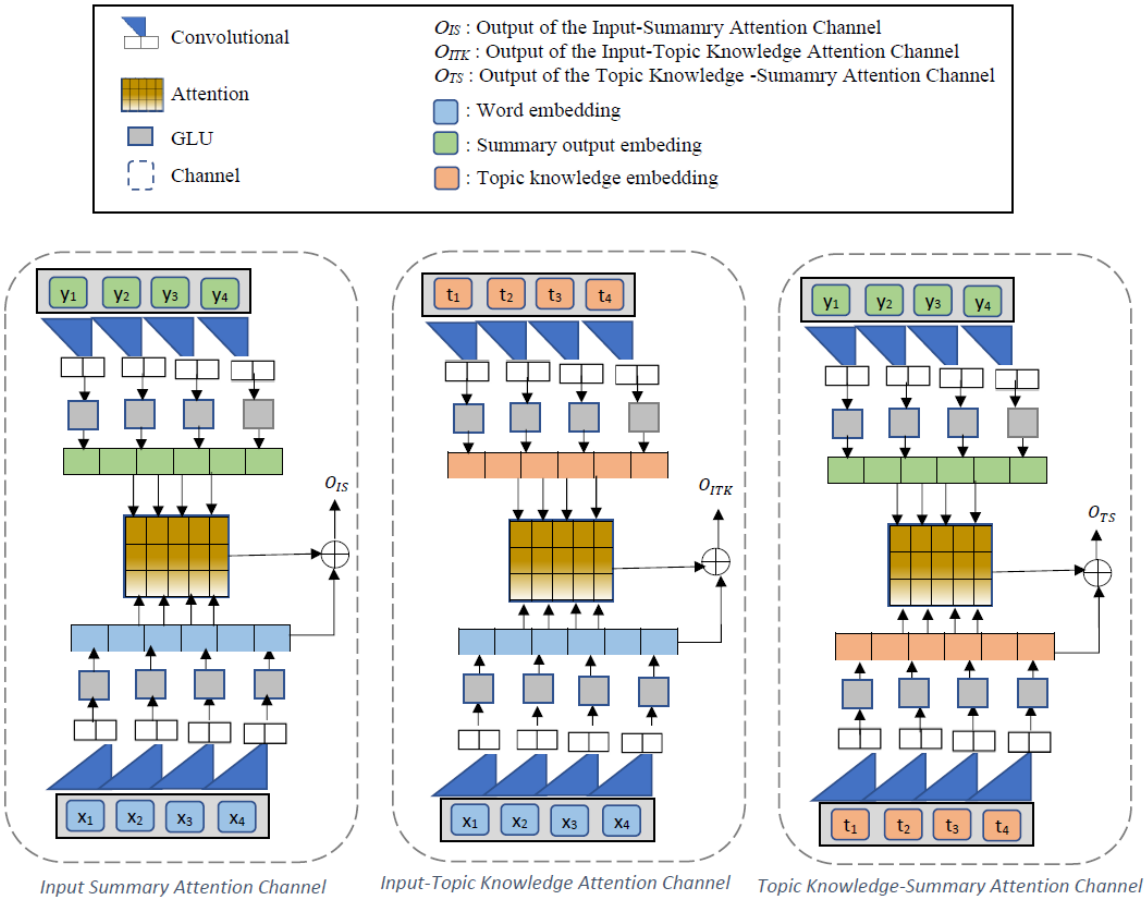


Figure 6.8: Mechanism of the three attention channels: Input-Summary, Input-Topic Knowledge, Topic Knowledge-Summary attention channel.

We call the topic knowledge-powered text summarization model with double attention channel (DTopCSN). DTopCSN measures attention from two aspects: the word and topic level CSN through the double attention channel. Then, the conditional input is computed by

$$\ddot{c}_i^l = \sum_{j=1}^{n} \text{Ots}_{ij}^l \left(u_j^{e_s} + t_j\right) \quad 6.18$$

where u^{es} is the output of the last topic level encoder block e_s and t_j is the topic embedding at the encoder of the topic level CSN. The two-conditional input c_i^l and $\ddot{c}_i^l$ are joined to the output of the corresponding decoder layer s_i^l and are a part of the input to s_i^{l+1} for DTopCSN model. However, this model does not consider semantically relevant topics or coherence for the input elements. Therefore, we improve this model by adding one more attention channel (described in the next section) to provide more semantically relevant topKs in terms of source documents as the target summary output elements. For this, instead of using equation 6.17, we compute high-level topK attention for convolution unit i on the *l*-th layer in the decoder of the topic level over the summary output individually using the following formula similar to multi-hop attention.

$$Ots_{ij}^l = \frac{\exp\left(\ddot{v}_i^l . u_j^{e_s}\right)}{\sum_{t=1}^{n} \exp\left(\ddot{v}_i^l . u_t^{e_s}\right)} \quad 6.19$$

Then, the conditional input $\ddot{c}_i^l$ is computed using the same equation 6.18 and will join with other conditional inputs in next section as the output of the corresponding decoder layer s_i^l. We use this attention to measure the importance of each topK in the summary output elements.

ITK Attention Channel

In the tri-attention mechanism, we add one more individual channel to pay attention to the topKs for the input sequences in the knowledge level CSN and drive the model to preserve the strong association of the topic information with the source text in the generated summaries. We use a hop method which is similar to one we used previously. Currently, the encoder state d_i^l is embedded as $\ddot{v}_{ij}^l$ to measure the attention using the following formula.

$$\ddot{v}_{ij}^l = W_d^l d_i^l + b_d^l + p_i \quad 6.20$$

where $p_i \in \mathrm{R}d$ is the previous decoded topic embedding at the knowledge level CSN and the weight parameter is denoted as $W^l_{d.}$ Here the weight of attention is $Oitk^l_{ij}$ from the i-th

concept regarding the input elements *j*. A large value of $Oitk^{l}_{ij}$ means that the *i*-th concept is more semantically similar to the source element j. We measure $Oitk^{l}_{ij}$ through the dot product between v^{l}_{ij} and the output u^{et}_{j} of the last encoder block *et* and normalize the attention weight of the topKs as follows.

$$Oitk^{l}_{ij} = \frac{\exp\left(\ddot{v}^{l}_{ij}.\, u^{e_t}_{j}\right)}{\sum_{t=1}^{n} \exp\left(\ddot{v}^{l}_{ij}.\ u^{e_t}_{t}\right)} \tag{6.21}$$

This attention weight is used to get the coherence among topKs in terms of input. We compute the conditional input. $\ddot{c}^{l}_{i} \in R^{d}$ of the current layer of the decoder as follows.

$$\ddot{c}^{l}_{i} = \sum_{j=1}^{m} \text{Oitk}\ ^{l}_{ij}\left(u^{e_t}_{j} + g_j\right) \tag{6.22}$$

where g_i is the encoded input embedding at the knowledge-level CSN. After computing. $\ddot{c}^{l}_{i}$ this is joined to the output of the corresponding decoder layer d^{l}_{i} of knowledge-level CSN and serves as a part of the input to $d^{l+1}{}_{i}$.

Tri-Attention Channel

Finally, we introduce a tri-attention channel which combines the above attention of the three channels to one to drive the model to produce more relevant and coherent summaries which can preserve the main the topics and meaning of the document. We joined the three *Ois*, *Oitk* and *Ots* attention weight to obtain one final attention weight of each concept. The final attention weight is computed by

$$\begin{aligned} \pi &= softmax\left(\alpha Ois^{l}_{i} + \beta Ots^{l}_{i} + \gamma Oitk^{l}_{i}\right) \\ &= \frac{\exp\left(\alpha Ois^{l}_{i} + \beta Ots^{l}_{i} + \gamma Oitk^{l}_{i}\right)}{\sum_{t=1}^{m} \alpha Ois^{l}_{t} + \beta Ots^{l}_{t} + \gamma Oitk^{l}_{t}} \end{aligned} \tag{6.23}$$

where α, β, γ ∈ [0; 1] are the learnable parameters of the network to adjust the importance of the three attention weights jointly. The embedding matrix Q_{topic} which is normalized from the final attention weights is employed to compute a weighted sum of the concept vectors *t* to represent the concepts through semantic vectors.

$$Q = \sum_{i=1}^{M} \pi\, t_i \tag{6.24}$$

We joined $c_i^l, \ddot{c}_i^l$ and $\dddot{c}_i^l$ to the output of the corresponding decoder layer $s_i^l$of topic level and are fed back as input to s_i^{l+1}. KTOPAS capture the topKs attention over the summary elements and the source elements while DTopCSN capture the topKs attention over summary elements only.

6.4.4. Final Probability Generation

The probability distribution over every possible output element for the next target at time step *t*, $\hat{y}_{i+1} \epsilon R^T$, is computed as follows.

$$\bar{p}(y_{i+1}) = p(\hat{y}_{i+1}|\hat{y}_i, \dots, y_1, x) \in R^T \quad 6.25$$

We transform the last decoder outputs $a_i^{L_0}$ of the word level CSN and decoder outputs s_i^{Lt} of the topic level CSN through a linear $\Delta(.)$ using the following equation.

$$\Delta(h) = Wh + b \quad 6.26$$

where *W* and *b* are the learning parameters. Then the final probability distribution is generated by the following equation.

$$\bar{p}(y_{i+1}) = \frac{1}{Z}[exp\left(\Delta\left(a_i^{L_0}\right)\right) + exp(\Delta\left(s_i^{Lt}\right)) \otimes G_{\{w \in K\}}] \quad 6.27$$

where the normalizer is denoted by *Z* and *G* is the indicator vector which expresses whether each candidate word *w* in y_{i+1} is a topK or not. If *w* is a topK, the generation distribution is biased through the topic information. Otherwise, the topic part is ignored. Figure 6.9 shows illustrates a flowchart to describe the sequence of process of our entire KTOPAS model.

6.4.5. Dataflow of the KTOPAS Model

In this section, we describe the dataflow of the model as shown in Figure 6.9. This method comprises several processes: multi-layer structure, topic knowledge generation, an attention mechanism, tri-attention channel and final probability distribution. The multi-layer structure is the backbone of the CSN which represents a sequence of the entire text

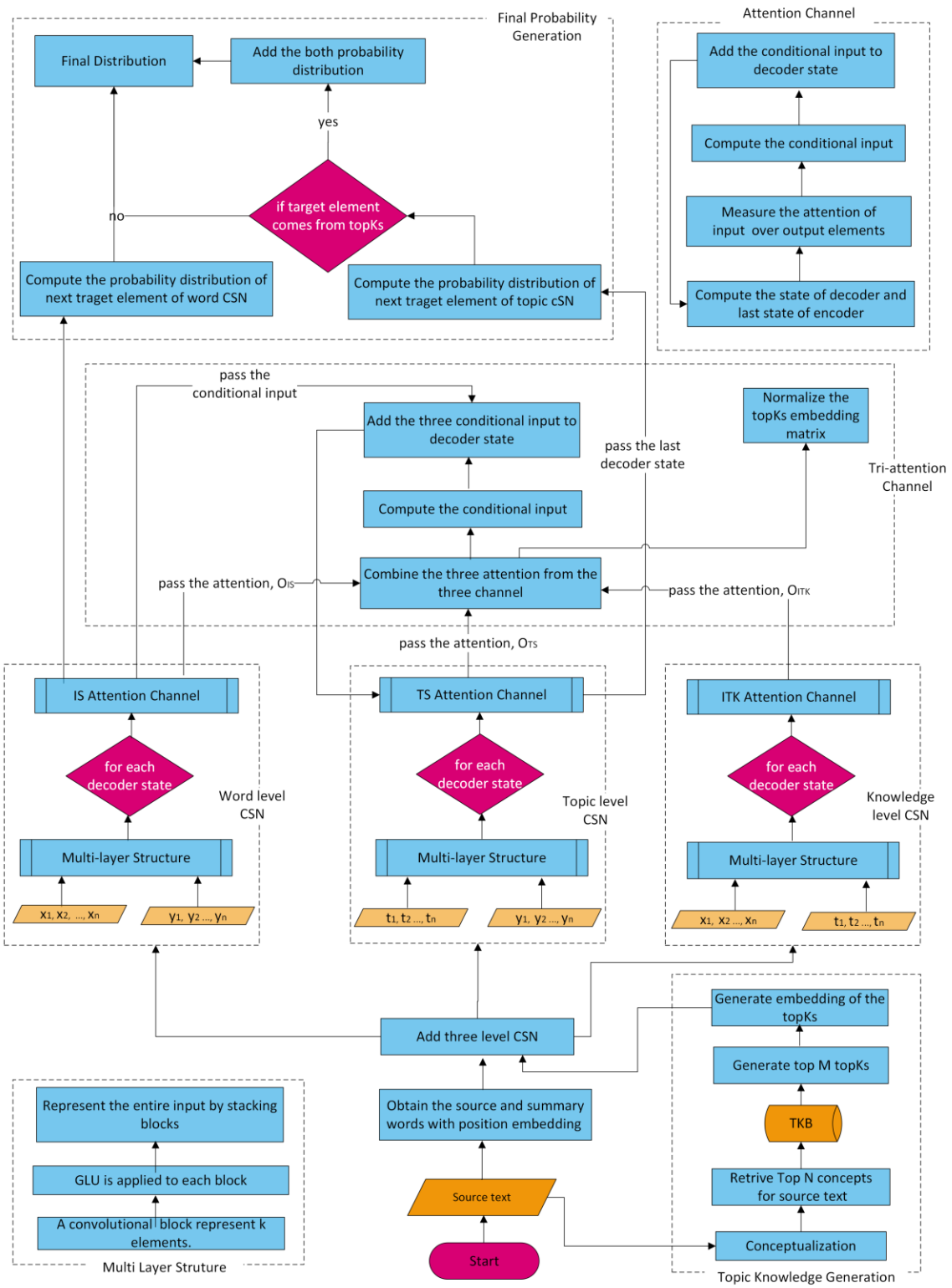


Figure 6.9: A flowchart showing the dataflow of the KTOPAS model.

by piling the segments of text together. In the multiplayer structure, a convolutional block which acts as a unit represents k elements using word embeddings with their position, then GLU is applied to transform k elements of each block into a single output. After this, the entire text is represented by stacking one by one. Topic knowledge generation produces the knowledge-powered topic information (topKs) from the TKB. First, we retrieve the relevant and informative background knowledge using the conceptualization algorithm. Next, we produce the topKs using this knowledge information and obtain the embedding of the topKs. An attention mechanism is a method which represents three channels: IS, TS and ITK channel to compute each state of the encoder and decoder, measure the attention, compute the conditional input and finally feed the conditional input into the decoder state of the word, knowledge and topic level CSN respectively. The tri-attention channel combines the three attention models from the word, knowledge and topic level CSN using SoftMax.

The final probability distribution produces the probability distribution of the next target element of the summary output at each state at the decoder of the word and topic level CSN. x_i; t_i; y_i are the embedding of the source, topKs and summary elements respectively. First, this model adds three CSN: word, knowledge and topic, embeds the source and summary, retrieves the topKs from the topic knowledge generation and embeds the topKs to the CSN. This includes the multi-layer structure and the attention channel with each CSN which measures attention jointly for each state and passes the attention information to the tri-attention channel, and finally generates the probability distribution to predict the next target element for each state.

6.4.6. Learning

We train the α, β and γ through the network jointly. We calculate α, β and γ using the following formula:

$$\rho = \sigma(W^t[\alpha, \beta, \gamma] + b) \qquad 6.28$$

where σ is the sigmoid function. Once the final probability is computed, we train our model using three steps which is introduced by Paulus [37]. In the first step, we exploit the cross-

entropy to minimize the objective function in our model. The standard maximum likelihood objective is obtained to minimize the loss in the training and defined as follows:

$$L_{ml} = -\sum_{i=1}^{L} \log\, p_\theta(\mathbf{y}_i^* \mid \mathbf{y}_1^*, \mathbf{y}_2^*, \ldots, \mathbf{y}_{i-1}^*) \quad 6.29$$

Then, reinforcement loss is minimized in the training using the reward $r(\acute{y})$ and $r(\hat{y})$ as follows:

$$\mathcal{L}_{RL} = \sum_{t} -\log\, p_\theta^*(y_t \mid y'_{t-1}, s_t, c_{t-1}, \mathbf{X}) \times \big(r(\hat{y}_1, \cdots, \hat{y}_T) - r(y'_1, \cdots, y'_T)\big) \quad 6.30$$

Finally, we define a mixed training objective L_{mix} [84] for further minimization by associating the policy learning objective function L_{rl} and the original maximum likelihood L_{ml} which is given below. We tarin the KTOPAS with respect a mixed training objective L_{mix} with the parameter $\gamma \epsilon$ [0,1].

$$L_{mixed} = \gamma L_{rl} + (1-\gamma) L_{ml} \quad 6.31$$

6.5. System Evaluation

We set up the implementation environment and develop our work in this environment. We implemented our proposed scheme on two datasets and evaluated the results with other baselines. First, we evaluate the accuracy of the generated topic information using KPTopicM. Then, we evaluate the accuracy of the generated results of our proposed summarization model KTOPAS. Next, we run an ablation study to learn the effect of each contribution in the model KTOPAS and compare the computation cost with the baseline. Finally, we discuss the advantages, limitations, findings and novelty of our model.

Table 6.1: Basic statistics of the CNN/Daily Mail and Gigaword dataset.

Datasets	CNN/Daily Mail			Gigaword		
	Train	Valid	Test	Train	Valid	Test
Documents	287 K	13 K	11 K	3.8 M	189 K	2 K
Avg.Len.Doc. (word)	790	769	777	31	31	29
Ave.Len.Ref. (word)	55	61	58	8	8	9

Avg.Len.Doc. indicates that average number of words in a document. Avg.Len.Ref. indicates that average number of words in a reference summary.

6.5.1. Datasets

Our experiments are conducted over two datasets: Gigaword [123] and CNN/Daily Mail [42] for our topic model KPTopicM and summarization KTOPAS. In the Gigaword datasets, summaries are generated by combining the first sentence of each source article and its headline. This dataset has 3.8M training samples, 400k validation samples, and 400k test samples. CNN/Daily Mail contains news articles and the corresponding human-written summaries and has 287K training samples, 13K validation samples, and 11K test samples. Table 6.1 show the basic statistic of the dataset we used for our experiments.

6.5.2. Automatic Evaluation Methods

We describe the evaluation method used to measure the performance of the proposed model KPTopicM and KTOPAS.

Perplexity: We evaluate the results of our proposed KPTopicM with LDA by comparing the performance using perplexity. A lower score of perplexity indicates better performance in generalization. We compute perplexity using the following equation:

$$perplexity = \exp\left\{-\frac{\sum_{d=1}^{M} \log\, p(\mathbf{w}_d|\varphi,\alpha)}{count\ of\ token}\right\} \quad 6.32$$

where w denotes the words in document d for the given topics φ and the hyperparameter α for topic-distribution θ_d of documents.

Topic Coherence. We evaluate our KPTopicM with LDA using topic coherence [90]. We compute the topic coherence score for given a topic t of top m words (z_1, z_2, . . ., z_m) with the highest probabilities *P(w|t)* as follows:

$$C\left(t; Z^{(t)}\right) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{f\left(z_m^{(t)}, z_l^{(t)}\right) + 1}{f\left(z_l^{(t)}\right)} \quad 6.33$$

Let f (z) be the frequency of word t in the document and f (z, $\acute{z}$) is the number of documents where the words z and $\acute{z}$ co-occur. A higher coherence score indicates higher topic quality.

Table 6.2: Example of topic words for LDA [50], KB-LDA [105], KPTopicM trained over two datasets (TOP-10 WORDS ARE SHOWN).

LDA [50]	KB-LDA [105]	KPTopicM
government, election, politics, leader, opposite, people, power, parliament, democrats, climate	party, trump, poll, vote, election, debate, change, candidate, minister, state	election, debate, candidate, campaign, majority, win, party, political, vote, president
product, service, market, industry, farm, company, busy, corporation, fund, customer	Executive. company, market, stock, research, corporation, profile, chief, quote, industry	Investment, business, corporation, market, product, employee, management, farm, organization, profit
military, time, government, security, troops, war, like, country, attack, right	people, guns, force, government, article, military, know, weapons, war	army, weapons, terrorist, war, violent, death, loss, country, military, escalate.
patients, time, good, disease information, heading, people, think, medical, diagnosis	patients, blood, good, disease diagnosis, medical, care, heart, physical, examination	patient, history, treatment, disease, pain, examination, information, diagnosis, blood, care
game team think time hockey play players good games Friday	game team article football play league players good games season	Play, win, fun, score, sport, team, rule, football, loose, team
Japan, us, plant, Korea, oil, deal, disaster, nuke, earthquake, radiation	Japan, plant, disaster, nuke, crisis, power, oil, radiation, quake, nuclear,	Japan, earthquake, tsunami, disaster, shake, loss, nuclear, crisis, radiation, Asia

The incorrect topic words for each topic in table are marked in orange.

ROUGE: We use the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [87] metric to evaluate our summarization model. ROUGE(RG) is a set of metrics to compare the quality of the generated summary with reference (human-written) summaries. We compare the quality by counting the number of times a series of n-grams (mostly two and three) overlap the generated summaries with the reference summaries. The score is computed as:

$$\text{ROUGE} - n = \frac{\sum_{S \in \text{Ref}} \sum_{gram_n} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{Ref}} \sum_{\text{gram}_n} \text{Count}(\text{gram}_n)} \quad 6.34$$

We measure several ROUGE scores ROURG-1 (RG1), ROURG-2 (RG2) and ROURG1 (RGL) for unigram, bigram, and the longest common subsequence respectively.

6.5.3. Baseline

We evaluate the KPTopicM topic model with the KB-LDA [33] and LDA [28] model. KB-LDA [33] is a simple knowledge-based LDA model which incorporates input elements represented by concepts in terms of the source document while our proposed model KPTopicM incorporates input elements represented by semantically relevant and coherent

concepts which are retrieved using the proposed conceptualization algorithm. We evaluate our model KTOPAS with different baselines which are described as follows. RNN-ATS [36] is an RNN-based ATS model with an attentive encoder and decoder. RNN- ATS+ [40] uses a further extension with additional features for the optimization of RNN-ATS. LEAD [38] is also an attention based RNN model proposed by Nallapati (2017) which achieves better results for a large vocabulary. Deep-RL [37] uses the policy gradient algorithm for ATS for learning by exploiting metric rewards at the sequence level. Deep-RL+ML [45] weight the mixed loss for stability and fluency of the summaries. PTGN [42] is a pointer generator method which allows words to be copied from the original document. SEASS [127] is the extension of the RNN sequence-to-sequence model for selective encoding. CSN-ATS [114] is a text summarization model based on a convolutional sequence neural network. Mats [33] is a multi-task learning approach for ATS. We then compare the improvement of our model step by step from the base line TopicCSN [3] to KTOPAS. We also evaluate our current model against our previous model TEXSCTTA discussed in Chapter 5.

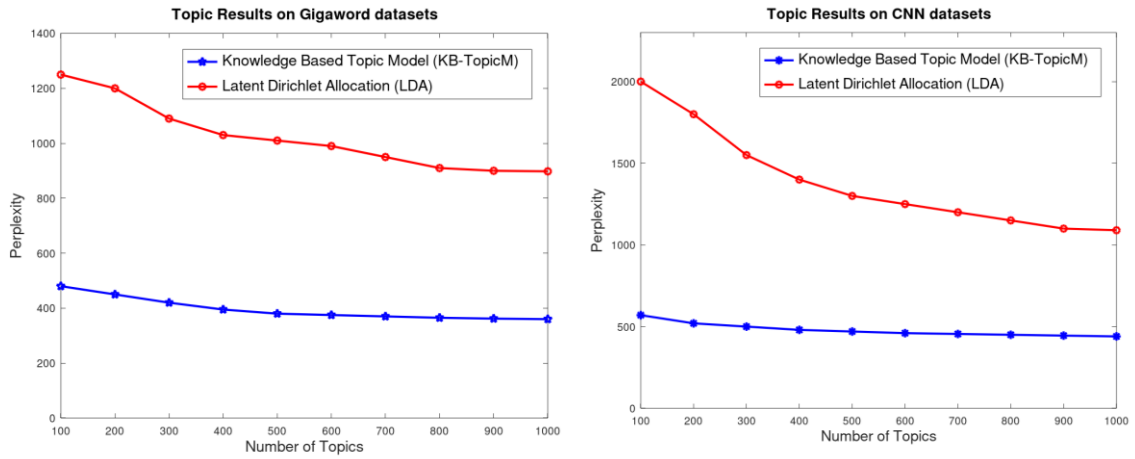


Figure 6.10: Perplexity of LDA and KPTopicM from 100 to 1000 topics over CNN/Daily Mail and Gigaword datasets.

The lower the score, the better the topic quality.

TopicCSN [3] is a summarization model which incorporates topic information using CNN. This model obtains topic from source documents using the LDA technique rather than background knowledge information. DTopCSN, described in section 4.3.2, measures attention from two aspects: word and topic level while our improved KTOPAS model

measures attention jointly from three aspects: word, knowledge, and topic level through the tri-attention channel to generate more coherent summaries.

6.5.4. Implementation Setup

We implemented our scheme using PyTorch and Fairseq in the Python 3.7 environment on a university GPU Linux cluster. We use the Stanford core NLP package NLTK1 to preprocess the text and pyrouge2 to compute the ROUGE score. We use ConceptNet [50] to retrieve the concept or knowledge for conceptualization. We extract the top 50 concepts per document to choose the topic information using KPTopicM. We set both Dirichlet prior α and β to .01. For the Gib sampler, the number of iterations is set to 500. We chose 5 to 50 topK for each topic. We obtained 512 topics based on topKs. We initialize the dimension of the word embedding to 256. The dimension of the GRU hidden states for both the encoder and decoder is set to 512. We limit the size of the input and output vocabulary to 110,000. For training, the initial learning rate is set to 0.001 utilizing Adadelta. The batch size is set to 50 and the training data are randomly shuffled at every epoch. The scaling factor is set to 0.1. The Gensim package is used for measuring the perplexity and coherence of the topic models.

Table 6.3: Topic coherence of various size of topics (N) over CNN/Daily and Gigaword datasets.

	CNN/Daily Mail Datasets			Gigaword Datasets		
N	5	10	15	5	10	15
LDA [28]	-210.75	-960.69	-2488.65	-260.68	-1298.43	-3352.56
KP-LDA [33]	-180.65	-920.42	-2375.85	-230.74	-1206.22	-3211.58
KP-TopicM	**-172.52**	**-903.12**	**-2295.73**	**-201.23**	**-1148.30**	**-3174.63**

The more relevant topics achieve a higher score. The best scores are expressed as boldface.

6.5.5. Analysis of Experiment Topic Results

Daily and Gigaword datasets. We obtain the top 50 concepts for each topic with the highest probabilities. We compute the perplexity of the two models (KPTopicM and LDA) over the datasets from 100 to 1000 number of topics. The curve in Figure 6.10 represents the value of perplexity over the number of topics for the two models. We can see that perplexity declines as the number of topics in both datasets increases. The value of perplexity in KPTopicM is much smaller than LDA, which indicates that KPTopicM performs

significantly better than LDA. KPTopicM provides semantically well-formed and relevant topics to the KTOPAS summarization model to facilitate the generation of meaningful and informative summaries which is discussed in the next section. Table 6.2 shows examples of topics with 10 words sets obtained by the LDA, KB-LDA and KPTopicM techniques. We then compute the topic coherence for KPTopicM, KBLDA and LDA and compare the results. Table 6.3 illustrates the topic coherence score for a different number of words in topics over the CNN/Daily Mail and Gigaword datasets. A higher topic coherence score implies more coherent and consistent topic words in the topic. We can see from Table 6.2 that our model achieves higher quality results than the KBLDA and LDA model in terms of coherence score. This is expected because this model incorporates the latent knowledge of the document and captures its semantic relevance to the document. We can see from Table 6.1 that KPTopicM topics are more consistent and coherent than the other techniques.

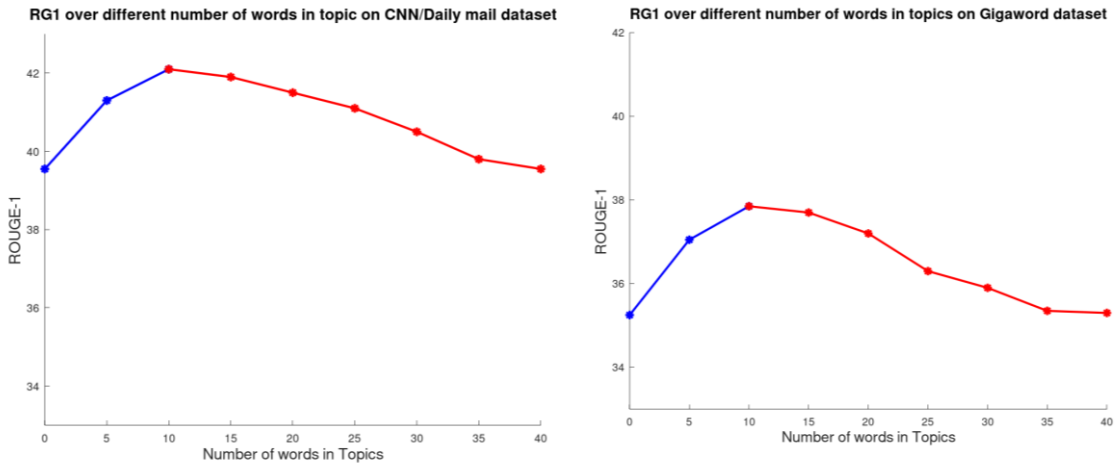


Figure 6.11: ROUGE 1 results score of KTOPAS over the number of topics on two different datasets.

The blue curve indicates the increase of the ROUGE score while the red curve indicates the decline of the ROUGE score.

6.5.6. Analysis of Experimental Summary Results

We evaluate the summary results of the KTOPAS using the following steps. First, we evaluate the summary results on a different number of topic knowledge sets for a topic reflected by the RG1 score. Then, we analyze the effect of topic knowledge in KTOPAS. Finally, we compare the summary results of our model KTOPAS with baselines.

Results on Different Size of Topic

In this section, we evaluate the accuracy of the generated summaries in relation to topic size. Topic size means the number of word or concepts in a topic. We compute the ROUGE-1(RG1) score of different baselines and KOPAS for topic sizes ranging from 5 to 10. Figure 6.11 shows the results of the RG1 score for the different sized topics for the Gigaword and CNN datasets. We can see that KTOPAS achieves a higher score when the topic information has been incorporated than when no topic words have been incorporated. The RG score of the summary results increases consistently with an increase in topic size and reaches the highest score for a topic size of 10. Then, RG score begins to decline after this until topic size of 35, but this score still higher than KTOPAS obtained when no topic information is incorporated. After that, RG score dropped to same level as it was for the model with no topic at topic size 40. We see that the most accurate summary results are obtained when topic size is 10. Therefore, we chose a word set of 10 to train the parameters.

Table 6.4: RG-1, RG-2, and RG-L metric over the CNN/Daily corpus for different approach of text summarization.

Methods	RG-1	RG-2	RG-L
RNN-ATS [36]	29.56	11.31	26.41
RNN-ATS+ [40]	29.78	11.88	26.94
LEADS [38]	35.32	0	0
Deep-RL [37]	35.80	16.62	32.44
Deep+RL+ML [45]	35.17	16.76	32.46
PTGN+ [42]	33.44	16.1	31.45
CNN-ATS [114]	35.88	17.48	33.29
SLASS [127]	35.93	17.51	33.35
Mats [132]	35.54	17.09	32.93
LDA-ConvTSM	36.38	17.48	33.40
KB-LDA-ConvTSM	36.57	18.50	33.92
TEXSCTTA	37.56	18.62	33.93
KBTOPAS	**37.85**	**18.71**	**33.96**

Higher score is displayed in boldface.

Compare Results with Baselines

We analyzed the performance of our KTOPAS over the CNN/Daily Mail and Gigaword datasets. First, we measured the RG1, RG2, and RGL metrics of different state-of-the-art methods. Then, we measure the RG metrics of our approach in the way that has been extended: TopicCSN [52], DTopCSN and our proposed model KTOPAS. Table 6.4 and 6.5 shows the RG metrics of the different approach for the CNN/Daily Mail and Gigaword datasets. The results show that TopicCSN has a better RG metrics score than CSN-ATS

[55] which demonstrates that topic information helps to produce a better summary. Incorporating topKs in DTopCSN improves the results more than TopicCSN. This demonstrates that coherent and meaningful topics based on conceptual information contribute to better results since the conceptualization algorithm provides coherent and informative knowledge of concepts for the document which cannot be found in the source document and the KPTopicM algorithm provides quality topic information using conceptual information.

Table 6.5: RG-1, RG-2, and RG-L metric over the Gigaword corpus for different approach of text summarization.

Methods	RG-1	RG-2	RG-L
RNN-ATS [36]	35.45	13.31	32.71
RNN-ATS+ [40]	35.61	13.83	35.48
LEADS [38]	39.15	15.65	35.56
Deep-RL [37]	40.95	15.83	36.35
Deep+RL+ML [45]	40.09	15.84	36.52
PTGN+ [42]	39.55	17.18	36.65
CSN-ATS [114	39.86	17.25	36.63
Mats [132]	40.71	18.12	36.73
LDA-ConvTSM	40.38	18.82	36.64
KB-LDA-ConvTSM	41.54	19.50	37.92
TEXSCTTA	41.39	19.34	38.43
KBTOPAS	**42.10**	**20.01**	**38.45**

Higher score is displayed in boldface.

KTOPAS which uses the tri-attention channel achieves higher scores for the RG metrics than DTopCSN since the DTopCSN model does not observe the relevancy of the topKs over the input elements. This shows that the incorporation of more relevant topics in terms of source elements improves the performance of the summary results based on R metrics. It can be seen that the RG scores increase gradually in each step by enriching topics with coherent latent knowledge in KTOPAS. The words in blue in the KTOPAS summary are captured from the topic information and are associated with the pink words in the corresponding source document. We further evaluated KTOPAS against the other baselines and the ROUGE scores are shown in Table 6.4 and 6.5. The results show that the topKs, knowledge-powered topic level attention, tri-attention channel and the mixed learning procedure improve the quality of text summarization in terms of accuracy. The results in the that tables show that our KTOPAS model achieves the highest ROUGE scores and outperform the various baselines. Examples of the generated summaries of the various

models are shown in Table 6.7. We can see from the examples that some of the topK words appear correctly in the generated summaries after the topic information from TKB are merged in our model. These words do not come from the reference summaries or the source document. So, we can say that the tri-attention channel with a pre-trained TKB provides informative knowledge as the topic and improves the coherency of the summaries. KPTopicM provides the topic information to construct TKB and it also improves the effectiveness in terms of the accuracy of KTOPAS by generating meaningful topics. The results show our current model KTOPAS improves the performance compared to our previously proposed model TEXSCTTA, discussed in Chapter 5. This demonstrates that our extension of this model improves the results.

6.5.7. Ablation Study

In this section, we describe the ablation studies to investigate which of our individual improvements in the model contribute to the performance of the ATS in achieving better results. This study removes our individual contribution to the model and evaluates these ablations against each other. CSN+LDA (TopicCSN) [3] integrates the topic information into a CSN where the topics are retrieved using the statistic LDA topic model only. We retrieve the background knowledge using our proposed conceptualization algorithm for all ablations. We use five ablation models for the analysis. CSN + Concept-LDA: CSN with topic information based on the background knowledge instead of based on source words using the statistical LDA model. CSN+TKB: Utilizes our pretrained improved KPTopicM model (TKB) to obtain and incorporate the topic information based on background knowledge to a CSN model. CSN+ TKB + Dual Attention: Applying a dual high-level attention to the CSN+TKB model. CSN+TKB+Tri-Attention: Adding one more refined attention to the model for relevant topics to the source text. CSN+TKB+Tri-Attention+RL: Full model with knowledge--powered high-level topic attention cooperating with the RL objective. We also evaluate these ablations with RNN and CSN based baselines. Table 6.6 shows the results of each ablation and baseline based on the RG metrics on the validation set of the CNN/Daily Mail dataset. We observe from Table 6.6 that our base model: CSN-based ATS

Table 6.6: Ablation experiments investigating the effectiveness of topic information and the attention mechanism on the CSN model over the CNN/Daily Mail dataset.

No	Models	RG1	RG2	RGL	Tt
RNN based ATS method					
1	RNN-ATS [36]	35.61	13.83	35.48	12
2	PTGEN [42]	39.55	17.18	36.65	6.5
CNN based ATS method					
3	CSN-ATS [114]	39.86	17.25	36.63	
Our Ablations					
9	CSN+ Tri Attention + TKB+RL	**42.10**	**20.01**	**38.45**	**2.3**
8	CSN + Tri Attention + TKB	41.96(↓ .33%)	19.88(↓ .65%)	38.33(↓ .31%)	3.9
7	CSN + Dual Attention+ TKB	41.54 (↓ 1.01%)	19.65 (↓ 1.17%)	37.92 (↓ 1.08%)	3.6
6	CSN+TKB	41.32(↓ .53%)	19.49(↓ .82%)	37.69(↓ .61%)	3.0
5	CSN+ Concept-LDA	40.88 (↓ 1.42%)	19.29 (↓ 1.04%)	37.13 (↓ 1.51%)	3.0
4	CSN+LDA [3] (base model)	40.38 (↓ 1.24%)	18.82 (↓ 2.5%)	36.71 (↓ 1.06%)	2.9

Rows 1-2 and 3-4 of the table are the results of the other proposed sequential RNN and CSN based ATS models, respectively. Next, we show each improvement of the model and the effectiveness of the improvements step by step, indicating the decrease in percentage of the results score for each ablation of our model. The best results are shown in boldface for our proposed model improvements steps (rows 5-9). *Tt* is the training times (in hours) per epoch for each method and ablation.

(CSN-ATS) is more effective than RNN-based ATS. It is clear from the results that incorporating topic information in the model TopicCSN (CSN+LDA) provides better accuracy than the baselines. Now, we evaluate our five ablation models for the analysis. We see that our full model (CSN+ TKB+ Tri-Attention + RL) beats the novelty baseline. Then we compare the full model with CSN+ TKB+ Tri-Attention stage ablation and see a decrease in the performance of the results in terms of RG metrics when we remove the reinforcement learning approach from our model. Then, we evaluated the influence of additional attention by removing it from the CSN+ TKB+ Tri-Attention model. The results show that the performance of the model (CSN+TKB+Dual-Attention) drops by 1.08%, 1.17% and 1.08% for RG1, RG2 and RGL scores respectively. This indicates that utilizing the tri-attention channel drives the model to be more effective than the dual attention-based model. Next, we eliminate the high-level dual topic attention from the CSN+TKB+Dual-

Attention model and we observe a decrease in the performance of the model CSN+TKB, suggesting that high-level topic attention makes the model more effective. At a later stage, we drop the incorporation of topic information from our pretrained proposed KPTopicM model (TKB) and incorporate topic information based on knowledge using the LDA topic model into a CSN instead.

We observe that the performance of the model dropped by 1.24%, 2.5% and 1.06% for RG1, RG2 and RGL scores respectively when we do not merge the topic information from our TKB. Finally, we remove the use of background knowledge entirely while capturing topic information from the model. The results show that the performance of the model CSN-LDA (which incorporates topic information based on source words only) reduces by 1.24%, 2.5 % and 1.06% in terms of RG1, RG2 and RGL scores, respectively. This proves that utilizing the background knowledge from our conceptualization algorithm to capture topic information in the generated summaries is effective. The ablation study demonstrates that each contribution is important for our complete model, and the improvements are statistically significant on all metrics.

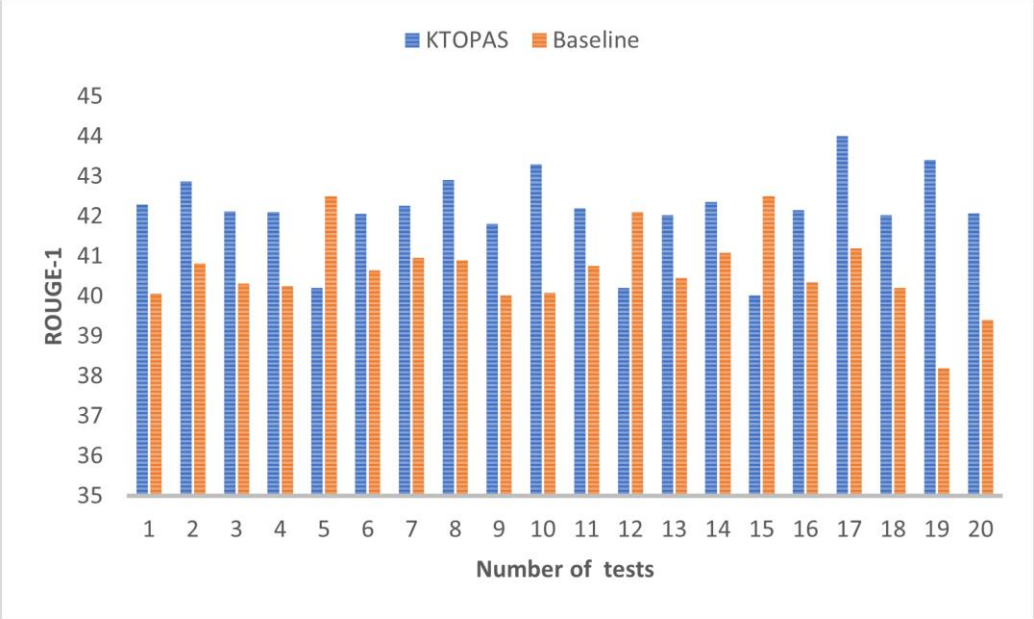


Figure 6.12: Statistical test on the sample of the CNN/DailyMail datasets reflected by the R1 score.

6.5.8. Computation Cost

We compute the time consumption of the training of our model and baselines. The training time of each model is shown per epoch and hour in the columns of Table 6.6 as T_t. It is clear from Table 6.6 that RNN-based ATS consumed much higher time to train the data than CSN-based ATS. The base CSN-ATS is 5.2 times faster than the base RNN-ATS model and 3 times faster than the PTGEN model. We can see from the results that the computation cost increases gradually from the base (CSN-LDA) model to the KTOPAS (CSN+Tri Attention+ TKB+ RL) due to the improvement in our model. However, the cost is still much lower than the base RNN (around three times) and PTGEN (around 1.6 times) model. This is because we use the CSN model for our ATS approach which supports the parallel computations while training.

6.5.9. Statistical Test

We run a statistical test over the CNN/Daily mail dataset to ensure the superiority of the proposed approach as reflected by the R1 score. We randomly sample 100 source documents each time and run the test 20 times on the test set and measure the R1 score for each sample. Our standard variance is 2.5 and p value <.006. We compare our results with the baseline mats [33] which have the highest scores compared to the other baselines we used for our evaluation. Figure 6.12 show the confidence interval is 95% of the R1 scores for each sample over the CNN/Daily mail datasets. Our hypothesis null testing shows the statistical significance of our model with respect to the baseline model mats [105] as given by the 95% confidence interval in the R1 score.

6.5.10. Discussions

TopicCSN is our base model which does not utilize background knowledge and high-level attention while generating summaries. The main advantage of our model is that we employ background knowledge in capturing topic information while generating summaries. We use our conceptualization algorithm to retrieve semantically relevant and salient background knowledge and obtain topic information based on the knowledge of document

using the three-layer LDA model (also called Concept-LDA). We can see from Figure 6.13 that when topic information is incorporated from Concept-LDA into CSN, the accuracy of the model CSN+Concept-LDA improves the performance as reflected by the RG scores (i.e., RG2 scores are increased by 2.5%) compared to base TopicCSN. This implies that our conceptualization algorithm supports our model to improve the performance by providing coherent and informative background knowledge to capture topic information in the summaries.

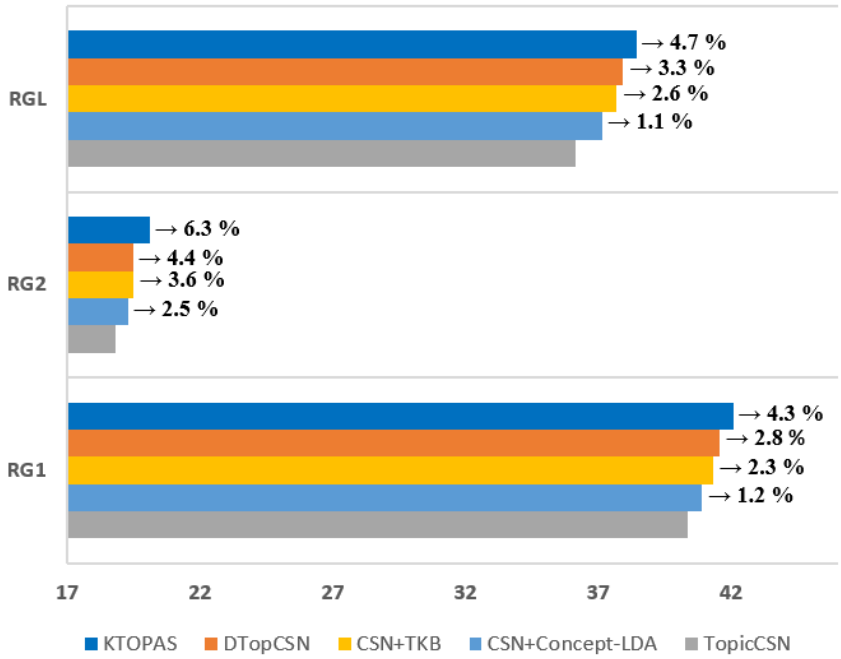


Figure 6.13: Analysis of the effect of topic knowledge and the attention mechanism on the improvement of our model from TopicCSN to KTOPAS over the CNN/Daily mail and Gigaword Datasets.

→indicates an increase in the percentage of the results compared to the base model TopicCSN.

After this, we improved the topic model with a four-layer architecture (also called KPTopicM) which not only captures the direct dependencies of the background knowledge or concepts, it also captures the indirect dependencies of words in the topic information through the conceptual information so that the model is able to generate more coherent and relevant topic information of the source document, whereas Concept-LDA only captures the direct dependencies of concepts in the topic information. When we incorporate topic information from the TKB (pre-trained KPTopicM) into a CSN, the performance of our improved model (CSN+TKB) increases (i.e., RG2 score improves by 3.6% as shown in Figure 6.13) which shows that our prior TKB is more efficient in providing coherent and

relevant topics in generated summaries since this trained model acknowledges the dependencies of the source words in conceptual information as well.

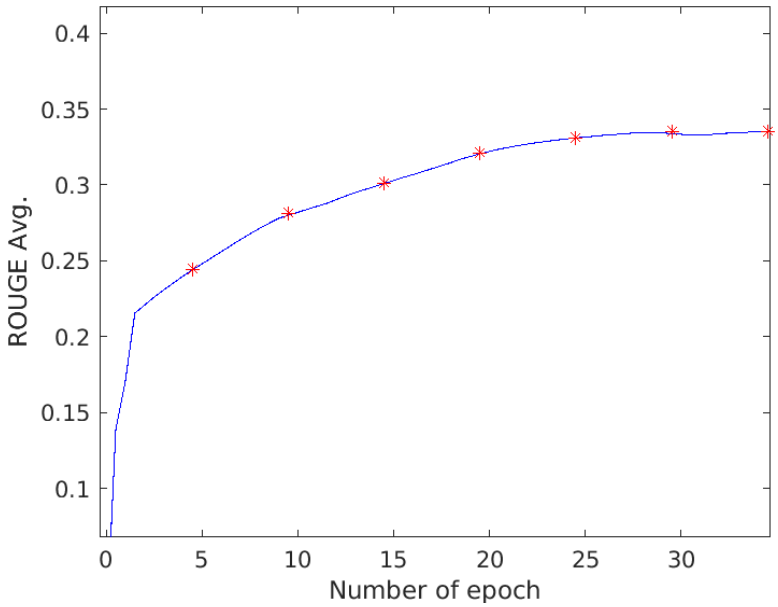


Figure 6.14: Learning curve of our model corresponding to average R1, R2 and RL scores over CNN/Daily Mail dataset with 40 epochs.

The next advantage of the model is that we utilize the high-level topic attention level to incorporate topKs into CSN+TKB. First, we use two CSNs: the word and topic level CSN in our model DTopCSN to capture the information of the attention of source elements and topic elements to summary elements respectively from two aspects jointly through a high-level dual topic attention mechanism while generating summaries. We see from Figure 6.13 that the performance of the model DTopCSN improves for RG metrics (i.e., RG2 scores increase to 4.4% compared to the base model TopicCSN). However, this model does not acknowledge the attention of topic knowledge over the source elements in the model which may lead to unconcise summaries that focus on topics which are irrelevant to the source text. Next, our improved model KTOPAS uses an additional knowledge level CSN in the model which learns the contextual information of the topic elements (topKs) in terms of source elements, learns the attention of the three CSNs jointly and introduces a tri-attention channel which combines these attentions using the softmax function. It is clear from Figure 6.13 that KTOPAS achieves higher RG1, RG2 and RGL results by up to 4.25%, 6.26% and 6.45% respectively than the base TopicCSN. This implies that using the

tri-attention channel enables the model to produce coherent and semantic topics in the summaries in terms of the source text.

Table 6.7: Summarization examples of source texts for various modes and KTOPAS.

Source Text	Hong Kong signed a breakthrough air services agreement with the United States on Friday that will allow U.S. airlines to carry freight to Asian destinations via the territory.
Reference	Hong Kong us sign breakthrough aviation pact
TopicCSN	The United States and Hong Kong have agreed on a deal to buy the territory of the United States.
DTopCSN	The United States and Hong Kong have signed a new deal.
KTOPAS	Hong Kong signs new deal over airline transportation.
Source Text:	Canada 's government is investigating a decision by the U.S.-owned Walmart retail chain to pull pajamas from its Canadian stores, international trade minister Art Eagleton said Wednesday.
Reference:	Walmart being probed by Canada for withdrawing Cuban goods.
TopicCSN	The Canadian government has announced that it is shutting up trade with Walmart.
DTopCSN	The world 's largest shopping retailer Walmart has been closed to the public in Canada.
KTOPAS	Canadian is investigating a decision by Walmart to withdraw trousers from their market.
Source Text	A fairly strong earthquake measuring a magnitude of 6.7 on the Richter scale rocked wide areas of central and western Japan Sunday, followed by four aftershocks, the meteorological agency said.
Reference	Earthquake shakes wide areas in Japan.
TopicCSN	The international space agency said it is very concerned about ongoing major earthquakes in Japan.
DTopCSN	Japan has been hit by a strong earthquake.
KTOPAS	Powerful earthquake shakes a wide area of Japan.
Source Text	Malaysian experts say they may have discovered a new species of <unk>, the world 's largest flower which is famous for its putrid smell, according to a report Thursday.
Reference	Malaysia probes possible new species of world's largest flower.
TopicCSN	One of the world 's most famous world, Malaysian, has been found dead in a lake in the Swiss city of famous.
DTopCSN	One of the world 's largest flowers has been found in Malaysian.
KTOPAS	Malaysia finds new species one of the worlds 's largest flower.

The words in blue in the KTOPAS summary are captured from the topic information and are associated with the yellow words in the corresponding source document. This topic information is generated from the latent know edge of the words in pink in the source document using the proposed topic model KPTopicM.

We observe that limiting the source document to less than 350 tokens (about 15 sentences) help the model to achieve significantly higher ROUGE scores than limiting the document to less than 700 tokens. This proves that our model achieves around three times faster training computation time than the RNN-based ATS model since the CSN-based model allows each state to be performed in parallel. Due to the limited resources in our experiments, we have not utilized pre-trained models such as BERT. To take advantage of our model, we specify the length of the summary is less than 100 during testing. We also see that topics with 10 words helps the model to perform better than topics with more than

10 words. We can see from Figure 6.14 that our model converges in 20 epochs on the CNN/Daily dataset. Furthermore, after applying reinforcement learning, our full model performs better than the baselines and our previously improved model.

In this chapter, we provided a thorough assessment of the evaluation in our summarization model KTOPAS over the datasets. We deduce that our conceptualization algorithm improves the performance of the topic results by providing coherent and informative background knowledge of the document. Also, the strong association of the background knowledge in the document in addition to the association of the background knowledge in the topic helps our KPTopicM topic model generate more coherent and consistent topics than the base topic models, such as LDA and KB-LDA. We reveal the importance of background knowledge in capturing topic information in the generated summaries. We train KPTopicM and used this learned data as a prior repository called TKB. It is clear from our results that the topic information from TKB assists the model to achieve higher accuracy. The experiment results show that utilizing the high- level topic dual attention in the model to measure the attention from the word and topic level CSN jointly (CSN +TKB+ dual attention) performs better than one attention mechanism such as TopicCSN. This demonstrates the effect of the high-level topic attention mechanism in our model. When we introduce a tri-attention channel which measures the attention of the word, topic and knowledge level CSN jointly in the model to help the model provide coherent, syntactic and semantic topic information in the generated summaries. It is proven that the CSN+ TKB+ tri-attention model achieves higher ac curacy than the model with dual attention (CSN+TKB+ dual attention). We generate a probability distribution to provide the information of a target element of the summary at the decoder state of the word and topic level CSN. Moreover, employing reinforcement learning based on the mixed training objective function shows the improvement in the performance of our model.

We find there is a research gap in utilizing background knowledge in capturing topic information while generating summaries. We propose a conceptualization algorithm to retrieve semantic and informative background knowledge, and KPTopicM to obtain coherent and consistent topic information using this knowledge. We construct a prior topic knowledge base (TKB) using the pre-trained KPTopicM model to provide coherent

knowledge-powered topic information to our ATS model. We highlighted the strong features in CSN-based abstractive summarization compared to RNN which reduces the computation cost and improves the performance of the ATS model in terms of accuracy. We explored the advantage of proposing an abstractive text summarization model which follows the same procedure as humans do and finds the challenges for this by identifying topics using background knowledge to generate human-like summaries using topic, contextual and semantic information. We introduce a high-level tri-attention mechanism to increase the chance of capturing more relevant latent semantic and contextual salient information from the source document in the generated summaries and also to produce coherent and semantically well-formed summaries. Moreover, we use a final probability distribution to predict the target element in the summary and reinforcement learning as a mixed training objective function to maximize our model. After utilizing our conceptualization algorithm, TKB, three level CSNs, the high-level tri-attention channel, probability distribution and reinforcement learning, our full model KTOPAS is able to generate coherent, concise and relevant summaries with word diversity and outperforms the novel baselines and the preceding improvements of our model.

6.6. Summary

In this work, we investigated the challenges of abstractive text summarization in identifying the salient and meaningful information from the document while generating a summary. We improve the summarization model so that it can handle these challenges from an attention-based CSN to a tri-attention based CSN by incorporating topics based on the semantic knowledge of the text. In particular, we proposed a topic model to provide knowledge-powered topic information to ATS and proposed a CSN-based summarization model to incorporate this topic information via the tri attention channel. Our experiment results demonstrate its superior performance over various baselines. We also shed light on how the model performance is affected by important topic and background knowledge of textual data. For future work, it would be interesting to further improve the summarization performance by developing a topic model using more structured conceptual information and test its robustness. (CSN+Tri Attention+ TKB+ RL) due to the improvement in our model. However, the computation cost of our model is three times lower than the base

RNN-S2S model and 1.6 times lower than PTGEN model. This is because, we use the CSN model for our ATS approach which support the parallel computations while training.

Chapter 7.

Conclusions and Further Research Directions

In this research, we have discussed the need for an automatic text summarization method due to the exponential growth in information on the Internet. We also investigated the issues and challenges in the existing methods and the strategies to handle them in the text summarization research fields. During the research, we identified the research gap of utilizing background knowledge in capturing topic information while generating summaries. We highlighted the strong features in abstractive summarization compared to extractive summarization which results in significant progress and improvement in the research on automatic text summarization. We explore the advantage of constructing an abstractive text summarization system which follows the same procedure as humans and identifies the challenges from the three aspects of the research in building this system. The three challenges are: i) representation of knowledge to enable the system to understand the text, ii) generation of relevant and coherent topic information of the source text using the knowledge to be included in the generated summaries, iii) the incorporation of topic information into the ATS approach to allow the system to generate meaningful and human-like summaries. The main goal of this thesis is to enable a system to summarize documents by tackling the challenges. We built a complete ATS system based on deep learning to summarize the document in a way that resembles human-written summaries which resolves the aforementioned three challenges step-by-step using our proposed tasks.

7.1. Representation of Knowledge

Machines have a limited understanding of text as they do not have the knowledge that humans have. Knowledge base systems such as DBpedia, WordNet and ConceptNet can provide knowledge of a term to machines. However, most of the information as knowledge from these sources is not fully machine interpretable or informative. Hence, this is important to represent knowledge so that machines can read and apprehend the knowledge of text. Our main objective in addressing this challenge is to build a knowledge base system which can provide fully machine interpretable and meaningful information. In chapter 3,

we constructed an ontology-based knowledge-based system (OMRKBS) to handle the challenges in representing machine-readable information. To do this, we first extract knowledge from various sources and obtain the representation of knowledge as rich structured information (RSI). We use NLP techniques to preprocess the information to RSI. We then map this RSI into OMRBS by utilizing the natural language independent knowledge representation (NLIKR) scheme. NLIKR regards each word as a concept and each concept is defined by relating it with other concepts. So, we first discover each word in the RSI as a concept and its relationship in the RSI, and then the RSI is mapped as their relationship among concepts in OMRKBS. We use the mapping expression to map the RSI information in OMRKBS. Finally, OMRKBS enables the system to read and apprehend the knowledge about text by providing important features, and machine readable and informative information.

7.2. Topic Generation

Systems often have difficulty generating coherent and meaningful topic information because they focus on irrelevant information on the source text. Recently, the classic LDA topic model based on source words of the document has been a very popular topic model which can provide topic information on the source text. However, this information is still inconsistent in terms of the source text. This model considers only the words in the source text but does not consider background knowledge while identifying the topic information. Our main goal to address this issue is to incorporate the background knowledge into the statistical topic model so that the machine can fill the gap in the background knowledge in topic information. We retrieve the background knowledge from knowledge bases such as OMRKBS, ConceptNet and Probase. We introduce the conceptualization algorithm to compute the distribution of background knowledge as concepts of the text. In chapter 4, we use the statistical LDA model to generate topic information based on background knowledge instead of words in the source text. However, this approach does not acknowledge the word dependencies in the background knowledge while generating topics which may result in inconsistency in the topic information. Therefore, in chapter 5, we extend our research and propose the knowledge-powered topic model (KPTopicM) which incorporates the distributional information of background knowledge into the statistical

topic model to generate coherent and consistent topic information. This model acknowledges the word dependency in the background knowledge. We then use the pretrained data based on LDA (based on background knowledge) or KPTopicM as a prior topic knowledge base (also called TKB) to provide the topic information for our proposed ATS model.

7.3. Future Work

The research presented in this thesis can provide interesting directions for further research as follows.

- In chapter 4, a machine-readable knowledge base system is presented where we define concepts and their relationship. However, we only consider the structural information using NLP to enable the machine to read the information. We consider only concept-level similarity but have not considered sentence-level similarity or information while mapping the information in OMRKBS to find the connection among the terms in OMRKBS. We can utilize sentence-level similarity to capture and preserve the semantic information in OMRKBS.
- We proposed a deep learning based ATS system called a Joint Knowledge-based Topic Level Attention for a Convolutional Sequence Text Summarization System using Natural Language Representation (KTSNR). However, this model only uses word-level embedding and does not utilize sentence-level embedding using pretrained models such as BERT, the universal sentence encoder and so on. We can utilize pretrained sentence-level embedding to embed sentences of the source and summarization outputs.
- In chapters 4 and 5, we identify topic information based on the background knowledge of the source text. Instead of using the background knowledge of the source text to determine the topic, we can use the background knowledge of widely used pretrained summary models such as XSUM, PTGEN or BERTSUMEXT.
- Our pretrained KTOPAS model can be used to improve the classification and clustering problem in the NLP research area.

- Our system summarizes single documents but not multiple documents. In the future, we can focus on summarizing multiple documents to address the research gap on background knowledge.
- Our research has high time complexity to train our system. In the future, we will focus on improving the time complexity.
- In this research, we focus on text summarization using background knowledge, however, we can move our research direction to text generation using background knowledge and topic information.
- In this research, we only use the CSN advantage for our summarization model. In the future, we will try to utilize the advantage of the LTSM-based RNN by combining this with the CSN model to measure the inter-attention in the source text while generating summaries.

7.4. Abstractive Text Summarization

Systems face challenges in generating concise and coherent summaries because of their failure to identify coherent and relevant latent topic information. Recently, a deep learning based ATS has attracted much research interest because of its significant achievements in ATS. However, most approaches do not utilize background knowledge in their models which results in the failure to understand the text and identify topic information while generating summaries. Our main objective is to resolve these challenges in incorporating topic information based on background knowledge to bridge the gap between the topic information and background knowledge of the document in the summary. We use machine readable knowledge base systems such as OMRKBS, Probase, ConcpetNet and TKB to learn about the text and obtain topic information on the source text. In chapter 4, we propose a convolutional-based ATS model with high-level topic attention where we use a pretrained LDA topic model based on background knowledge as TKB to retrieve the topic information. However, due to the inconsistency in the topic information in the generated summaries, we extend our research in chapter 5. In this chapter, we use the pretrained KPTopicM as TKB to retrieve the topic information for the knowledge. We introduce a high-level topic attention based on background knowledge to incorporate topic

information. A convolutional sequence network (CSN) has some advantage over the recurrent sequence network. We use CSN for our proposed ATS model. We call this convolutional sequence based ATS with high-level topic attention (KTOPAS) which enables the system to provide coherent and meaningful summaries.

Bibliography

[1] Gambhir, M. and Gupta, V., 2017. Recent automatic text summarization techniques: a survey. Artificial Intelligence Review, 47(1), pp.1-66.

[2] Radev, D. R, Hovy, E. and McKeown K., 2002. Introduction to the special issue on summarization. Computational Linguistics, 28(4), pp. 399-408. doi: 10.1162/ 089120102762671927.

[3] Narayan, S., Cohen, S.B. and Lapata, M., 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. arXiv preprint arXiv:1808.08745.

[4] Jones, K.S., 2007. Automatic summarising: The state of the art. Information Processing & Management, 43(6), pp.1449-1481.

[5] Alguliyev, R., Aliguliyev, R. and Isazade, N., 2016, October. A sentence selection model and HLO algorithm for extractive text summarization. In 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT) (pp. 1-4). IEEE.

[6] Moratanch, N. and Chitrakala, S., 2017, January. A survey on extractive text summarization. In 2017 international conference on computer, communication and signal processing (ICCCSP) (pp. 1-6). IEEE.

[7] Gupta, V. and Lehal, G.S., 2010. A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence, 2(3), pp.258-268.

[8] Dalal, V. and Malik, L., 2013, December. A survey of extractive and abstractive text summarization techniques. In 2013 6th International Conference on Emerging Trends in Engineering and Technology (pp. 109-110). IEEE.

[9] Guan, W., Smetannikov, I. and Tianxing, M., 2020, October. Survey on automatic text summarization and transformer models applicability. In 2020 International Conference on Control, Robotics and Intelligent System (pp. 176-184).

[10] Lin, H. and Ng, V., 2019, July. Abstractive summarization: A survey of the state of the art. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 9815-9822).

[11] Luhn, H.P., 1958. The automatic creation of literature abstracts. IBM Journal of research and development, 2(2), pp.159-165.

[12] Liu, S., Zhou, M.X., Pan, S., Qian, W., Cai, W. and Lian, X., 2009, November. Interactive, topic-based visual text summarization and analysis. In Proceedings of the 18th ACM conference on Information and knowledge management (pp. 543-552).

[13] Lin, C.Y. and Hovy, E., 2000. The automated acquisition of topic signatures for text summarization. In COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics.

[14] Zhang, C., Sah, S., Nguyen, T., Peri, D., Loui, A., Salvaggio, C. and Ptucha, R., 2017, November. Semantic sentence embeddings for paraphrasing and text summarization. In 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 705-709). IEEE.

[15] Barzilay, R., 2003. Information fusion for multidocument summarization: paraphrasing and generation. Columbia University.

[16] Harabagiu, S. and Lacatusu, F., 2010. Using topic themes for multi-document summarization. ACM Transactions on Information Systems (TOIS), 28(3), pp.1-47.

[17] Zajic, D.M., Dorr, B.J. and Lin, J., 2008. Single-document and multi-document summarization techniques for email threads using sentence compression. Information Processing & Management, 44(4), pp.1600-1610.

[18] Goldstein, J., Mittal, V.O., Carbonell, J.G. and Kantrowitz, M., 2000. Multi-document summarization by sentence extraction. In NAACL-ANLP 2000 Workshop: Automatic Summarization.

[19] Moratanch, N. and Chitrakala, S., 2016, March. A survey on abstractive text summarization. In 2016 International Conference on Circuit, power and computing technologies (ICCPCT) (pp. 1-7). IEEE.

[20] Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J. and Li, H., 2018, April. Generative adversarial network for abstractive text summarization. In Thirty-second AAAI conference on artificial intelligence.

[21] Le, H.T. and Le, T.M., 2013, December. An approach to abstractive text summarization. In 2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR) (pp. 371-376). IEEE.

[22] Inkpen, D. and Hirst, G., 2006. Building and using a lexical knowledge base of near-synonym differences. Computational linguistics, 32(2), pp.223-262.

[23] Stevens, R., Goble, C.A. and Bechhofer, S., 2000. Ontology-based knowledge representation for bioinformatics. Briefings in bioinformatics, 1(4), pp.398-414.

[24] Alani, H., Kim, S., Millard, D.E., Weal, M.J., Hall, W., Lewis, P.H. and Shadbolt, N.R., 2003. Automatic ontology-based knowledge extraction from web documents. IEEE Intelligent Systems, 18(1), pp.14-21.

[25] Wimalasuriya, D.C. and Dou, D., 2010. Ontology-based information extraction: An introduction and a survey of current approaches. Journal of Information Science, 36(3), pp.306-323.

[26] Rosario, B., 2000. Latent semantic indexing: An overview. Techn. rep. INFOSYS, 240, pp.1-16.

[27] Hofmann, T., 2001. Unsupervised learning by probabilistic latent semantic analysis. Machine learning, 42(1), pp.177-196.

[28] Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y. and Zhao, L., 2019. Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. Multimedia Tools and Applications, 78(11), pp.15169-15211.

[29] Bin, S. and Fang, L., 2010. A survey of topic evolution based on LDA. Journal of Chinese Information Processing, 24(6), pp.43-49.

[30] Pilault, J., Li, R., Subramanian, S. and Pal, C., 2020, November. On extractive and abstractive neural document summarization with transformer language models. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 9308-9319.

[31] Barde, B.V. and Bainwad, A.M., 2017, June. An overview of topic modeling methods and tools. In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 745-750). IEEE.

[32] Yang, Y., Downey, D. and Boyd-Graber, J., 2015, September. Efficient methods for incorporating knowledge into topic models. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 308-317).

[33] Yao, L., Zhang, Y., Wei, B., Qian, H. and Wang, Y., 2015, May. Incorporating probabilistic knowledge into topic models. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 586-597). Springer, Cham.

[34] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[35] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[36] Nallapati, R., Zhou, B., Gulcehre, C. and Xiang, B., 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.

[37] Paulus, R., Xiong, C. and Socher, R., 2017. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304.

[38] Nallapati, R., Zhai, F. and Zhou, B., 2017, February. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Thirty-First AAAI Conference on Artificial Intelligence.

[39] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[40] Rush, A.M., Chopra, S. and Weston, J., 2015. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.

[41] Khandelwal, U., Clark, K., Jurafsky, D. and Kaiser, L., 2019. Sample efficient text summarization using a single pre-trained transformer. arXiv preprint arXiv:1905.08836.

[42] See, A, Liu, P.J and Manning, C.D., 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, (1), pp. 1073–1083.

[43] Liu, Y. and Lapata, M., 2019. Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345.

[44] Gehrmann, S, Deng, Y. and Rush, A., 2018. Bottom-up abstractive summarization. in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium. pp. 4098–4109.

[45] Kryściński W, Paulus R, Xiong C and Socher R. (2018). Improving abstraction in text summarization, in the Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium. pp. 1808– 1817.

[46] Khanam S. A, Liu, F. and Chen YPP., 2019. Comprehensive structured knowledge base system construction with natural language presentation. International Journal of Computational Intelligence Systems. 13(1), pp. 904-913.

[47] Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G. and Musen, M.A., 2009. BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic acids research, 37(2), pp. W170-W173.

[48] Bair, A.H., Brown, L.P., Pugh, L.C., Borucki, L.C. and Spatz, D.L., 1996. Taking a bite out of CRISP. Strategies on using and conducting searches in the Computer Retrieval of Information on Scientific Projects database. Computers in nursing, 14(4), pp.218-24.

[49] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S. and Bizer, C., 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web, 6(2), pp.167-195.

[50] Speer, R., Chin, J. and Havasi, C., 2017, February. Conceptnet 5.5: An open multilingual graph of general knowledge. In Thirty-first AAAI conference on artificial intelligence.

[51] Wu, W., Li, H., Wang, H. and Zhu, K.Q., 2012, May. Probase: A probabilistic taxonomy for text understanding. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (pp. 481-492).

[52] Ji, H. and Grishman, R., 2011, June. Knowledge base population: Successful approaches and challenges. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies (pp. 1148-1158).

[53] Boas, H., 2017. Computational Resources: FrameNet and Constructicon. In B. Dancygier (Ed.), The Cambridge Handbook of Cognitive Linguistics (Cambridge Handbooks in Language and Linguistics, pp. 549-573.

[54] Hotho, A., Maedche, A. and Staab, S., 2002. Ontology-based text document clustering. KI, 16(4), pp.48-54.

[55] Maedche, A. and Zacharias, V., 2002, August. Clustering ontology-based metadata in the semantic web. In European conference on principles of data mining and knowledge discovery (pp. 348-360). Springer, Berlin, Heidelberg.

[56] Glimm, B., Horrocks, I., Motik, B., Shearer, R. and Stoilos, G., 2012. A novel approach to ontology classification. Journal of Web Semantics, 14, pp.84-101.

[57] Zhao, Y., Dong, J. and Peng, T., 2009. Ontology classification for semantic-web-based software engineering. IEEE Transactions on Services Computing, 2(4), pp.303-317.

[58] Hennig, L., Umbrath, W. and Wetzker, R., 2008, December. An ontology-based approach to text summarization. In 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (Vol. 3, pp. 291-294). IEEE.

[59] Wu, C.W. and Liu, C.L., 2003. Ontology-based Text Summarization for Business News Articles. Computers and their applications, 2003, pp.389-392.

[60] Lu, Y., Li, Q., Zhou, Z. and Deng, Y., 2015. Ontology-based knowledge modeling for automated construction safety checking. Safety science, 79, pp.11-18.

[61] Hasan, K.S. and Ng, V., 2014, June. Automatic keyphrase extraction: A survey of the state of the art. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1262-1273).

[62] Najmi, E., Hashmi, K., Malik, Z., Rezgui, A. and Khanz, H.U., 2014, November. ConceptOnto: An upper ontology based on ConceptNet. In 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA) (pp. 366-372). IEEE.

[63] Zghal, H.B. and Moreno, A., 2014. A system for information retrieval in a medical digital library based on modular ontologies and query reformulation. Multimedia tools and applications, 72(3), pp.2393-2412.

[64] Gorskis, H., Aleksejeva, L. and Polaka, I., 2016. Database analysis for ontology learning. Procedia Computer Science, 102, pp.113-120.

[65] Copestake, A., 1990, June. An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary. In First international workshop on inheritance in NLP.

[66] Miller, G.A., 1998. WordNet: An electronic lexical database. MIT press.

[67] Nadkarni, P.M., Ohno-Machado, L. and Chapman, W.W., 2011. Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), pp.544-551.

[68] Navigli, R. and Ponzetto, S.P., 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial intelligence, 193, pp.217-250.

[69] Nakhla, Z. and Nouira, K., 2017. Automatic approach to enrich databases using ontology: Application in medical domain. Procedia computer science, 112, pp.387-396.

[70] Martinez-Rodriguez, J.L., Lopez-Arevalo, I. and Rios-Alvarado, A.B., 2018. Openie-based approach for knowledge graph construction from text. Expert Systems with Applications, 113, pp.339-355.

[71] Kollia, I., Glimm, B. and Horrocks, I., 2011, May. SPARQL query answering over OWL ontologies. In Extended Semantic Web Conference (pp. 382-396). Springer, Berlin, Heidelberg.

[72] Liu, F., Khanam, S.A. and Chen, Y.P.P., 2020. A Human-Machine Language Dictionary. International Journal of Computational Intelligence Systems, 13(1), pp.904-913.

[73] Kim, C.J. and Nelson, C.R., 1999. State-space models with regime switching: classical and Gibbs-sampling approaches with applications. MIT Press Books, 1.

[74] Sherstinsky, A., 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, pp.132306.

[75] Kröse, B., Krose, B., van der Smagt, P. and Smagt, P., 1993. An introduction to neural networks.

[76] Zell, A., 1994. Simulation neuronaler netze, 1(5.3). Bonn: Addison-Wesley.

[77] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pp. 3104-3112.

[78] Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[79] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[80] Gehring, J., Auli, M., Grangier, D., Yarats, D. and Dauphin, Y.N., 2017, July. Convolutional sequence to sequence learning. In International Conference on Machine Learning, pp. 1243-1252.

[81] Oord, A.V.D., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A. and Kavukcuoglu, K., 2016. Conditional image generation with pixelcnn decoders. arXiv preprint arXiv:1606.05328.

[82] Dauphin, Y.N., Fan, A., Auli, M. and Grangier, D., 2017, July. Language modeling with gated convolutional networks. In International conference on machine learning, pp. 933-941.

[83] Khanam, S.A., Liu, F. and Chen, Y.P.P., 2019. Comprehensive structured knowledge base system construction with natural language presentation. Human-centric Computing and Information Sciences, 9(1), pp.1-32.

[84] Benferhat, S., Dubois, D. and Prade, H., 1997. Some syntactic approaches to the handling of inconsistent knowledge bases: A comparative study part 1: The flat case. Studia Logica, 58(1), pp.17-45.

[85] Wilson, M., 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. Behavior research methods, instruments, & computers, 20(1), pp.6-10.

[86] Vitrià, J., Radeva, P. and Aguiló, I. eds., 2004. Recent Advances in Artificial Intelligence Research and Development.

[87] Riloff, E., 1993, July. Automatically constructing a dictionary for information extraction tasks. In AAAI (Vol. 1, No. 1, pp. 2-1).

[88] Trinh, T.H. and Le, Q.V., 2018. A simple method for commonsense reasoning. arXiv preprint arXiv:1806.02847.

[89] Doing-Harris, K., Livnat, Y. and Meystre, S., 2015. Automated concept and relationship extraction for the semi-automated ontology management (SEAM) system. Journal of biomedical semantics, 6(1), pp.1-15.

[90] Alobaidi, M., Malik, K.M. and Sabra, S., 2018. Linked open data-based framework for automatic biomedical ontology generation. BMC bioinformatics, 19(1), pp.1-13.

[91] Qawasmeh, O., Lefranois, M., Zimmermann, A. and Maret, P., 2018, June. Computer-assisted ontology construction system: Focus on bootstrapping capabilities. In European Semantic Web Conference, pp. 60-65.

[92] Bast, H., Björn, B. and Haussmann, E., 2016. Semantic search on text and knowledge bases. Foundations and Trends in Information Retrieval, 10(2-3), pp.119-271.

[93] Khanam, S.A. and Youn, H.Y., 2016. A Web Service Discovery Scheme Based on Structural and Semantic Similarity. J. Inf. Sci. Eng., 32(1), pp.153-176.

[94] Suomela, S. and Kekäläinen, J., 2005, March. Ontology as a search-tool: A study of real users' query formulation with and without conceptual support. In European Conference on Information Retrieval. pp. 315-329.

[95] Amato, F., Moscato, V., Picariello, A. and Sperlí, G., 2017, January. Kira: A system for knowledge-based access to multimedia art collections. In 2017 IEEE 11th international conference on semantic computing (ICSC), pp. 338-343.

[96] Musen, M.A., 2015. The protégé project: a look back and a look forward. AI matters, 1(4), pp.4-12.

[97] Rebele, T., Suchanek, F., Hoffart, J., Biega, J., Kuzey, E. and Weikum, G., 2016, October. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In International semantic web conference (pp. 177-185). Springer, Cham.

[98] Jastrzębski, S., Bahdanau, D., Hosseini, S., Noukhovitch, M., Bengio, Y. and Cheung, J.C.K., 2018. Commonsense mining as knowledge base completion? A study on the impact of novelty. arXiv preprint arXiv:1804.09259.

[99] Lenat, D.B., 1995. CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM, 38(11), pp.33-38.

[100] Young, T., Cambria, E., Chaturvedi, I., Zhou, H., Biswas, S. and Huang, M., 2018, April. Augmenting end-to-end dialogue systems with commonsense knowledge. In Thirty-Second AAAI Conference on Artificial Intelligence.

[101] Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P. and Ré, C., 2018, May. Fonduer: Knowledge base construction from richly formatted data. In Proceedings of the 2018 international conference on management of data (pp. 1301-1316).

[102] Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C. and Ré, C., 2015, July. Incremental knowledge base construction using deepdive. In Proceedings of the VLDB Endowment International Conference on Very Large Data Bases (Vol. 8, No. 11, p. 1310). NIH Public Access.

[103] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S. and McClosky, D., 2014, June. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

[104] Goldman, S.R. and Rakestraw Jr, J.A., 2000. Structural aspects of constructing meaning from text.

[105] Al-Zaidy, R.A. and Giles, C.L., 2018, April. Extracting semantic relations for scholarly knowledge base construction. In 2018 IEEE 12th international conference on semantic computing (ICSC) (pp. 56-63). IEEE.

[106] Angeli, G., Premkumar, M.J.J. and Manning, C.D., 2015, July. Leveraging linguistic structure for open domain information extraction. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 1, pp. 344-354.

[107] Coronado, S.D., Haber, M.W., Sioutos, N., Tuttle, M.S. and Wright, L.W., 2004. NCI Thesaurus: using science-based terminology to integrate cancer research results. In MEDINFO 2004, pp. 33-37.

[108] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S. and McClosky, D., 2014, June. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp. 55-60.

[109] Horridge, M. and Bechhofer, S., 2011. The owl api: A java api for owl ontologies. Semantic web, 2(1), pp.11-21.

[110] O'Connor, M.J., Halaschek-Wiener, C. and Musen, M.A., 2010. M2: A Language for Mapping Spreadsheets to OWL. In OWLED, 614.

[111] Bailey, R.W., 2004. The meaning of everything: the story of the Oxford english dictionary. Dictionaries: Journal of the Dictionary Society of North America, 25(1), pp.169-174.

[112] Chopra, S., Auli, M. and Rush, A.M., 2016, June. Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 93-98).

[113] Wang, L., Yao, J., Tao, Y., Zhong, L., Liu, W. and Du, Q., 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. arXiv preprint arXiv:1805.03616.

[114] Zhang, Y., Li, D., Wang, Y., Fang, Y. and Xiao, W., 2019. Abstract text summarization with a convolutional Seq2seq model. Applied Sciences, 9(8), pp.1665.

[115] Pasunuru, R. and Bansal, M., 2018. Multi-reward reinforced summarization with saliency and entailment. arXiv preprint arXiv:1804.06451.

[116] Shi, T., Keneshloo, Y., Ramakrishnan, N. and Reddy, C.K., 2021. Neural abstractive text summarization with sequence-to-sequence models. ACM Transactions on Data Science, 2(1), pp.1-37.

[117] Lin, J., Sun, X., Ma, S. and Su, Q., 2018. Global encoding for abstractive summarization. arXiv preprint arXiv:1805.03989.

[118] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

[119] Qi, W., Yan. Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R. and Zhou, M., 2020. ProphetNet: Predicting future n-gram for sequence-to-Sequence Pre-training, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, pp. 2401–2410.

[120] Zhang, J., Zhao, Y., Saleh, M. and Liu, P., 2020, November. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In International Conference on Machine Learning, pp. 11328-11339.

[121] Chen, J., Hu, Y., Liu, J., Xiao, Y. and Jiang, H., 2019, July. Deep short text classification with knowledge powered attention. In Proceedings of the AAAI Conference on Artificial Intelligence, 33 (1), pp. 6252-6259.

[122] Li, J., Huang, G., Chen, J. and Wang, Y., 2019. Short text understanding combining text conceptualization and transformer embedding. IEEE Access, 7, pp.122183-122191.

[123] Graff, D., 2003. English gigaword. In Linguistic Data Consortium, Philadelphia, PA.

[124] Mimno, D., Wallach, H., Talley, E., Leenders, M. and McCallum, A., 2011, July. Optimizing semantic coherence in topic models. In Proceedings of the 2011 conference on empirical methods in natural language processing, pp. 262-272.

[125] Lin, C.Y., 2004, July. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pp. 74-81.

[126] Kompoliti, K. and Verhagen, L., 2010. Encyclopedia of movement disorders. 1, Academic Press.

[127] Zhou, Q., Yang, N., Wei, F. and Zhou, M., 2017. Selective encoding for abstractive sentence summarization. in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 1, Association for Computational Linguistics, pp. 1095–1104.

[128] Cheng, J. and Lapata, M., 2016. Neural summarization by extracting sentences and words. in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 1, Association for Computational Linguistics, pp. 484–494.

[129] Cao, Z., Wei, F., Li, W. and Li, S., 2018, April. Faithful to the original: Fact aware neural abstractive summarization. In Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

[130] Chen, Y and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. ArXiv abs/1805.11080.

[131] Chen, Y.C. and Bansal, M., 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics. pp. 3290–3301.

[132] Lu, Y., Liu, L., Jiang, Z., Yang, M. and Goebel, R., 2019, July. A multi-task learning framework for abstractive text summarization. In Proceedings of the AAAI Conference on Artificial Intelligence, 33 (1), pp. 9987-9988.

[133] Dauphin, Y.N., Fan, A., Auli, M. and Grangier, D., 2017, July. Language modeling with gated convolutional networks. In International conference on machine learning, pp. 933-941.

[134] Xu, J., Gan, Z., Cheng, Y. and Liu, J., 2019. Discourse-aware neural extractive text summarization. arXiv preprint arXiv:1910.14142.

[135] Abdi, A., Hasan, S., Shamsuddin, S.M., Idris, N. and Piran, J., 2021. A hybrid deep learning architecture for opinion-oriented multi-document summarization based on multi-feature fusion. Knowledge-Based Systems, 213, pp.106658.

[136] Mutlu, B., Sezer, E.A. and Akcayol, M.A., 2019. Multi-document extractive text summarization: A comparative assessment on features. Knowledge-Based Systems, 183, pp.104848.

[137] Sanchez-Gomez, J.M., Vega-Rodríguez, M.A. and Perez, C.J., 2019. Comparison of automatic methods for reducing the Pareto front to a single solution applied to multi-document text summarization. Knowledge-Based Systems, 174, pp.123-136.

[138] Glauber, R. and Claro, D.B., 2018. A systematic mapping study on open information extraction. Expert Systems with Applications, 112, pp.372-387.

[139] Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.

[140] Griffiths, T., 2002. Gibbs sampling in the generative model of latent dirichlet allocation.

[141] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), pp.391-407.

[142] Li, W., Xiao, X., Lyu, Y. and Wang, Y., 2018. Improving neural abstractive document summarization with structural regularization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4078-4087.

[143] Song, S., Huang, H. and Ruan, T., 2019. Abstractive text summarization using LSTM-CNN based deep learning. Multimedia Tools and Applications, 78(1), pp.857-875.

[144] Suleiman, D. and Awajan, A., 2020. Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges. Mathematical Problems in Engineering.

[145] Luong, M.T., Pham, H. and Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

[146] Cohan, A., Dernoncourt, F., Kim, D.S., Bui, T., Kim, S., Chang, W. and Goharian, N., 2018. A discourse-aware attention model for abstractive summarization of long documents. arXiv preprint arXiv:1804.05685.

[147] Zhang, H., Xu, J. and Wang, J., 2019. Pretraining-based natural language generation for text summarization. arXiv preprint arXiv:1902.09243.

Appendix A.

Experiment Source Code

Here, we provide implementation details to build the 'KTSNR' system such as platform, app or package requirements to develop each model. We also provide some important the source code of our system 'KTSNR'. First, MMExample.java is the source code for the mapping algorithm to construct OMRKBS, NLP.py source code preprocesses the content to identify the topic information. Conceptualization.java and RunConceptualization.java are the source code for conceptualization algorithm. MultiDomainTask.java source code identifies the topic information in the text summarization model.

OMRKBS Development

Platform: JAVA Platform

Requirements

i) Dataset: DBpedia and ConceptNet dataset

ii) Install protégé to see the ontology structure

iii) OWL API and mapping parser package to import data in the ontology.

iv) NLTK tool: import stanford-corenlp-3.9.2.jar library

vi) Install MySQL: to import dataset into MySQL database and retrieve the information from database

vii) Import Jena lib to use SPRQL query to retrieve the OMRKBS

Topic Modeling and Conceptualization

Platform: Python and JAVA Platform

Requirements

i) Dataset: Gigaword and CNN/DailyMail dataset

ii) Python package: Nltk and Genism package to preprocess the document from the dataset

iii) Install Mallet package to convert the dataset into vocabulary and index of the document.

iv) MySQL: use to retrieve the concept from the database which we imported from ConceptNet

v) Import MySQL connector jar file in the project

vi) Use Genism lib for topic modeling.

Summarization model

Platform: Python environment and University GPU Cluster

i) NVIDIA GPU and NCCL

ii) Dataset: Gigaword and CNN/DailyMail dataset

iii) Anaconda, CUDA 10.1 and Cundnn above 7

iv) Numpy and Pytorch package to embed matrix and vector operation, reinforcement learning, and preprocessing.

v) Gensim package to handle the topic information.

vi) Fairseq Package: to implement the convolutional sequence network and attention mechanism.

MMExample.java

```
package org.mm.example;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
import org.mm.core.OWLAPIOntology;
import org.mm.core.OWLOntologySource;
import org.mm.core.TransformationRule;
import org.mm.core.settings.ReferenceSettings;
import org.mm.parser.ASTExpression;
import org.mm.parser.MappingMasterParser;
import org.mm.parser.ParseException;
import org.mm.parser.node.ExpressionNode;
import org.mm.parser.node.MMExpressionNode;
import org.mm.renderer.owlapi.OWLRenderer;
import org.mm.rendering.owlapi.OWLRendering;
import org.mm.ss.SpreadSheetDataSource;
import org.mm.ss.SpreadsheetLocation;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.util.Optional;
import java.util.Set;

import static org.mm.ss.SpreadSheetUtil.columnNumber2Name;

public class MMExample
{
  public static void main(String[] args)
  {
    try {
      File owlFile = new File("D:/test.owl");
      File spreadsheetFile = new File(

MMExample.class.getClassLoader().getResource("Actor.xlsx").getFile());

      // Create an OWL ontology using the OWLAPI
      OWLOntologyManager ontologyManager =
OWLManager.createOWLOntologyManager();
      OWLOntology ontology =
ontologyManager.loadOntologyFromOntologyDocument(owlFile);

      // Create a workbook using POI
      Workbook workbook = WorkbookFactory.create(spreadsheetFile);

      // Create an ontology source and a spreadsheet source (which wrap
an OWLAPI OWL ontology and a POI workbook, respectively)
      OWLOntologySource ontologySource = new OWLAPIOntology(ontology);
```

```
      SpreadSheetDataSource spreadsheetSource = new
SpreadSheetDataSource(workbook);

      // Create a Mapping Master expression. A Mapping Master
expression is rendered over a range of cells in a sheet.
      final String sheetName = "Actor";
      final Integer startColumnNumber = 1, finishColumnNumber = 1,
startRowNumber = 1, finishRowNumber = 3000;
      TransformationRule mmExpression = new
TransformationRule(sheetName, columnNumber2Name(startColumnNumber),
        columnNumber2Name(finishColumnNumber),
startRowNumber.toString(), finishRowNumber.toString(),
        "Creating actor instances", "Individual: @B* Types: Actor
Facts: hasPropertyValue
@C*(mm:prepend(\"activeYearsEndYear@\")),hasPropertyValue
@D*(mm:prepend(\"activeYearsStartYear@\")), hasPropertyValue @E*
(mm:prepend(\"alias@\")),hasPropertyValue
@F*(mm:prepend(\"almaMater@\")), hasPropertyValue
@H*(mm:prepend(\"associatedAct@\")),hasPropertyValue
@K*(mm:prepend(\"award@\")),hasPropertyValue
@L*(mm:prepend(\"birthDate@\")),hasPropertyValue
@M*(mm:prepend(\"birthName@\")),hasPropertyValue
@N*(mm:prepend(\"birthPlace@\")), hasPropertyValue
@P*(mm:prepend(\"birthYear@\")),hasPropertyValue
@Q*(mm:prepend(\"child@\")),hasPropertyValue
@S*(mm:prepend(\"citizenship@\")),hasPropertyValue
@U*(mm:prepend(\"country@\")),hasPropertyValue
@X*(mm:prepend(\"deathCause@\")),hasPropertyValue
@Y*(mm:prepend(\"deathDate@\")),hasPropertyValue @Z*(mm:prepend
(\"deathPlace@\")),hasPropertyValue @AB*(mm:prepend
(\"deathYear@\")),hasPropertyValue
@AC*(mm:prepend(\"education@\")),hasPropertyValue
@AE*(mm:prepend(\"employer@\")),hasPropertyValue @AG* (mm:prepend
(\"ethnicity@\")),hasPropertyValue
@AI*(mm:prepend(\"field@\")),hasPropertyValue
@AK*(mm:prepend(\"genre@\")),hasPropertyValue @AM*
(mm:prepend(\"height@\")),hasPropertyValue
@AN*(mm:prepend(\"hometown@\")),hasPropertyValue
@AP*(mm:prepend(\"imdbId@\")),hasPropertyValue
@AQ*(mm:prepend(\"individualisedGnd@\")),hasPropertyValue
@AR*(mm:prepend(\"influenced@\")),hasPropertyValue
@AT*(mm:prepend(\"influencedBy@\")),hasPropertyValue
@AV*(mm:prepend(\"instrument@\")),hasPropertyValue
@AX*(mm:prepend(\"knownFor@\")),hasPropertyValue @AZ*(mm:prepend
(\"movement@\")),hasPropertyValue
@BB*(mm:prepend(\"nationality@\")),hasPropertyValue @BD*(mm:prepend
(\"numberOfFilms@\")),hasPropertyValue
@BE*(mm:prepend(\"occupation@\")),hasPropertyValue @BG*
(mm:prepend(\"parent@\")),hasPropertyValue @BI*(mm:prepend
(\"partner@\")),hasPropertyValue
@BK*(mm:prepend(\"recordLabel@\")),hasPropertyValue
@BM*(mm:prepend(\"relation@\")),hasPropertyValue @BO* (mm:prepend
(\"relative@\")),hasPropertyValue
@BQ*(mm:prepend(\"religion@\")),hasPropertyValue
@BS*(mm:prepend(\"residence@\")),hasPropertyValue
@BU*(mm:prepend(\"restingPlace@\")),hasPropertyValue
@BX*(mm:prepend(\"soundRecording@\")),hasPropertyValue
```

```
@BY*(mm:prepend(\"spouse@\")),hasPropertyValue @CA*(mm:prepend
(\"stateOfOrigin@\"))");


      // Create a Mapping Master parser for the expression, parse it,
and return an AST node representing the expression
      MappingMasterParser parser = new MappingMasterParser(
        new
ByteArrayInputStream(mmExpression.getRuleString().getBytes()), new
ReferenceSettings(), -1);
      MMExpressionNode mmExpressionNode = new
ExpressionNode((ASTExpression)parser.expression()).getMMExpressionNode(
);

      // Create an OWL renderer and supply it with an ontology and a
spreadsheet. An OWL renderer renders a set of OWLAPI-based OWL axioms.
      OWLRenderer owlRenderer = new OWLRenderer(ontologySource,
spreadsheetSource);

      // Loop through the cells specified by the Mapping Master
expression
      for (int columnNumber = startColumnNumber; columnNumber <=
finishColumnNumber; columnNumber++) {
        for (int rowNumber = startRowNumber; rowNumber <=
finishRowNumber; rowNumber++) {
          // A Mapping Master expression is rendered in the context of
a location in a spreadsheet
          spreadsheetSource.setCurrentLocation(new
SpreadsheetLocation(sheetName, columnNumber, rowNumber));

          // Render the Mapping Master expression as an OWL rendering
(which will contain a set of OWLAPI-based OWL axioms)
          Optional<OWLRendering> owlRendering =
owlRenderer.render(mmExpressionNode);

          // Display the OWL axioms rendered by the Mapping Master
expression
          if (owlRendering.isPresent())
            System.out.println("Rendered OWL axioms: " +
owlRendering.get().getOWLAxioms());
          Set<OWLAxiom> renderedOWLAxioms =
owlRendering.get().getOWLAxioms();
          ontologyManager.addAxioms(ontology, renderedOWLAxioms);
        //

        }
      }
  //     storer.storeOntology(ontology, new FileDocumentTarget(new
File("D:/news.owl")), new FunctionalSyntaxDocumentFormat());
        ontologyManager.saveOntology(ontology);

    } catch (OWLOntologyCreationException | RuntimeException |
ParseException | InvalidFormatException | IOException e) {
      System.err.println("Exception: " + e.getMessage());
      e.printStackTrace();
      System.exit(-1);
    }
```

```
    catch (OWLOntologyStorageException e) {
         System.err.println("Exception: " + e.getMessage());
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
  }
}
```

NLP.python

```
from __future__ import print_function
import re
import numpy as np
# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer,PorterStemmer
from nltk.corpus import stopwords
import re
import numpy as np
import pandas as pd
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS

np.random.seed(400)
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()

processed_docs=[]
stem_words=[]

np.random.seed(400)
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.ERROR)

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)

from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu',
'use','like','think','tell', 'article','sure', 'said', 'know','call'])

print(stop_words)

with open('D:/topic_model/data/word.vocab', 'r') as myfile:
     data = myfile.read()
     #data = list(df.split("\n"))
```

```
stem_words=[]
#pprint(data[:1])

def sent_to_words(sentence):
        #print(sentence)
        sentence=str(sentence)
        sentence = sentence.lower()
        sentence=sentence.replace('{html}',"")
        cleanr = re.compile('<.*?>')
        cleantext = re.sub(cleanr, '', sentence)
        #print(cleantext)
        rem_url=re.sub(r'http\S+', '',cleantext)
        rem_num = re.sub('[0-9]+', '', rem_url)
        tokenizer = RegexpTokenizer(r'\w+')
        tokens = tokenizer.tokenize(rem_num)

        return tokens;
data_words = sent_to_words(data)
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100)
# higher threshold fewer phrases.

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)


# See trigram example
#print(trigram_mod[bigram_mod[data_words[0]]])

# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not
in stop_words] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB',
'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        for token in doc:
         if token.pos_ in allowed_postags and len(token.lemma_)>3 and
token.lemma_ not in gensim.parsing.preprocessing.STOPWORDS and
token.lemma_ not in texts_out and token.lemma_ not in stop_words :
          texts_out.append(token.lemma_)
          #print(token.lemma_)
        #texts_out.append([token.lemma_ for token in doc if token.pos_
in allowed_postags])
    return texts_out

# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)
```

```
# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)
#print(data_words_bigrams)

# Initialize spacy 'en' model, keeping only tagger component (for
efficiency)
# python3 -m spacy download en
nlp = spacy.load('en_core_web_sm')

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams,
allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
#print(data_lemmatized)

# Create Dictionary
#id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
#texts = data_lemmatized
i=0
with open('D:/topic_model/data/inputs.vocab', 'w') as f:
    for item in data_lemmatized:
        print(item)
        #print(i)
        f.write("%d:%s\n" % (i, item))
        i=i+1

myfile.close()
f.close()
```

Conceptualization.java

```
package conceptualization;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import prior.Prior;
import util.Corpus;
import util.Probase;
import util.ReadWriteFile;

public class Conceptualization {

    public static Map<String, Map<String, Double>> conceptualization(
            String domain) throws IOException, SQLException {

        List<String> vocab =
Corpus.getVocab("/home/shirin/topic_model/data//" + domain
                + ".vocab");

        int[][] docs =
Corpus.getDocuments("/home/shirin/topic_model/data//" + domain
                + ".docs");

        int line = 0;

        StringBuilder sb = new StringBuilder();

        Connection conn = Probase.getConnectionMySql();

        Map<String, Map<String, Double>> entity_concept_rep = new
HashMap<>();

        for (int[] doc : docs) {

            List<String> entities = new ArrayList<>();

            for (int word : doc) {
                entities.add(vocab.get(word));
            }

            String conceptual = Prior.getConceptualiztion(entities,
                    entity_concept_rep, conn);
```

```
            line++;
            System.out.println(line + "\t" + conceptual);
            sb.append(line + "\t" + conceptual + "\n");

        }
        conn.close();

        String filename = "file//Conceptualization_" + domain
            ".txt";

        ReadWriteFile.writeFile(filename, sb.toString());

        return entity_concept_rep;

    }
public static void conceptualizations(Connection conn) throws
IOException, SQLException {

        List<String> entities = new ArrayList<>();
            File f = new
File("/home/shirin/topic_model/data/word.vocab");
                BufferedReader reader = new BufferedReader(new
InputStreamReader(
                                        new FileInputStream(f), "UTF-
8"));
                    String line = "";
                        while ((line = reader.readLine()) != null) {
                                    entities.add(line);
                                        }
                                Prior.getConceptualiztions(entities, conn);
                                    conn.close();
                                        reader.close();
}


public static Set<String> getConceptualizationSet(String domain)
            throws IOException {

        String filename = "file//Conceptualization_" + domain
                + "_NaiveBayes_0.8.txt";
        File f = new File(filename);
        BufferedReader reader = new BufferedReader(new
InputStreamReader(
                new FileInputStream(f), "UTF-8"));
        String line = "";

        Set<String> concept_set = new HashSet<>();

        while ((line = reader.readLine()) != null) {

            String[] temp = line.trim().split("\t");

            for (int i = 0; i < temp.length; i++) {
                if (i != 0 && i < 4)
                    concept_set.add(temp[i]);
            }
```

```
            }

            reader.close();

            return concept_set;
        }

    }
```

RunConceptualization.java

```
package test;

import java.sql.Connection;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import net.sf.javaml.core.Dataset;
import prior.Prior;
import util.Probase;
import util.ReadWriteFile;
import conceptcluster.Kmedoids;
import conceptualization.Conceptualization;

public class RunConceptualization {

    public static void main(String[] args) throws Exception {

        List<String> entities = new ArrayList<String>();

        entities.add("bayes");

        entities.add("svm");

        Connection conn = Probase.getConnectionMySql(); // connnect
with mYSQL Probase databse

        Map<String, Map<String, Double>> entity_concept_rep = new
HashMap<>();

    String domain = "input";

        // Mixture
        Conceptualization.conceptualization(domain);

        Set<String> concept_set = Conceptualization
                .getConceptualizationSet(domain);

        List<String> concepts = new ArrayList<>(concept_set);

        Dataset data =
Prior.getConceptEntityRepSparseDataSet(concepts);

        int[] assignment = Kmedoids.RunKmedoidsCosine(data, 15);

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < assignment.length; i++) {
            System.out.println(concepts.get(i) + ":" + assignment[i] +
"\n");
            sb.append(concepts.get(i) + ":" + assignment[i] + "\n");
        }
```

```
        String filename = "file//concept_cluster.txt";

        ReadWriteFile.writeFile(filename, sb.toString());

    }
}
```

MultiDomainTask.java

```
package test;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import prior.Prior;
import topic.LDA;
import topic.PriorLDABurnLag;
import util.Common;
import util.Corpus;
import util.ReadWriteFile;

public class MultiDomainTask {

    public static void main(String[] args) throws Exception {


        File[] files = new
File("/home/shirin/topic_model/data//input//").listFiles();

        List<String> domain_list = new ArrayList<String>();

        for (File f : files) {
            System.out.println(f);
            String file_path = f.toString();
            String domain = file_path.substring(file_path.indexOf("\\")
+ 1,
                    file_path.length());

            domain_list.add(domain);
        }

        double coherence = 0;

        StringBuilder sb = new StringBuilder();

        for (String domain : domain_list) {

            double domain_coherence = runPriorLDA(domain);
            coherence += domain_coherence;
            sb.append(domain + "\t" + domain_coherence + "\n");
            System.out.println(domain + "\t" + domain_coherence +
"\n");
        }

        sb.append("average : " + coherence / domain_list.size() +
"\n");

        String filename = "output//topic//concept.txt";

        ReadWriteFile.writeFile(filename, sb.toString());
```

```
    }


public static double runPriorTopic(String domain) throws Exception {

        List<String> vocab =
Corpus.getVocab("/home/shirin/topic_model/data/input.vocab");

        Map<String, Map<String, Double>> vocab_concept_map = Prior
                .getVocabConceptMap(vocab);

    int K = 15;
        //stem.out.println(vocab_concept_map);

        String filename = "output//topic//concept_cluster.txt";

        // ReadWriteFile.writeFile(filename, sb.toString());

        Map<String, Integer> concept_cluster = ReadWriteFile
                .getConceptCluster(filename);
        System.out.println("Concept Cluster"+concept_cluster);

        int V = vocab.size();

        double[][] beta = Prior.getAsymmetricBeta(K, vocab,
vocab_concept_map,
                concept_cluster);

        int[][] docs =
Corpus.getDocuments("/home/shirin/topic_model/data/input.docs");
        System.out.println(docs);

        double[][] alpha = Prior.getAsymmetricAlpha(docs, beta);


        int iterations = 2000;

        int top_word_count = 30;

        PriorLDABurnLag plda = new PriorLDABurnLag(docs, V);

        plda.markovChain(K, alpha, beta, iterations);

        double[][] phi = plda.estimatePhi();

        double[][] phi_copy = Common.makeCopy(phi);


        double[][] phi_for_write = Common.makeCopy(phi);

        StringBuilder sb = new StringBuilder();

        for (double[] phi_t : phi_for_write) {

            for (int i = 0; i < 10; i++) {
```

```
                int max_index = Common.maxIndex(phi_t);

                sb.append(vocab.get(max_index) + "\t");

                phi_t[max_index] = 0;

            }
            sb.append("\n");

        }

        filename = "file//input.txt";


        double average_coherence = Corpus.average_coherence(docs,
phi_copy,
                top_word_count);


        sb.append("average coherence\t" + average_coherence);

         ReadWriteFile.writeFile(filename, sb.toString());

        double[][] theta = plda.estimateTheta();

        // perplexity
        double perplexity = Corpus.perplexity(theta, phi, docs);
        System.out.println("perplexity : " + perplexity);

        return average_coherence;

    }
     */
    public static double runTopicM(String domain) throws IOException {

        List<String> vocab = Corpus.getVocab("data//" + domain + "//" +
domain
                + ".vocab");

        int[][] docs = Corpus.getDocuments("data//" + domain + "//" +
domain
                + ".docs");

        int K = 15;
        double alpha = 1;
        double beta = 0.1;
        int iterations = 2000;

        int top_word_count = 30;

        LDA lda = new LDA(docs, vocab.size());

        lda.markovChain(K, alpha, beta, iterations);

        double[][] phi = lda.estimatePhi();

        double[][] phi_copy = Common.makeCopy(phi);
```

```
        double average_coherence = Corpus.average_coherence(docs,
phi_copy,
                top_word_count);

        System.out.println("average coherence : " + average_coherence);

        double[][] theta = lda.estimateTheta();

        // perplexity
        double perplexity = Corpus.perplexity(theta, phi, docs);
        System.out.println("perplexity : " + perplexity);

        return average_coherence;
    }

}
```