TECHNISCHE UNIVERSITÄT DRESDEN

Computer Science European Masters in Computational Logic, Institute for Artificial Intelligence

Chair of Computational Logic

Topic Detection and Labeling for Online Meetings and Webinars

Genet Asefa Gesese

Born on: 23 December 1988
Matriculation number: 4567749
Matriculation year: 2015

Master Thesis

to achieve the academic degree

Master of Science (M.Sc.)

Supervisor

Dr. Daniel Esser (LogMeIn)

Dr. Thomas Springer

Supervising professor

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Submitted on: 06.06.2018

Task Description

Declaration

Author	:	Genet Asefa Gesese
Matrikel-Nr	:	4567749
Title	:	Topic Detection and Labeling for Online Meetings and Webinars
Degree	:	Master of Science
Date of Submission	:	06 June 2018

I hereby certify that I have authored this thesis entitled *Topic Detection and Labeling for Online Meetings and Webinars* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. They were no additional persons involved in the spiritual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, Germany
June, 2018

Genet Asefa Gesese

Abstract

A huge amount of webinars are held online for various purposes, for instance, education, entertainment, and business. Besides webinars, meetings are being held online in different sectors and companies. It is common to find more than one webinar or meeting discussing the same topic. On the other hand, multiple topics could be discussed in a single webinar or meeting. This implies if users are interested in finding relevant formation on a certain topic, then they may have to go through each and every webinar or meeting content to identify the topic. Performing this manually is highly inadvisable as it is time consuming and inefficient. Therefore, it is desirable to have an automated system that efficiently detects the topics discussed in webinars or meetings. Given this fact, in this research, an effort has been made to build a Topic Detection and Labeling (TDL) system for online meetings and webinars.

The proposed TDL system is designed and implemented with the purpose of solving the two main research questions associated with this study. The first question is how to customize a clustering algorithm for single-document topic detection and find a way to automatically determine the optimal number of topics without user involvement. The other is how to use a KB to give semantic labels to identified topics. The TDL system addresses the first research question by using an agglomerative clustering technique with a variant of the elbow algorithm. The distance function used in the clustering algorithm is defined by combining an euclidean distance with a newly defined distance function named $in\_transcript$ distance which determines the similarity between sentences by their position in the transcript. The elbow algorithm is used as a technique to determine optimal number of topics. The second research question is solved by integrating an external knowledge base, i.e. DBpedia, with the system to help in identifying relevant labels for topics. The proposed system is composed of Topic Detection (TD) and Topic Labeling (TL) core components.

The evaluation of the TD component proved that combining euclidean distance function with $in\_transcript$ distance function results in better performance than using any of these functions alone. In addition, the TD system showed slightly better quality and speed when it is built with Word2Vec word embedding model than with Glove. In the case of TL, the evaluation showed that all the four properties used for ranking of candidate labels, namely, popularity, term specificity, topic specificity, and coherence, are required to gain better performance. Each of these properties play a vital role in determining the importance of a candidate label to a topic.

Key Words: Topic detection, Topic labeling, Topic modeling, Ontology, DBpedia, Webinars, Clustering, Hierarchical agglomerative clustering, Word embedding, WordToVec, Glove, Text similarity.

Contents

List of Figures

List of Tables

[1] http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/
[2] https://nlp.stanford.edu/projects/glove/

Acronyms

AI Artificial Intelligence. 1, 23, 67

B2B Business to Business. 1

BD Boundary Detection. 1, 34

CAC Customized Agglomerative Clustering. vi, 1, 43, 44, 46–48, 52–54, 56, 68, 98

CBOW Continuous Bag of words. 1, 12

CL Cluster Linkage. 1, 43, 44, 48, 51, 52, 56, 67, 68, 72

DL Deep Learning. 1, 10, 12, 15

DONT Determining the Optimal Number of Topics. vi, 1, 43, 44, 53, 55, 57, 67, 68, 76, 91, 98, 100

Glove Global Vectors for Word Representation. iii, 1, 12–14, 46, 78–82, 86, 94, 98

IR Information Retrieval. 1, 2, 6, 16, 85

KB Knowledge Base. iii, 1, 3, 5, 6, 34, 36, 40

LDA Latent Dirichlet Allocation. 1, 3, 27, 34, 35, 37–39, 58

LG Label Generation. vi, 1, 43, 59, 60, 68, 87, 98

LR Label Ranking. vi, 1, 43, 59, 60, 69, 86–89, 92, 98

ML Machine Learning. 1, 6, 10, 15, 16, 22, 33

NLP Natural Language Processing. 1, 7, 10, 12, 16, 33, 38

RDF Resource Description Framework. 1, 23, 28

SA Sentiment Analysis. 1, 7

SE Sentence Embedding. vi, 1, 43–46, 50, 56, 67, 68, 98

SS Sentence Similarity. 1, 43, 44, 48–50, 52, 56, 67, 68

TD Topic Detection. iii, vi, ix, xi, 1–6, 8, 9, 16, 17, 22–25, 31, 33–37, 39–41, 43–49, 51, 53, 55, 57, 62, 63, 65, 67–69, 72, 73, 75, 76, 78–87, 89, 91–94, 97, 98

TDL Topic Detection and Labeling. iii, v, vi, ix, xi, 1, 3–9, 24, 26, 28, 30–33, 39, 41, 43, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64–67, 69, 70, 72, 73, 90, 94, 95, 97, 99

TDT Topic Detection and Tracking. 1, 33, 34

TF-IDF Term Frequency/Inverse Document Frequency. 1, 11, 12, 15, 35, 45, 61, 89, 98, 99

TL Topic Labeling. iii, vi, ix, xi, 1–6, 9, 25, 30, 31, 33, 37–41, 43, 58, 59, 61–63, 65, 66, 68, 69, 72–75, 85, 86, 89, 90, 92, 93, 95, 97, 98

VSM Vector Space Model. 1, 14, 44

1. Introduction

1.1. Overview

Different medias, such as text, audio and video are used to share information or knowledge over the internet. Nowadays, video and audio streams have become popular multimedia as they enable people to present content in different forms to their audiences. One way of sharing information using those medias is through a webinar (i.e. web-based seminar) which is a term used to describe seminars held over the web and it can also be referred to as webcasts, online trainings/classes/events. Webinars can be either a paid or free presentations, demonstration, discussion or any other form of sessions. In various sectors, people make use of different applications to have either online meetings and webinars or to distribute pre-recorded videos. For instance, GoToMeeting[1] and GoToWebinar[2] are applications which are used to let people host online events.

According to the Business to Business (B2B) Content Marketing Research of 2017 by the Content Marketing Institute and marketingProfs[3], 58% of B2B marketers use webinars as a content marketing tactic. As a result of this, there is an enormous amount of online videos and webinars available on the internet. It has been challenging to search for webinars hosted in specific domains or topics as there are webinars almost in any field. Besides, searching for those webinars and meetings manually is time consuming and inefficient. Therefore, in order to access the relevant information enclosed in webinars and meetings it is desirable to have an automated searching system.

In addition to the fact that users intend to use text to search for video and audio files, in the field of data analysis and data science it is quite effective and easier to work

[1] https://www.gotomeeting.com
[2] https://www.gotowebinar.com
[3] http://contentmarketinginstitute.com/wp-content/uploads/2016/09/2017_b2b_Research_FINAL.pdf

directly with text data instead of using the actual audio or video file. Hence, various automatic speech recognition systems are available to create transcriptions of video and audio files. The transcriptions can be used as input to different text-based data analysis systems to get relevant information discussed in a webinar.

Various sectors or companies which provide webinars intend to reach as many users as possible either online or by sharing pre-recorded webinars and users watch these recorded webinars and meetings if they want to catch up on an event they couldn't attend online or to find a set of webinars discussing related topics. Hence, besides just providing an automated system, it would be more desirable to make the system a semantic-based text analysis system that identifies the topics discussed in a webinar based on its semantics so that users can easily find the exact webinars they are looking for. This will make both sides (i.e. the webinar providers and users) beneficial time and money wise.

1.2. Problem Analysis

As discussed above, in order to address the problem of accessing relevant information in a huge set of webinars covering multiple domains or fields, it is important to have an automated system that identifies the topics efficiently. There are various techniques available to process natural language text. Among the most commons are Information Retrieval (IR), classification, clustering, summarization, sentiment analysis, Topic Detection (TD), and Topic Labeling (TL). Even though all these methods are designed to accomplish different tasks, they have one thing in common which is capturing information out of a text corpus easily and effectively. TD, one of the text analytics techniques, is a process of identifying topics discussed in a text object [1]. TL is a process of assigning representative terms as labels to identified topics that well interpret the meaning of the topics.

In addition to providing the result of the TD system to users so that they can see the list of topics covered in a specific webinar, the result can also be integrated with other text analysis techniques to improve their performance. For instance, if we take a scenario where users try to retrieve information out of a transcript of a webinar by using a keyword-based IR system, they may miss relevant documents if they don't contain the exact search keywords. However, if we integrate the result of a TD system with the IR system then there will be higher probability of returning documents that don't share exact keywords but are semantically similar. The result of topic modeling can also be used for text summarization [2].

Moreover, since multiple topics can be discussed in a single webinar or meeting, to find a specific topic in a webinar users have to scan the video to locate the exact position. However, this would be easier if we could use a topic detector to point to the exact location in the video by using topic boundaries and their labels.

In order to build such TD systems, numerous approaches have been proposed so far. The most common ones are probabilistic topic models like Latent Dirichlet Allocation (LDA) [3]. However, there are still a lot of research gaps in this area as there is a lot to be improved. One of the main challenges in the process of TD is to find the optimal number of topics. Most of the existing methods let the user decide the number of topics they want to get out of the transcripts and some try to use some methods to determine the number of topics automatically but the result is not always accurate. The other important issue with TD is to determine the boundaries of topics in a document/corpora. Since most TD systems as in LDA [3] use bag of words which don't consider the order of words/sentences, it is quite difficult to locate where a specific topic starts and ends in a document.

Regarding TL, it is recently becoming an active research topic with some research works which propose different approaches [4, 5, 6, 7, 8]. However, there is a lot to be improved with these methods as each of them has their own drawbacks. In addition, these labeling algorithms may not work well with the TD approach proposed in this research. Therefore, it is appropriate to design and implement a TL component which finds labels for the topics generated with the TD component proposed in this thesis work.

To sum up, the motivation behind this thesis work is to propose a TDL system for online webinars and meetings by taking into account the semantic-similarity among sentences of a transcript with the intention of addressing the issues with the existing TDL systems. Figure 1.1 depicts an example scenario how the proposed system will behave against a text document input. In this example, three topics (with their corresponding boundaries) are detected and for each one of them a possible list of candidate label terms are generated.

1.3. Research Questions

The main goal of this thesis work is to design and implement a **TDL System for online transcripts of meetings and webinars** by integrating machine learning (i.e. clustering) algorithms with external KB. Besides evaluating the system with a test corpus, as optional, a graphical prototype will be developed which will be used for visualizing topics generated from meetings and webinars.

Thus the key research questions that will be addressed in this thesis work are stated as follows:

1. How to use sentence clustering algorithm for single-document topic detection?
    a) How to adapt an existing clustering algorithm in order to divide a document into segments or topics ?
    b) How to determine the optimal number of topics?
2. How to use a Knowledge-base to give semantic labels to identified topics?

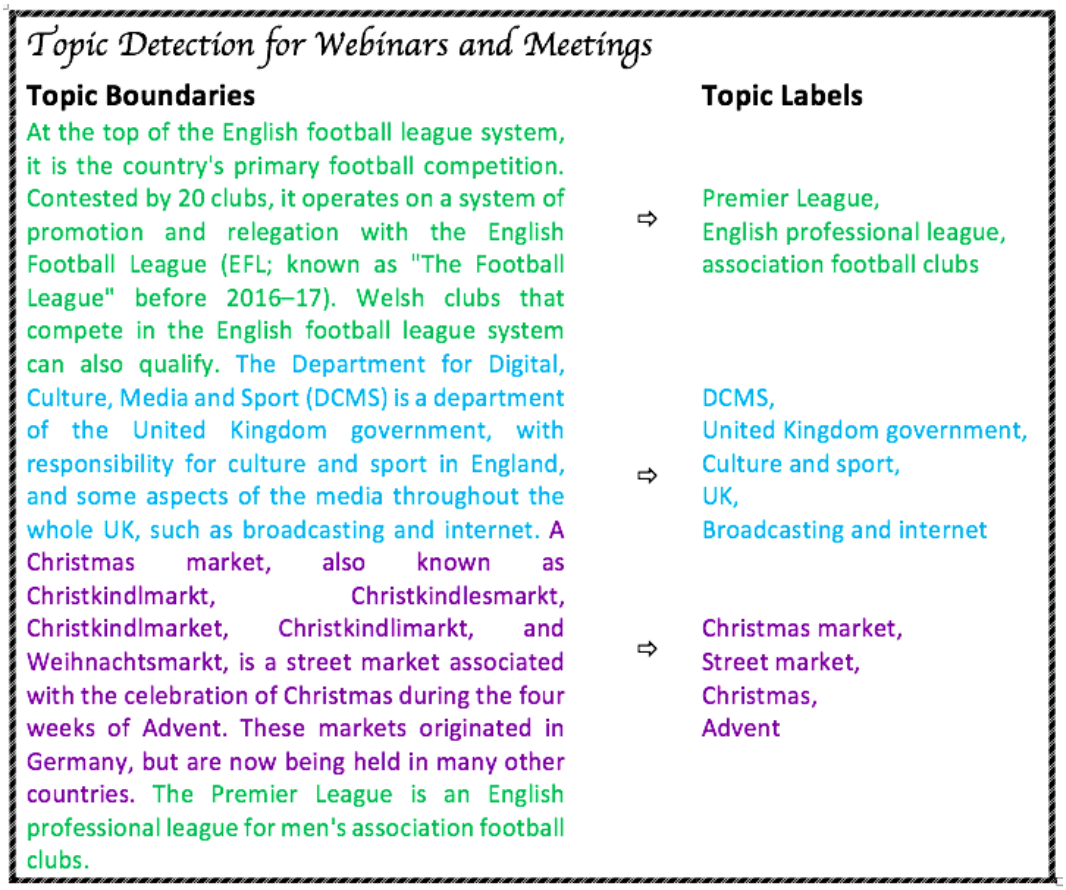


Figure 1.1.: An example scenario to show how the proposed system is intended to work

a) How to generate candidate terms to represent an identified topic?
b) How to assign weight to each candidate term used as a label to a topic in order to rank them based on importance?

1.4. General and Specific Objectives

The overall objective of this master's thesis research is to propose semantic-based automatic TDL system for webinars and meetings. In order to accomplish the aforementioned general objective, the following specific objectives are devised.

- Asses the capabilities of the existing TD and TL systems
- Adapt an appropriate clustering method that can be used to cluster sentences of transcripts
- Identify the appropriate method used in clustering for determining the number of

optimal clusters which can be used in this research for accurately identifying the right number of topics in a transcript.

- Choose the appropriate external knowledge base which can be integrated with the TDL system to support the design and implementation of the TL component
- Propose a method which can generate candidate labels for identified topics and rank them according to their importance to their respective topic
- Build the actual TDL system by integrating the machine learning (clustering) algorithm with the chosen external KB
- Gather or prepare test data which can be used for evaluating the performance of the system
- Choose an appropriate system measuring metric which can be used for measuring the value of the proposed approach
- Test the implemented TDL system against the prepared test data and measure its relevance using the chosen metrics and compare its result to existing approaches

1.5. Scope of the Study

This master's thesis research has been conducted to detect topics discussed in webinars or meetings using their transcripts. The process of generating transcripts of webinars is outside the scope of the research. Rather, already available transcripts are used as input to test the proposed system. The TDL process is designed to work only for one document at a time, i.e. it is a single-document topic detection. Therefore, multi-document topic detection is not part of the scope of the thesis. Moreover, creating correlation among identified topics of a document is not considered in this master's thesis.

1.6. Methodology

In favor of meeting the outlined general and specific objectives of this research, different methodologies have been applied.

- **Literature Review**: Literature reviews will be conducted to acquire enough understanding of the components of the TDL system. Specifically, literatures in the area of automatic TD, TL, linked data sets or ontologies, word embedding, and clustering. Reading literatures and related works helps to gain the required knowledge on a certain subject and also assists in identifying the right methods and tools for implementing the different components of the system. Hence, different research papers, books, and web sites will be reviewed.

- **Data collection**: Two sets of data are required for this study, a set of webinar transcripts for conducting the experiment for the whole system and KB for testing the TL component. Since it is easier to find transcripts of TED talks[4], the data source for this study will be the transcripts from TED talk videos.

- **Implementation Method**: In order to accomplish the objectives of the research, different methods and tools will be engaged in the implementation of the system. Python is known for its extensive libraries which provide support for different data analysis tasks. This led Python to become a widely used programming language to develop Machine Learning (ML) systems in the field of data science or data analytics. Due to the reason that clustering, one of ML algorithms, is used in this research, the implementation of the proposed system will be written using Python. The common libraries like Scipy[5], Numpy[6] and Scikit-learn[7] will also be used to facilitate some of the tasks in the development process.

  For the sake of showing the prototype of the system in a user friendly way, a website will be developed using Flask. The SPARQL (RDF query language) is used to access the concepts in the DBpedia data set.

- **Testing and Evaluation**: To evaluate the performance of the proposed TDL approach, the two core components of the system, namely TD and TL, will be tested separately using different data sets. Finally, the whole system will be assessed based on the research questions and the derived requirements of the system. For evaluating the TD component, v_measure and purity metrics will be used and for the TL component precision with human judgment will be applied.

1.7. Application of the Study

Besides being a research to fulfill the requirement of the Master's program, the result of this study is believed to be used either as an input for further researches or can be put into use in different fields. The possible areas where the developed TDL system can be used are:

- **IR**: TDL systems can play a vital role when they are integrated with text search engines. For instance, the proposed system can be used to facilitate the process of querying transcripts of webinars or any kind of text documents.

- **Webinar Classification**: Extracting the existing topics and determining the accurate number of topics in a document helps in increasing the quality of webinar classifier algorithms (or text document classifier algorithms in general).

[4] `https://www.ted.com`
[5] `https://www.scipy.org`
[6] `http://www.numpy.org`
[7] `http://scikit-learn.org/stable/`

- **Webinar Recommendation**: Identifying the topics discussed in webinars and representing the webinar with there respective topics (or labeled topics) will improve the capability of webinar recommendation systems in recommending the appropriate webinars to users.
- **Sentiment Analysis (SA)**: As a research [9] indicates, it is beneficial to first detect topics discussed in documents and then apply each tasks of the sentiment analysis over the topics instead of directly running the SA algorithm to the documents.
- **Document Clustering**: TDL systems can bring advantages in supporting quality clustering as they do in classification. Most importantly, running clustering algorithms over the representative labels of topics (clusters) is much better than running it on the original set of documents because of resource requirement issues.
- **Document Visualization**: In visualizing large volume text corpora, having a high level summary of documents is required and TDL system can be used in the process of generating such summaries.
- **Text Summarization**: Having the topics of a document identified helps in determining the content of the summary of the document. Specifically, since the proposed approach in this thesis work is designed in a way it can detect boundaries of topics as well, it will be easier to build a text summarizer system up on the TDL system. For further illustration on this matter, for instance by selecting top relevant sentences from each topic and appropriately merging them, a summary for the entire document can be made.
- **Linguistic Understanding**: The proposed TDL system can also be explored in the area of cognitive science for supporting the process of understanding a particual natural language. For instance, some research works have been undertaken to understand word correlations in a set of documents written in a certain language [10].
- **Qualitative Analysis**: Qualitative studies such as media or social-network analysis, opinion mining, and sociological researches can get the benefit of the result of this thesis work to capture the topics discussed in documents.
- **Multilingual modeling**: Since the proposed approach is language independent, i.e. works with transcripts written in any language with a little or no modification, it can be integrated in any of its application areas regardless of the language considered.
- **Further Research**: In addition to directly being used in different Natural Language Processing (NLP), data mining or machine learning tools, the research result can also lead to further research ideas in the same or closely related topics or domains.

1.8. Outline of the Thesis

The current chapter was designed to give an overall introduction to the master's thesis topic with the purpose of enabling the readers capture the core points that initiated the work and also to state the research questions that are expected to be answered as the result of the research work. The rest of the chapters in this thesis report are organized as follows:

Chapter 2 - Literature Review: This chapter presents the extensive literature reviews conducted to get enough insights on the master's thesis topic.

Chapter 3 - Requirement Analysis: The assumptions taken, the use case, functional and non-functional requirements driven for the achieving the objective of the study are discussed in this chapter.

Chapter 4 - Related Work: In this chapter, the approaches, contributions, and limitations of the research works conducted by various researchers in the area of TD so far will be discussed.

Chapter 5 - Design of the Proposed TDL System: In this part, the general framework, concept and design of the proposed approach will be discussed in detail.

Chapter 6 - Implementation: The methods and choices taken for the implementation of the designed algorithm will be presented in detail.

Chapter 7 - Experiment and Evaluation: We will discuss the experiment procedure followed and the available evaluation metrics used to evaluate the proposed approach and then present the result of the evaluation.

Chapter 8 - Conclusion and Future Work: In this chapter, the conclusions drawn from the research, the limitations and contributions of the study, and the directions for further possible future works related to this thesis topic are presented.

2. Literature Review

In this chapter, extensive review of literature on the general concepts which are somehow involved in the design and implementation of the proposed Topic Detection and Labeling (TDL) system is presented. The review covers the concepts such as techniques for generating word embedding, higher-level text (i.e. phrase, sentence, and document) embedding, text similarity measures, clustering, and ontologies. In the proposed system, higher level text embedding is required to represent sentences in a vectorized form and an appropriate text similarity measure is used to compute similarity between sentences for generating clusters or topics. DBpedia is used in the design and development of the Topic Labeling (TL) component. Therefore, this literature review will provide the information that readers require to understand the concept of the problem that is being addressed in the realm of this thesis work and to easily follow the topics that are discussed in the rest of the chapters. To this end, the conducted review will be discussed in the up coming subsections with a summary at the end to give an over all conclusion on points addressed in this chapter.

2.1. Text Similarity

In order to answer the first research question formulated in this thesis work, which is exploring the advantage of sentence clustering for Topic Detection (TD), it is found to be valuable to conduct a review on available text embedding and similarity measures given the fact that most clustering techniques are designed to be effective when they are applied to numerical data sets. Therefore, in this section, the review of word embedding algorithms and higher level text embedding techniques, with focus on unsupervised ones, will be presented. Moreover, the text similarity techniques which rely on text embeddings will be discussed.

2.1.1. Word Embedding

Word embedding is an arrangement of dense real-valued vectors representing the syntactic and semantic information of words along with their context so as to enable computers handle large volume text data. A word embedding for a text is a learned representation where words occurring in the text with the same meaning are given a similar representation. Relations between words can be derived by training word embedding models. Moreover, word embeddings are the base for deep learning algorithms to work with challenging NLP problems. Most widely known word embedding models learn word vector representations using very large corpora. However, it is also possible to train word embeddings on knowledge graphs.

Why Word Embedding

Most ML algorithms and almost all Deep Learning (DL) architectures are not designed to process plain text in their raw form rather to work with numeric data. It is challenging to extract information out of a big data which is encoded in text format. Traditionally, NLP systems treat words as discrete atomic symbols, for instance, the word "computer" and "machine" can be encoded using some arbitrary numbers "537" and "143" respectively regardless of their semantic similarity. In order to address this issue of processing large amount of text data without losing the natural semantics of the text in the data, various word embedding techniques have been proposed so far. These word embedding techniques are designed in such a fashion that they can handle huge text data by mapping a word using a dictionary to a vector.

Different Types of Word Embeddings

Different experts of NLP, ML, and lately DL have conducted various research in the area of word embedding so far and presented their approach with the obtained result. In this section, the overview of the well known approaches will be presented. We will start with discussing the simplest vectorization method called one-hot and continue with the rest by dividing them into two categories as frequency-based and prediction-based word embeddings.

One-hot Vectorization: The simplest embedding method is to represent a word using a one-hot vector composed of 0s and 1s where 1 indicates the occurrence of the word and 0 means otherwise. One-hot representation is a way of encoding categorical data which has finite set of label data. Categorical data are variables that contain label values rather than numeric values. For example, given the sentence "Python for data science", the dictionary can be ["Python", "for", "data", "science"] and the one-hot representation of the word "data" is [0, 0, 1, 0] whereas that of "science" is [0, 0, 0, 1]. One of the main drawbacks of one-hot embedding is that thousands or millions of dimensions are required when dealing with big data as it follows sparse word representation. The other crucial issue is that models

based on this embedding technique are not capable to handle words that do not occur in the training data [11].

Frequency based Embedding: Here, the common word embedding methods such as count vector, Term Frequency/Inverse Document Frequency (TF-IDF), and co-occurence vector which are based on frequency of tokens or words are discussed.

- **Count Vector:** It is a process of creating a matrix M of K X N for a corpus of documents where K is the total number of documents and N is the total number of unique tokens in the corpus. A value in the matrix M at the position (i, j) where $i \in (1, K)$ and $j \in (1, N)$ represents the frequency of the token at column j in the document at row i. A row can be referred to as document vector and a column as word vector. Instead of frequency, it is also possible only to consider the presence of a token while constructing the matrix. Moreover, for the sake of shrinking the size of the vocabulary, only the most frequent terms in the corpus are considered. For instance, given a corpus C composed of two documents d_1 ={Reviewing word embeddings. Word embeddings for ML.} and d_2 = {NLP vs ML. ML vs DL.} - C={d_1, d_2}, the matrix M which shows the count of occurrences of words in C, using the count vector method, is shown in Table 2.1 below.

	Reviewing	word	embeddings	for	ML	NLP	vs	DL
d_1	1	1	1	1	1	0	0	0
d_2	0	0	0	0	1	1	2	1

Table 2.1.: A simple example for count vector

- **TF-IDF vectorization**: TF-IDF assigns the relative frequency of a word in a document with regard to its inverse frequency in the entire corpus. The intuition behind TF-IDF is to give less weight to words that are common in the document collection and higher wight to those that occur only in some documents. Consider a corpus C, a document $d \in C$ and a term $t \in d$, the TF-IDF value for the term t in document d is computed using the formula shown in Equation 2.1 [12].

$$TF - IDF(t, d) = TF(t, d) * IDF(t) \tag{2.1}$$

where

$$TF = \frac{(Number\ of\ times\ term\ t\ appears\ in\ d)}{(Number\ of\ terms\ in\ d)}$$

and

$$IDF = log \frac{(number\ of\ documents\ in\ C)}{(number\ of\ documents\ in\ C\ a\ term\ t\ has\ appeared\ in)}$$

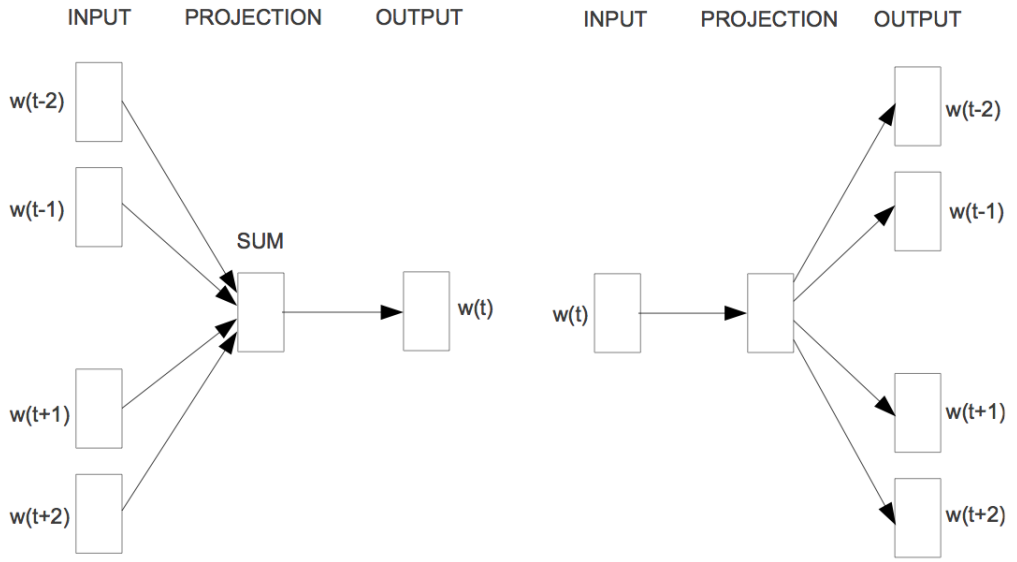
Despite its simplicity, TF-IDF doesn't capture the position of a term in a document and it also doesn't consider semantics of terms. Moreover, terms with high TF-IDF value in a document may not make sense with the topics discussed in the document.

- **Co-Occurrence Vector**: The idea that motivates this method is terms that tend to occur together have similar contextual meaning. The two main concepts of this approach are contextual window and co-occurrence . Contextual window is a slice of text specified by number and direction. Direction can be either right, left or around. For instance, the contextual window of 3 (around) for some term t means a segment of the text composed of 3 terms to the left of t and another 3 terms to the right of t. Once the context window is specified, the co-occurrence between two terms t_1 and t_2 can be defined as the number of times they appeared together in the context window. The core advantages of this method are it preserves semantics of terms, it uses singular-value decomposition (SVD), and once computed can be used anytime. On the contrary, the main disadvantage of this approach is it consumes a huge amount of memory to store the co-occurrence matrix.

Prediction based Embedding: The vectorization methods we have discussed so far are all deterministic. They are not capable of predicting words. With the intention of addressing the limitations with these methods different scientists have proposed methods which are based on neural networks. Unlike frequency-based embeddings, prediction based embeddings use neighboring words to predict a target word using learned dense embedding vectors [13].The two most common neural network based word embeddings are Glove and Word2Vec. In addition, a technique called "Embed, encode, attend, predict" which is based on DL is proposed.

- **Word2Vec** It is the most widely used neural network based word embedding relying on the assumption that words that occur together in similar contexts have similar meaning. Word2Vec comes with two alternative flavors *Continuous Bag of words (CBOW)* and *skipgram model* introduced in 2013 and has been a base for different NLP tasks [14]. The authors improved this method again in order to enhance its training speed and accuracy [15]. The CBOW architecture is based on the idea that a target word can be predicted by looking at some of the words that come before and after this specific target word as depicted in Figure 2.1a. It is named as CBOW because it follows the continuous representations where order is of no importance. On the contrary, skip-gram model, which is seen as the inverse of the CBOW method as shown in Figure 2.1b, predicts the neighboring words from the target word instead. In order to handle antonym relationships, a research was conducted which proposes a method to determine if two words with similar vector representations are antonyms or synonyms [16].

  Despite its contributions, there are various challenges with Word2Vec such



(a) cbow architecture (b) skip-gram architecture

Figure 2.1.: Word2vec architectures[15]

as:

- *Inability to handle previously unknown words*: Words that do not occur in the training data set (in the vocabulary of the the Word2Vec model) are assigned no vector representation as it is difficult for the model to interpret the word. Therefore, while using this model to build any application, if new words appear, then they will either be removed or be assigned random numbers. This negatively affects the performance of the application.
- *Not capable to capture morphological similarity*: Humans are able to guess if a word ends with the syllabus "less" there is a high probability that it means "a lack of something". However, since Word2Vec represents words as independent vectors, it is challenging to represent words of methodologically rich languages like German[1].
- *Scalability issues*: If it is required to make the model cross-lingual, new embedding matrices have to be added.

- **Glove** Glove [17] is the second mostly used word embedding technique next to Word2Vec. The authors of Glove believe that the Word2Vec embedding doesn't consider the global co-occurrence of words and focuses only on analogies. Therefore, there idea is to fill this gap by taking into consideration global word co-occurrence statistics explicitly when generating vectors for words. In

[1] http://blog.aylien.com/word-embeddings-and-their-challenges/

Glove co-occurrence probability is taken as a measure for similarity of words. Since both Glove and Word2Vec relys on the same assumption which states that words with similar meaning intend to occur in the same context, the result returned by both models are very much similar [18].

- **FastText**: This approach is introduced as an improvement over Word2Vec in 2017 by the AI research team at Facebook [19]. FastText is based on the concept modular embeddings which focuses on capturing the similarity between morphologically related words. The idea is to generate vectors for sub parts of a word and then combining these vectors using a composition function to form the embedding for the word instead of taking the word as a single entity and computing its vector representation directly. The algorithm is based on skip-gram architecture with negative sampling which was proposed previously for Word2Vec embedding. The core advantages with this embedding method are the capability of using smaller vocabulary with big corpora and capturing morphological information of words. Moreover, this approach motivates the application of embeddings to a more higher structure such as phrase or sentence embeddings because this method shows that embeddings can be composed by combining vectors of word parts to a single vector for the whole word.
- **WordRank**: This embedding algorithm is based on context window and designed to be trained on a corpora [20]. This Embedding model works by ranking the context words of each target word so as to optimize its word representations. This makes the approach more subtle for retrieving most similar words to a target word.

2.1.2. Higher Level Text Embedding

In order to compute similarity between text segments, most text similarity measures require the input text to be represented using vectors. To this end, different approaches have been proposed so far to represent more higher level structures such as phrases, sentences, paragraphs and documents using vectors. Most of these approaches are based on the unsupervised word embedding techniques discussed above. Therefore, in this section, we discuss higher level text embedding methods.

Vector Space Model (VSM): It is a model which represents a document using a vector with the size of the vocabulary where each value in the vector indicates a weighted frequency of a certain word in the document. Cosine similarity and dot product are the most common similarity measures that are used with VSM document representation scheme.

Embedding Centroids: One way to compute embeddings for high level structure of text is to compute the centroid of the vectors of all the words in the text. It is also possible to use VSM vectors as coffecients and compute weighted centroids. However, using the centroid method may result in loss of information [18].

Averaging Word Embeddings: The simplest method to generate a vector representation for a sentence/document is using a specialized word embedding algorithm which takes the average of the vectors of the words existing in the sentence or document [21, 22, 23, 24, 25]. However, with these method the actual semantics of the sentences are not captured because it loses the order among words of a sentence and also it gives all words equal importance. Another approach to compute a vector of a sentence is to use weighted mean of vectors of the words in the text by using the TF-IDF value of the words as weights [26]. Despite the ability of this approach in identifying relevant words in a sentence, it is still not able to capture the order of the words and this may lead to having two sentences with different meaning being represented with similar vectors just because they contain similar set of words.

Doc2Vec: Doc2Vec [27] is a paragraph level embedding proposed by the authors who first introduced Word2Vec [14]. Doc2Vec has the same structure as Word2Vec with the exception that Doc2Vec adds a global vector to the context representation of a document. However, Doc2Vec embedding does not work well with cosine-similarity metric as compared to other embedding techniques do and for this reason it is rather used as feature extraction method (i.e. the vectors generated using Doc2Vec are used as feature vectors) in ML algorithms [18]

Doc2VecC: It also follows the technique discussed before called *averaging word embeddings* to get document embedding where the word embedding is trained specifically to be averaged for document embedding purpose [28]. The main difference between Doc2Vec and Doc2VecC is the former trains its document embeddings where as the later just uses the trained Word2Vec embeddings directly for averaging instead. Moreover, Doc2VecC adds a corruption component which enables the model remove random words from the training data so as to improve its model's generality [25].

Word Mover's Distance (WMD): While all the document similarity methods discussed here are based on assigning embeddings to each document by composing word embeddings, WMD doesn't compute vector representations for each text item (document, phrase, sentence) instead uses a distance function to compute similarity between two documents by taking two sets of word embeddings one for each document [29]. Since the function used by WMD is computationally expensive it is not recommended to use it for applications like clustering that requires computing similarity between each pair of sentences [18].

Skip-thoughts: This method learns sentence embeddings by using the relations between consecutive sentences in the training data set by utilizing DL architecture [30]. It's learning scheme is similar to that of Word2Vec's skip-gram architecture. It takes the neighboring sentences as the context of the target sentence with loss function for computing prediction error and a stochastic gradient descent (SGD) method for optimization purpose.

Sent2Vec: This sentence embedding [31] follows the same technique as Doc2VecC except its optimization procedure focuses more on sentences or paragraphs compositionality while Doc2VecC's optimization is specifically designed for full document embeddings.

Embed, encode, attend, predict: This is an approach based on bidirectional Recurrent Neural Networks (RNNs) to generate vector matrix for sentences using the vectors of the words in the text[2]. The method starts with computing word embeddings for words and proceeds with representing sentences by employing an attention mechanism. The actual steps used in learning the sentence embeddings are discussed as follows:

1. **Embed**: The first step is to create an embedding table to represent words or tokens using vectors and for such purpose any of the appropriate methods from the word embedding techniques discussed so far can be chosen.

2. **Encode**: In this step, taking the word vectors as an input, a sentence matrix is computed where each row represents a token or a word in the sentence. A bidirectional RNN which combines the result of a forward pass and a backward pass is used to generate the sentence matrix.

3. **Attend**: In order to pass the sentence matrix as an input for the prediction phase, it is required to reduce it to a single vector representation. To that end, the current step, the Attend step, performs the matrix reduction process by making use of an attention mechanism. Since the attention mechanism takes context into consideration, this approach doesn't entertain the common problem of losing information that occur due to reduction. Researchers in the area of NLP and ML have introduced methods to reduce two sentence matrices [32] in to a single vector and a single sentence matrix in to a vector [33]. The attention method is a pure reduction operation because instead of deriving a context from the vector input, it learns a context vector using the model.

4. **Predict**:This step uses the generated single vector in the Attend step to acquire the final intended representation of the text or sentence which can be a class label, a vector, etc.

2.1.3. Text Similarity Measures

Text similarity measures are one of the core components which play important roles in text-based applications such as TD, classification, clustering, IR, and summarization. Text similarity metrics ranges from word similarity methods up to document similarity methods. Computing word similarity is a base for all other higher level text similar-

[2] `https://explosion.ai/blog/deep-learning-formula-nlp`

ity techniques such as phrase, sentence, paragraph, and document similarity measures. These metrics can be broadly categorized as string-based similarity, corpus-based similarity, and knowledge-based similarity measures [34]. The string-based similarity is further divided into character-based and term-based similarity measures. Since one of the research questions being addressed in this thesis work is adapting a clustering technique for TD process and the sentences being clustered are represented in vectorized form, we discuss the similarity measures that work well for this scenario. Therefore, here we only present the term-based similarity measures as those are the ones which are very commonly being used by various clustering algorithms.

The well known term-based similarity measures consist of block distance, cosine similarity, dices coefficient, euclidean distance, jaccard similarity, matching coefficient, and overlap coefficient [34]. The description of these metrics is depicted in Table 2.2.

Table 2.2.: Text Similarity Measures

Metric	**Description**
Block distance	aka Manhattan distance, boxcar distance, absolute value distance, L1 distance, city block distance. computes the traveling distance from one point to another in a grid like path. The block distance between two points is the sum of the differences of their corresponding components or coordinates. It is not recommended for k-means clustering [35]
Cosine similarity	computes similarity between two points or vectors by taking the cosine of the angle between them
Dices coefficient	defines the distance between two strings as twice the number of terms that are common for both strings divided by the total number of terms in both strings
Euclidean distance	aka L2 computes square root of the sum of squared differences between corresponding components of the two points. Recommended for k-means clustering [35] and very popular with hierarchical clustering algorithms as well
Jaccard similarity	defines the distance between two strings as the number of terms that are common for both strings divided by the total number of terms in both strings
Matching coefficient	computes similarity between two vectors by counting the number of similar terms they have (i.e. dimensions on which both vectors are non zero)
Overlap coefficient	same as Dice's coefficient except it considers two strings exactly the same if one is subsumed by the other

2.2. Clustering

Clustering is unsupervised machine learning which aims at grouping together similar elements of a data set in to the same cluster and those dissimilar in to different clusters. It is called among the unsupervised learning schemes because it doesn't make use of pre-labeled data in order to train its model and learn from its experience to cluster new observations. Mostly, clustering quality is measured by its intra-connectivity (i.e. density of a cluster) and inter-connectivity (i.e. connectivity between different clusters) values. A high intra-connectivity indicates a quality clustering arrangement because the observations categorized within the same cluster are highly related to each other whereas a low inter-connectivity shows that individual clusters are dissimilar to each other.

2.2.1. Types of Clustering Techniques

There are different ways to categorize clustering techniques and algorithms. Generally, a clustering can be called as *soft* or *hard* depending on how an observation is assigned to a cluster. A hard clustering is one in which each observation is in exactly one cluster whereas a soft clustering assigns a degree or probability to which an observation belogs to a cluster. On the other hand, clustering techniques can also be broadly classified into three categories (i.e. partitioning methods, density-based clustering, and hierarchical clustering) based on the perspective the technique considers for solving the problem [36]. The clusters considered for review here are those that are common for clustering big data.

Partitioning methods: These methods are used to classify data points, within a data set, into multiple clusters based on their similarity. Most of these algorithms require the data analyst or the user to specify the number of clusters needed to be generated.

The well known algorithms in this category are briefly discussed as follows:

- **K Means Clustering**: It is one of the easiest unsupervised machine learning algorithms that is used to partition a data set into a predefined number of clusters [37]. K indicates the number of clusters that are required to be generated. The main purpose of this algorithm is to group together observations by making within-cluster variations as small as possible.

- **Other techniques**: *K-medoids* clustering also called as *PAM* (Partitioning Around Medoids) [38] and *CLARA (CLastering Large Algorithms)* [39] are some of the other common partitioning techniques. K-medoids has the same structure as k-means except it uses mediods instead of means in order to be more robust to outliers. However, it also suffers from the same problem that k-means entertains which is difficulty in pre-defining the desired number of clusters. CLARA is designed based on PAM and it focuses mainly on

addressing the practicality issue which occurs with both k-means and k-medoids as they are slow to handle large volume data.

Density-based clustering: This technique works by grouping together data points into arbitrary-shaped clusters. It follows the principle that an area where the density of data points exceeds some predefined threshold will be considered as a cluster. This helps to identify outliers and in the data set. There are different density-based clustering algorithms [36, 40]. We will briefly discuss some of the most common ones here.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: It [41] is widely known as the first density-based clustering algorithm introduced and has been a base for more clustering techniques in this category. In DBSCAN, areas with high density of observations or data points are taken as clusters and those with low density are more likely noises or outliers. It works by classifying the data points as core, border, and noise points and takes three parameters as inputs; the size of he neighbor points, the radius of a neighborhood, the minumum number of data points in a neighborhood.

- **Other Algorithms**: RDBC is an algorithm designed as an extension of DBSCAN and it doesn't need the desired number of clusters to be predefined. RDBC performs better as compared to DBSCAN as it changes the parameters intelligently and separates the process of identifying the core data points from the actual clustering of each data point. Some of the other known density-based clustering algorithms are OPTICS(Ordering Points to Identify the Clustering Structure) [42] and DENCLUE(DENsity-based CLUstEring) [43, 44]. OPTICS is an algorithm for finding density-based clusters in spatial data whereas DENCLUE uses kernel density estimation based model and it doesn't work well on data with uniform distribution which leads to its incapability of handling high-dimensional data.

Hierarchical clustering: The hierarchical clustering techniques group data points into a tree of clusters which is called dendrogram [45]. A dendrogram is constructed either in top-down or bottom-up approach containing a sequence of nested cluster. A bottom-up construction of dendrograms is called agglomerative clustering whereas the one that follows top-down approach is called divisive clustering. A dendrogram is formed in such a manner that all inclusive cluster resides at the top and singleton clusters of individual data points are positioned at the bottom of the tree. At each intermediate level of the dendrogram, a cluster is formed by either combining two clusters from the next lower level or by splitting a cluster from the next upper level depending on the type of the approach used for constructing the dendrogram.

- **Agglomerative clustering**: It starts by assigning each data point to a singleton cluster and follows by merging the two cluster which are most similar

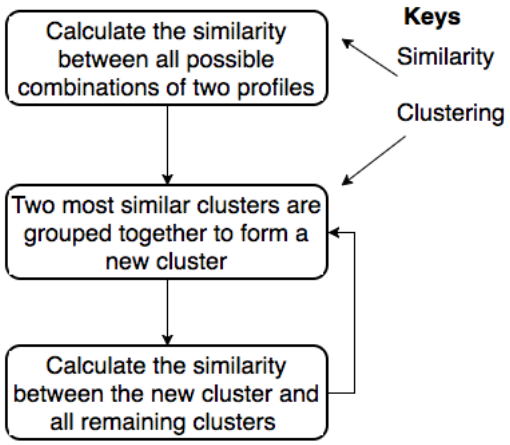


Figure 2.2.: Agglomerative Clustering[36]

(or closest) clusters together at each iteration. It stops when only one cluster is left containing all of the observations (data points) from the data set or when a certain stopping criterion is met. Any chosen distance metric can be used to determine the distance between two observations. Moreover, there are various methods which can be used in order to decide which clusters have to be merged at each iteration. The workflow involved in agglomerative clusering showing the main steps is depicted in Figure 2.2 [36]. One of the widely used agglomerative clustering techniques is ward's agglomerative clustering algorithm which uses the sum-of-squares criterion [46].

- **Divisive clustering**: The opposite of the agglomerative clustering is divisive clustering which starts with making an all inclusive single cluster and continues by splitting the most sparse cluster into two clusters based on distance functions.

A tabular representation of the comparison of the most popular algorithms from each category discussed above is shown in Table 2.3.

2.2.2. Determining Optimal Number of Clusters

Apriori information about the number of clusters mostly in partitioning clustering algorithms is required. Moreover, it is desirable to be able to know the number of clusters in hierarchical clustering as well. In addition to allowing analysts to look at the resulting dendrogram, it will be an interesting feature for the system to be capable enough to propose an appropriate number where the dendrogram can be cut.

In general, determining the optimal number of clusters has been a challenge and it

Table 2.3.: Comparision of Clustering Algorithms

Algorithm	**Advantages**	**Disadvantages**	**Applicability to Big data**
K-Means	Easy to implement.	Challenging to determine the number of clusters in advance[36]. Doesn't handle noisy data [40]. Sensitive to outliers	Suitable as it can be easily parallelized
DBSCAN	Capability to generate clusters of different size and shape	does not work well with clusters of different densities	Suitable as it is robust to outliers
Agglomerative clustering	No apriori information about the number of clusters required. It is easier to decide on the number of clusters by looking at the dendrogram Easy to implement. More informative.	The algorithm can never undo what it has already done. Time complexity of at least O(n2 log n) is required	Not so recommended because of its time complexity

is still an open research as there is no best method to do so. In a broader sense, there are two ways to decide on the number of clusters; manually by looking at the result of the clustering process and automatically by using different approaches. Even if the manual process yields better result, it is time consuming and not ideal to do it manually if the data set is big. Therefore, it is more desirable and appropriate to have an automatic approach for determining the correct number of clusters.

1. **Elbow Method**: Given the function within-cluster sum of square (WSS) which is used to measure the compactness of a clustering, with the intention of making it as small as possible, the main concept behind the elbow method is to take a number of clusters where adding another cluster doesn't improve the total WSS value. The first step in applying elbow method is to run the clustering algorithm for each k in some range, and then for each k compute WSS, finally plot the graph according to each wss - k pair and take the knee on the graph as the optimal number of clusters [47].
2. **Average silhouette method**: It is an alternative to elbow method and can be used by any clustering technique. It measures the quality of a clustering by com-

puting the average silhouette of data points for different values of k. A high average silhouette value indicates a good clustering and low value shows low quality clustering. It is quite similar to the elbow method except instead of WSS it uses average silhouette value and instead of an elbow it looks for the maximum value on the graph [48].

3. **Gap statistic method**: An approach which works with any clustering technique and it works by comparing the total intra-cluster variation for different values of k with their corresponding expected values under null reference distribution of the data set. The value which maximizes the gap statistic will be taken as the value for k - the optimal number of clusters [49].

Table 2.4.: Comparision of automatic methods for determining optimal number of clusters

Method	Advantages	Disadvantages
Elbow Method	Very clear and easy to implement	Ambiguous in some cases, measures a global clustering characteristic only
Average silhouette method	Unambiguous	Measures a global clustering characteristic only
Gap statistic method	Accurately estimates single clusters	The performance decreases as the clusters separate. Reference distribution has to be chosen by the data analyst

2.2.3. Applications of Clustering

In general, clustering can be employed in different kind of data analysis services. Its result can be utilized directly for decision making purpose or taken as an intermediate step in other data-centeric tasks. For instance, it can be used in Data Mining, Pattern Recognition, Image Analysis, Bioinformatics, ML, Voice Mining, Image Processing, Text Mining, Web Cluster Engines, Whether Report Analysis [50]. In addition to these, it can be put into use in applications such as Recommendation Engines, Market Segmentation, Social Network Analysis, Search Result Grouping, Medical Imaging, Image Segmentation, and Anomaly Detection.

Besides, there are different researches conducted to build TD by incorporating different clustering algorithms. For instance, "text clustering for TD" [51] and "A topic detection approach through hierarchical clustering on concept graph" [52] are some of the TD approaches which are based on clustering techniques. More detailed review of these papers is presented in Section 4.

2.3. Ontologies

According to its use in computer science specifically in the area of Artificial Intelligence (AI), an ontology refers to an engineering artifact, built using a specific vocabulary to describe the domain being modeled, having a set of explicit assumptions about the intended semantics of the terms in the vocabulary and the relationship among the terms [53]. In a similar notion, an ontology can be defined as a "specification of a conceptualization" [54]. In the following sub sections, the basic components of ontologies are presented followed by the review of the main roles of ontologies in different data analysis service more specifically in TD.

2.3.1. The Common Components of Ontologies

The most common components of ontologies are individuals, classes, attributes, and relations. Individuals are also known as instances or objects and they are the basic or ground elements of an ontology. Classes (or concepts) are made up of individuals with common attributes. Attributes also called as properties are the characteristics that individuals entertain. The possible association between individuals is refereed to as relations or relationships.

Ontologies are encoded using ontology languages in order to let computers understand and interpret the stored information in an ontology and make intelligent inferences or reasoning activities. Even if there are different ontology languages available such as description logic, first order logic, RIF (Rule Interchange Format), KIF (Knowledge Interchange Format), Resource Description Framework (RDF), and RDFS (RDF Schema), OWL (Web Ontology Language) is the widely used and the recommended ontology language by W3C.

In regards to OWL ontologies, a knowledge base is defined as a combination of TBox and ABox where TBox is a set of terminological axioms and ABox is a set of assertional axioms. In most cases, a terminological axiom is a subsumption of the form $C \sqsubseteq D$ which states that all of the instances of the concept C are also that of D. Thus, the axioms in TBox are used to construct the taxonomy of the ontology. On the other hand, assertional axiom is used to describe a concrete situation such as an individual being an element of some class or two objects being related to each other through a certain relationship. Based on this definition, it is possible to entail the statement that an ontology can be seen us a special kind of knowledge bases which is mainly dedicated for the semantic web.

Nowadays, as it is depicted in Figure 2.3, there are plenty of linked data sets available on the internet from which DBpedia[3] is a widely used domain-generic data set covering variety of topics. DBpedia has been used as a dataset for different linked data

[3] `http://wiki.dbpedia.org`

based applications. The W3C standard query language which is used to query ontologies and linked data sets like DBpedia is SPARQL (SPARQL Protocol and RDF Query Language)[55].

2.3.2. The Roles of Ontologies in TDL

Generally, ontologies are used either by experts or by computers. They can provide support in building semantic web applications. Ontologies have been employed in different data mining tasks such as classification, clustering, information extraction, recommendation system, link prediction, association rule mining [56].

Ontologies have been used for the purpose of TD by researchers in the area of semantic web so far. In most researches, ontologies are used to generate semantic labels for identified topics. OntoLDA [57] is one of the studies that make use of ontologies for topic labeling purpose. Besides, there are works which use ontologies for identifying topics as well. For instance, Wikipedia can be used as a knowledge graph to help build a topic identification system [58].

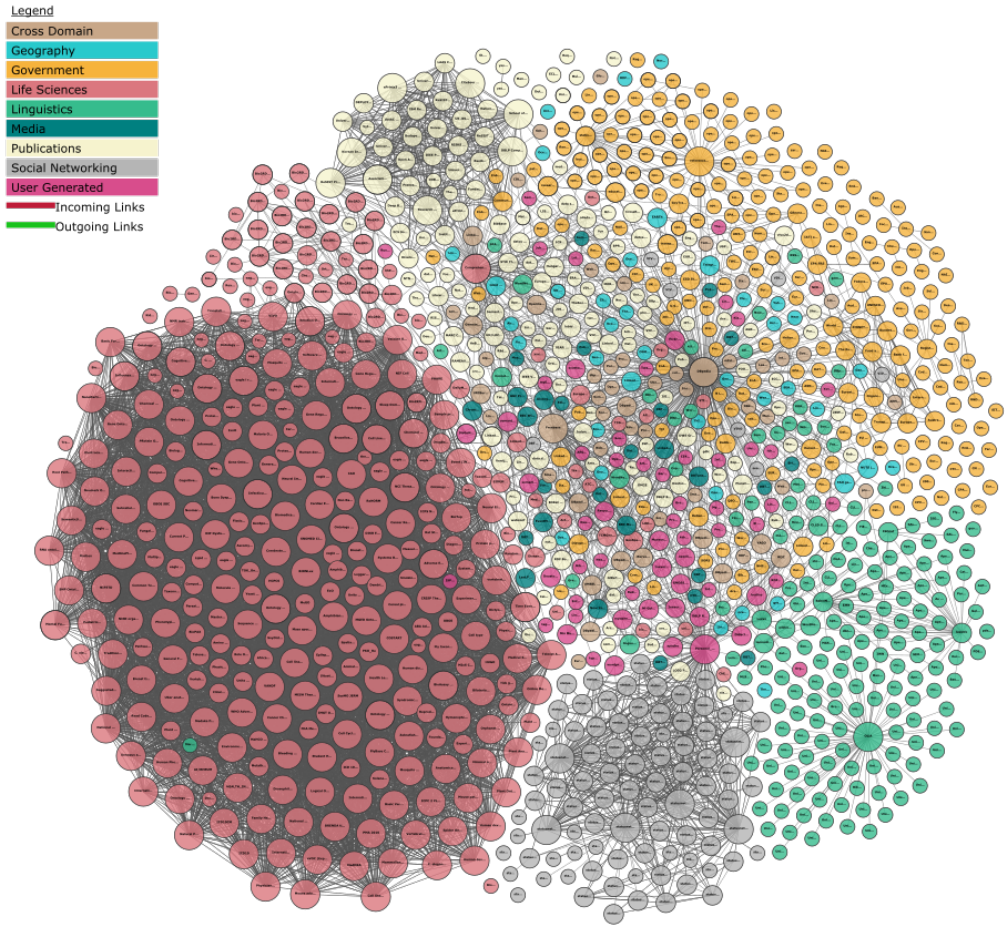


Figure 2.3.: The Linked Open Data cloud diagram[4]

2.4. Summary

In this chapter, the topics such as word embedding, higher level text embedding, text similarity measures, clustering and its applications in TD, and overview of ontologies and its use for TL are presented. As it is discussed in the literature review about word embedding, the prediction based word embedding techniques are more popular lately and perform well with various ML and DL algorithms. The word embedding algorithms Word2Vec, GloVe, FastText, Sent2Vec, and Sent2VecC are a matrix factorization problems with Word2Vec being the base for almost all of the other prediction based techniques discussed so far. On the other hand, for higher level text embedding we have

[4] http://lod-cloud.net

covered various methods in the review. The easier and mostly used method for computing embeddings for higher level text structures is using weighted-average of vectors of words that make up the text.

Even though there are various text similarity measures classified in broader categories as knowledge-based, corpus-based, string-based metrics, in this chapter we focused on string-based (specifically term-based) metrics as those are the ones which are more applicable to the clustering methods used in this research. Moreover, various types of clustering techniques and algorithms are discussed with the intention of providing information to readers on how clustering algorithms work and how different they are with each other. Finally, a high level description of ontologies and their use in the design and development of TDL applications with foucs on DBpedia has been given.

3. Requirements Analysis

In this chapter, the assumptions driven to achieve the intended result with this thesis work will be described followed by the requirement analysis specification, derived for the proposed system, consisting of description of the general use case with functional and non-functional requirements.

Requirements are features of a software system or a software system function used to fulfill system purpose. Requirements analysis is the first phase in software engineering process encompassing the process of eliciting, analyzing and recording requirements. It can play a vital role in the success of a software development because it provides a model of system information, function and behavior for software designers. The generated model can be used later in other software development phases mostly in designing phase.

Even though requirements can be categorized in many ways, the most common way to classify them is as functional and non-functional requirements. Functional requirements are those that are used to capture the important tasks or actions involved in the system whereas non-functional ones are those used for evaluating the operation of a system. Requirements can be captured in many ways. One of the most common ones is using use cases.

3.1. Assumptions Taken

In order to address the research questions defined in Section 1.3, the following assumptions have been made.

- **Topic**: In other related works like Latent Dirichlet Allocation (LDA) [3], a topic is taken as a set of semantically related terms. However, in this masters thesis, a topic is defined as a set of sentences which are semantically similar. The similarity between sentences is determined using a text similarity measure.

- **Topic Boundaries**: A boundary of a topic p is a set of pairs of positions in a transcript where each pair, {i, j} with $i <= j$, consists of positions of two sentences S_i and S_j with every sentence between S_i and S_j discusses the topic p and the sentence that comes before S_i (S_{i-1} if there is any) and the sentence that comes after S_j (S_{j-1} if there is any) do not belong to the topic p.

  In a more formal sense, a transcript t is defined as a sequence of sentences as shown in Equation 3.1 where n is the total number of sentences in t and i is the index of the sentence $S_i \in t$. Given t, a boundary B of a topic p in t is defined in Equation 3.2.

$$t = \bigcup_{i=1}^{n} S_i \tag{3.1}$$

$$B(p,t) = \bigcup_{1<=i<=n, i<=j<=n}^{n} \{i,j\} \tag{3.2}$$

  where i and j are the positions/indices of sentence s_i and s_j respectively with all the sentences from s_i up to s_j belong to the topic p.

- **Data Point**: In the clustering algorithm, we take a sentence to be the smallest data point to be clustered.

- **Sentence-Topic correlation**: A sentence is assumed to have exactly one topic i.e. a hard clustering algorithm will be used which assigns one sentence to only one cluster.

3.2. Use Case

Use case modeling is one of the methods used for describing the software requirements of a system. A use case model is composed of use case diagrams and their descriptions. Use cases play a great role in capturing and showing the interactions among actors, components of the system, and any external systems. The use case diagram depicted in Figure 3.1 shows how the Topic Detection and Labeling (TDL) system interacts with external actors or external systems.

The description of the terms used in the Use Case diagram are stated as follows:

- *Webinar Transcript*: The text equivalent of a video/audio file generated manually or by speech to text recognizer.

- *External Knowledge base*: An ontology or a linked data set like DBpedia which contains a set of RDF triples.

- *User*: A person who interacts with the system to get the service provided by the system.

As it is illustrated in Figure 3.1, first, a user has to upload a transcript to the system. Then the system will detect topics (topic boundaries) in the uploaded text document. Once the boundaries are identified, the terms which can be used to label the topics will be generated with the help of external knowledge base. Finally, the result, the topics and their labels, will be returned to the user.

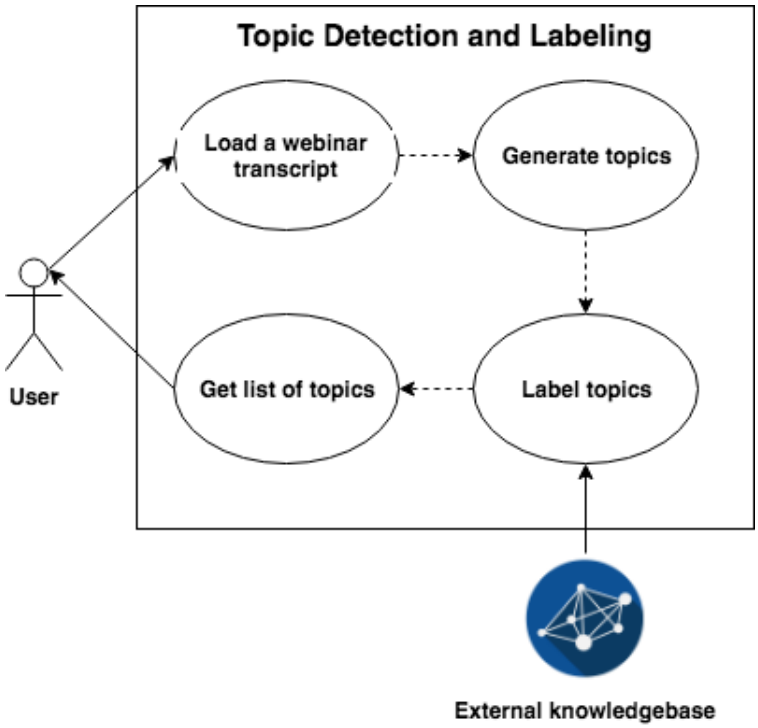


Figure 3.1.: A Use case diagram.

3.3. Functional Requirements

Functional requirements are used to describe what the system is required to do. Specifically, these requirements state clearly what data is entered to the system and by who, what operations are performed by each component of the system, and the work-flows within these components and with external systems as well. The requirements are intended to show the behavior of the system.

In compliance with the use case diagram in Figure 3.1, the following functional requirements have been specified for the proposed system.

- FR-1 **Topic Detection**: The system should be able to identify topics discussed in a transcript by putting sentences that are semantically similar into the same group. Therefore, a topic identified has to be a set of sentences as the way the concept of *topic* is defined in this research.
- FR-2 **Optimal Number of Topics**: The system should be capable enough to find the optimal (the most desirable) number of topics in a transcript with no or very less user involvement.
- FR-3 **Boundary Detection**: For every detected topic in a transcript, the system should be able to identify the boundary of the topics in the transcript.
- FR-4 **Topic Labeling**: For each identified topic, maximum of 5 terms should be assigned by the system to be used as labels for the topics. These terms have to be generated in a way that they can semantically represent their respective topic.
- FR-5 **Utilizing External Knowledge-base**: In the process of TL, the system should be able to make use of external knowledge base to rank candidate labels according to their relevance.

3.4. Non-functional Requirements

Non-functional requirements are designed to support functional requirements by imposing quality constraints related to performance, security and reliability on the system. Having these requirements defined earlier helps in making sure the system achieves the level of quality that users expect from it. The following non-functional requirements are derived for the proposed TDL system.

- NFR-1 **Response Time**: As it is stated by Tedtalk[1], one of the most widely known online webinars/talks providers, the maximum length of a single TED talk is 18 minutes starting from about 2 minutes minimum duration. The word count of an 18 minute TED talk is about 2800 - 3800 (average -3300) words[2]. The average number of words for a single sentence is about 15-20 (average -18) words[3]. Therefore, a tedtalk of the maximum allowed length contains in average $\frac{3300}{18} = 188$ sentences. On the other hand, a single webinar or meeting held in a company (like LogMeIn), conferences, seminars, and educational online events may take up to 60 - 90 minutes. Thus, such videos may take in average 786 sentences as calculated using the formula in equation 3.3 with $averageWordCount = 3300$,

[1] https://www.linkedin.com/pulse/20140313205730-5711504-the-science-behind-ted-s-18-minute-rule

[2] http://expertenough.com/3077/10-steps-to-create-a-standing-ovation-worthy-ted-talk

[3] https://strainindex.wordpress.com/2008/07/28/the-average-sentence-length/, http://countwordsworth.com/blog/what-is-a-good-average-sentence-length/

$minLength = 60$, $maxLength = 90$, $maxDuration = 18$, $minWords = 15$, and $maxWords = 20$. These parameter assignments are discussed above in detail.

$$transcript\_size = \frac{(\frac{averageWordCount*(\frac{minLength+maxLength}{2})}{maxDuration})}{\frac{(minWords+maxWords)}{2}} \tag{3.3}$$

Since the proposed system should be able to work for both TED talks and Webinars/Meetings, an average length of these two categories will be considered for evaluation purpose, i.e. a video can have a duration of 2 - 90 (average - $\frac{2+90}{2} = 46$) minutes. The transcript generated for a video of 46 minutes has 482 sentences in average calculated using the formula in equation 3.3. The parameters in the equation are assigned the same values as discussed above except $minLength$ and $maxLength$ are given 2 and 90 respectively. Therefore, The time that the proposed TDL system takes to return topics and their labels after a user uploads a transcript of average size (482 sentences) should not be more than 1 minute.

- NFR-2 **Scalability**: The system should be capable enough to work regardless of the size of an input transcript increases.
- NFR-3 **The system should always return a result**: The system should be capable enough to always return results back to the user whether it is a list of topics or just 1 topic (i.e. It has to work accurately even if all or most of the words in the transcript do not exist in googles word2vec vocabulary).

3.5. Summary

In this chapter, the assumptions taken to solve the research questions along with the use case, the functional requirements, and the non-functional requirements derived in order to be satisfied by the proposed TDL system were discussed. These requirements have different level of importance in the system and for that reason they are classified into three categories as Must-have, Should-have, and Could-have. Must-have requirements are guaranteed to be satisfied by the proposed system. On the other hand, Should-have requirements are very much valuable if they are met whereas Could-have requirements are a plus to have met but not required to be delivered.

Table 7.6 summarizes all the requirements discussed in this chapter and classifies them as must-have, should-have, and could-have. As it can be seen from the summary, the main purpose of this thesis is to fully design, implement, and evaluate the Topic Detection (TD) component and to do as much as possible with the Topic Labeling (TL) component as it is given lower priority.

Table 3.1.: Classification of the TDL system requirements

	Requirements	Must-have	Should-have	Could-have
Functional	FR-1: Topic Detection	✓		
	FR-2: Optimal Number of Topics	✓		
	FR-3: Boundary Detection	✓		
	FR-4: Topic Labeling	✓		
	FR-5: Utilizing External Knowledge base	✓		
Non-Functional	NFR-1: Response Time			✓
	NFR-2: Scalability		✓	
	NFR-3: The system should always return a result		✓	

4. Related Work

Before attempting to find a new method which will help solve a certain research problem, it is required to first conduct reviews on the existing approaches and research works undertaken to solve the same problem and identify their contribution and limitations. To this end, since the main objective of this master's thesis is to propose a Topic Detection and Labeling (TDL) system which detects topics with their boundaries in a transcript and assigns the topics with representative labels, it is found to be vital to first study the existing automatic approaches which are used for detection and labeling of topics.

Therefore, in this chapter, the existing related works in the area of Topic Detection and Labeling (TDL) will be discussed. First, the research works proposed so far for the purpose of addressing the Topic Detection (TD) problem will be presented followed by the review on studies conducted on Topic Labeling (TL) task. Then, requirement-based high level comparison of the existing systems with the method proposed in this master's thesis will be given. Finally, the chapter will be concluded by giving a brief summary of the whole chapter.

4.1. Existing Works in Topic Detection (TD)

TD has been undertaken as a research topic for over two decades, since 1996, being a part of the well known Topic Detection and Tracking (TDT) research in the area of Natural Language Processing (NLP) and Machine Learning (ML) project sponsored by DARPA[1]. TDT is a process of detecting and monitoring new events in text, audio, or video broadcast news stories [59]. In a more general and simpler sense, it can also be defined as a process of detecting previously unknown topics and tracking the reappearance of the existing ones. It comprises five main tasks namely, Story Segmentation, Topic Tracking, TD, First Story Detection, and Link Detection tasks. Despite these tasks

[1] `https://www.darpa.mil`

existing by their own, there is no clear boundary between them regarding the approaches used to solve them. For instance, Story Segmentation can be considered as part of TD based on the kind of methods used for solving the TD task. In our case, in the way TD is defined in this research, we can consider Story Segmentation as the same as Boundary Detection (BD) which is one of the tasks in the TD component of our proposed system.

Apart from being a part of the TDT project, various researches have been conducted solely on the TD task [60, 52, 61]. In different researches, the definition of the term topic, or in more broader sense TD, may be defined in a sightly different manner depending on the structure and model proposed in the particular research. For instance, one way to define TD is to formulate it as the problem of detecting stories in multiple continuous news streams that discuss new or previously undetected events [51]. In this master's thesis, TD is a process of identifying topics and their boundaries in a transcript where topic is defined as a set of sentences with syntactic and semantic similarity.

The existing solutions provided for solving the TD problem will be categorized as supervised and unsupervised and discussed as follows.

Unsupervised Approaches

Most of the TD research works conducted so far follow an unsupervised approach due to the challenge in finding a data set labeled with topics which can be used to train a model. The most common unsupervised approaches employed for solving the problem of TD so far are based on either probabilistic clustering techniques, external KBs, latent tree models, or formal concept analysis method.

In 1999, an unsupervised topic modeling approach entitled *probabilistic latent semantic indexing (pLSI)* was proposed to capture hidden topics in documents by recreating each document from corpus using probabilistic distributions [62]. A document is a probabilistic distribution over a set of topics whereas a topic is over a set of words extracted from the corpus. Even though pLSI has advantages over its non-probabilistic previous version LSA [63] such as handling polysemy, it has its own drawbacks that have been addressed in researches that came later on. For instance, the number of parameters in the model grows linearly with the size of the corpus, which may lead to overfitting. Moreover, since the model is based on the concept of bag-of-words, the order of words in the document is not taken into consideration i.e. the words in the document are exchangeable, this may lead to generating incoherent topics. Most of all, there is no clear way to assign probability to a new document outside of the training data set which makes the model not a true generative model. This makes pLSI not a suitable approach for topic identification as topics are generated rather online.

In another research, a generative model called Latent Dirichlet Allocation (LDA) came with valuable improvements over pLSI [3]. LDA is a generative probabilistic model of a corpus which takes a document as a random mixture of latent top-

ics, where a topic is of probability distribution over words of the corpus. LDA is bayesian version of pLSI with vital improvements. It makes pLSI a generative model by assigning probability to documents outside of the training set i.e. by imposing drichilet prior on the model parameters. In LDA parameters are regularized so as to avoid the problem of overfitting. However, LDA has quite unavoidable limitations such as focusing on word probability rather than the actual semantics of the words in a document, being incapable of determining the optimal number of topics, and not suitable enough to get topic boundaries in a document.

Various approaches that are based on clustering techniques are proposed so far to address the TD problem. The paper [64] presents a TD approach which is based on a simple-linkage agglomerative hierarchical clustering and IDF-weighted cosine coefficient document similarity metric. Topics generated using this method are susceptive to off-topic materials which degrades the quality of the TD system. Another clustering-based TD approach is to use an incremental k-means algorithm for grouping stories together so as to generate topics [60]. This method uses the bayesian belief network (BBN) probabilistic similarity metric to find the most similar cluster for a story and a TF-IDF weighted cosine distance metric with BBN topic spotting metric as threshold metrics to decide whether a story should be merged with a cluster. Another related paper [64] presents a TD system which uses k-means clustering with cosine similarity for TD by representing text with weighted chosen feature words and running the clustering algorithm on the vectors. This method doesn't perform very well if the keywords selected do not cover the whole content and semantics of the document. Moreover, in another research work, a TD algorithm is designed by using a modified TF-IDF algorithm called TF-Density to give weight to words in a text and then by applying clustering algorithm with cosine similarity measure [65].

Another research [66] presents a TD system by utilizing a clustering technique to group together sentences with similar meaning/topics. For the clustering algorithm, the distance matrix is created by using the semantic distance between two nouns considering their hypernym relation in Wordnet. The clustering algorithm applied in this paper works in a way that the initial clusters are generated by 10% or 20% of the total number of sentences. This amount, k, is determined by the compression rate needed for the summary. These initial clusters are generated by putting together k pairs of closest sentences. These clusters are further updated by adding the rest of the sentences to the nearest clusters. Finally, a topic strength summarizer for single documents is used to evaluate the approach. One of the weaknesses of this approach is that the number of topics have to be given first. The other one is that the similarity of sentences is not symmetric and they used average value for compensation which is still not quite appropriate. Moreover, the order of sentences in a document is not considered which we claim affects the performance of the clustering process.

Applying a hierarchical clustering approach on concept graphs constructed from documents is another method proposed for TD [52]. The idea in this paper is to represent a document with concept feature vector where each concept contains two words from the document and a concept graph is generated using the collection of concepts from documents. Given the concept graph, an agglomerative hierarchical clustering algorithm is used to get clusters out of the graph. Another clustering approach to TD is a constructive-competition clustering algorithm which is inspired by competitive learning from the neural network research [51]. A keyword based TD [67] system that uses the induced k-bisecting clustering algorithm with Jensen-Shannon divergence of probability distributions as a distance measure has also been proposed.

The role of formal concept analysis (FCA) for TD has been investigated to generate topics from tweets [61]. In this research, a set of formal concepts (which are made of tweets as objects and terms as attributes) are generated to form concept lattices and from these latices a set of topics or clusters are chosen. Some of the drawbacks of this approach are redundancy and sparsity that rise due to the representation of text/tweet with just terms with out involving text structure or semantics. Another related paper [68] also discusses a TD system based on FCA.

KBs such as WordNet, Wikipedia, and DBpedia can be used in different ways in TD process. A research [69] proposed a TD system by training a perceptron with pair-wise comparisons of documents to determine if two documents discuss the same topic or not. It works by representing documents with semantic classes and using a separate similarity measure for each class to perform class-wise document comparison. Places, names, temporal expressions and general terms are the classes considered for the representation of documents. Geographical and temporal domain specific ontologies are used for formulating the similarity metrics for places and temporal classes respectively. Despite the advantage of representing documents semantically, the performance of the system rather degraded because of the similarity functions used. A similar approach which generates topics from webages using complete link clustering with cosine similarity measure has also been proposed [70]. Another paper [71] presents a TD method which models topics as concept graphs where a node is a concept representing a topic and the edges are semantic relationships extracted from WordNet. However, this approach is not suitable for detecting topics in a dynamic text stream and online event detection due to the fact that these processes require incremental algorithms and the approach doesn't support that.

In addition to all the papers discussed above those based on clustering, KB, and FCA, there are some which investigate hierarchical latent tree model (HLTM) for TD purpose as well [72, 73, 74].

Supervised Models

Even though most of the research works in TD follow unsupervised approach, some works have been undertaken in order to utilize the advantage of supervised or semi-supervised modeling for detecting topics. A paper [75] presents an approach to find topics/clusters from a set of tweets which are priorly associated with a certain entity. Since the entity to which a tweet is relevant has to be given first, this approach can not be considered as a standard TD system. Another research work [76] has been conducted to detect topics in tweets about organizations. In spite of their attempt to use a semi-supervised method to learn emerging topics, the algorithm is not capable of handling general entities other than organizations such as location, people and so on. The reason behind this is that their data is collected only for organizations with the intention to capture emerging topics with respect to organizations only.

4.2. Existing Works in Topic Labeling (TL)

TL is the process of generating a set of representative terms for a topic that well capture the underlined meaning of the topic. Labeling topics with short and expressive phrases play a vital role in giving the topic a human understandable representation with meaningful interpretation and yet the labels can be used for further tuning of the TD algorithm. Various researches have been conducted in the area of TL giving a potential algorithm for labeling topics which are generated by using different TD algorithms.

The paper entitled "Automatic Labeling of Topic Models" [4] proposes a topic labeling method which is used to label topics generated using LDA topic modeling algorithm. In general, the approach proposed in this paper is composed of two parts; candidate generation and candidate ranking. In order to generate a label candidate set, it starts by querying Wikipedia with the top-N terms from the topic using Wikipedia's native search engine and site-restricted Google's search and extracting the titles from the top-ranked 8 documents. Then the initial candidates will be further processed to extract noun chunks along with their component n-grams excluding those for which their is no separate article in Wikipedia. Since this candidate term set contains stop words or words which are no relevance to the topic, the method uses the RACO lexical association method [77] to remove these irrelevant words. The candidate ranking component starts by representing each candidate label with features using lexical association measures (point wise mutual information, Student's t-test, Dice's coefficient, Pearson's X^2 test, and the log likelihood ratio), lexical properties of the candidate (the actual number of terms and the relative number of terms in the label candidate that are top-10 topic terms), and a search engine score for each label candidate. Two alternative candidate ranking algorithms, unsupervised and supervised algorithms, are proposed in this paper.The unsupervised model can make use of any of the defined features for selecting candidate labels. On the other hand, for the supervised method, a support vector regression (SVR) model is trained by combining all the features together and using a gold-standard labeling of the candidate

labels.

Using summarization technique to label topics generated using topic models is also one of the methods proposed for automatic labeling so far. A paper with the title "Automatic Labeling of Topic Models Using Text Summaries" [5], introduced a method for adapting the document summarization algorithm based on submodule optimization for solving the TL problem. The algorithm works by first selecting a candidate sentence set for each topic and then forming a topic summary using the sentences. In order to create the candidate sentence set for a given topic, the Kullback-Leibler (KL) divergence between the word distribution of the topic and each sentence in the corpus is computed. The sentences are yet ranked according to the corresponding divergence value and the top 500 sentences are taken for the summarization phase of the algorithm.

Researchers in the area of NLP have been investigating the use of ontologies for the purpose of both topic detection and topic labeling. The research *"Automatic Topic Labeling using Ontology-based Topic Models"* [6] presents a method that utilizes DBpedia to label topics which are extracted by topic models which use ontologies for the topic generation algorithm (i.e. the topic model is trained over concepts instead of words), specifically LDA. Since a topic is defined as a distribution over concepts, the system takes the top topic-concepts and maps them to an ontology in order to find ontology classes which best fit as labels for the topic.

In 2016, a method which models the topic labeling problem as a k-nearest neighbor (KNN) search problem [7] with main focus on developing fast real time labeling system has been proposed. It works by constructing a topic-label database which stores the probability distributions over words and corresponding labels and using KL divergence and Jensen-Shannon Divergence (JSD) to compute distribution similarity.

An approach which is initially proposed for labeling nodes in hierarchical clusters can also be adapted for labeling of topics generated using hierarchical topic detection algorithms [8]. It creates a label for a node in a way that the label identifies the node from the rest of the nodes in the same level (i.e. sibling nodes). To this end, the algorithm proposed in this paper starts by collecting unigram, bi-gram and tri-gram phrase statistics (i.e. document frequency and term frequency) for a target cluster S. The candidate labels for S are chosen from the phrases based on their document frequency in the cluster S, in S's parent cluster which is also a parent to all S's sibling clusters, and in an English corpus. Once the candidate labels are selected, their descriptive score is computed and used to sort them accordingly. Finally, a cut-off point is determined to choose the appropriate number of labels from the candidate set.

Table 4.1 presents the limitations of the existing TL approaches discussed above.

Table 4.1.: Drawbacks of existing TL approaches

Approaches	Description	Drawbacks
Automatic Labeling of Topic Models [4]	Selects candidate labels using Wikipedia and applies both supervised and unsupervised ranking and generates flat labels	Dependent on External Knowledge base which makes it incapable of labeling emerging topics discussed in a meeting or webinar for which there is no wikipage
Automatic Labeling of Topic Models Using Text Summaries [5]	Selects sentences and summarizes them using submodule optimization based summarization technique to use the summary as a label for a topic	It doesn't check whether the summary generated for a topic is coherent
A Novel Fast Framework for Topic Labeling Based on Similarity-preserved Hashing [7]	Uses topics generated with Labeled LDA to train the proposed approach to learn labels for new topics	Since it uses only the information from the topic words, it is vulnerable to the problem of missing the underlying meaning of the words in a topic
Automatically Labeling Hierarchical Clusters [8]	Finds a set of labels for a node in hierarchical cluster by choosing phrases from the cluster at that node which are more unique to that specific cluster	The algorithm doesn't perform well in ranking labels selected for clusters with few number of observations (i.e. topics with few number of sentences)
Automatic Topic Labeling using Ontology-based Topic Models [6]	Uses ontology to find labels for topics	Dependent on DBpedia to determine the connection or coherence between topic-concepts

4.3. Comparisons of the Existing Systems

In this section, we compare the existing and the proposed TD and TL approaches based on the requirements specified in Chapter 3. As it is shown in Table 4.2, only those existing researches which are highly related to the TDL method proposed in this masters thesis are chosen for the comparison. In this table, the check marks ✓and ✗are interpreted as fulfilled and not-fulfilled respectively.

Table 4.2.: Comparisons of TD and TL approaches based on requirements

Approaches	**Functional Requirements**					**Non-Functional Requirements**		
	FR-1: Topic Detection	**FR-2: Optimal Number of Topics**	**FR-3: Boundary Detection**	**FR-4: Topic Labeling**	**FR-5: Utilizing External KB**	**NFR-1: Response Time**	**NFR-2: Scalability**	**NFR-3: The system should always return a result**
Topic detection and tracking using idf-weighted cosine coefficient [64]	✓	✓	✗	✗	✗	Not specified	Not specified	✓
Topic detection in broadcast news [60]	✓	✓	✗	✗	✗	Not specified	Not specified	✓
A topic detection and tracking system with TF-Density [65]	✓	✓	✗	✗	✗	Not specified	Not specified	✓
A non-linear topic detection method for text summarization using wordnet [66]	✓	✓	✗	✗	✓	Not specified	Not specified	✓
A topic detection approach through hierarchical clustering on concept graph [52]	✓	✓	✗	✗	✓	Not specified	Not specified	✓
Automatic Topic Labeling using Ontology-based Topic Models [6]	✓	✗	✗	✓	✓	Not specified	Not Specified	✗
Automatic Labeling of Topic Models [4]	✗	✗	✗	✓	✓	Not specified	Not Specified	✗
A Novel Fast Framework for Topic Labeling Based on Similarity-preserved Hashing [7]	✗	✗	✗	✓	✗	Not specified	Not Specified	✓
Automatically Labeling Hierarchical Clusters [8]	✗	✗	✗	✓	✗	Not specified	Not Specified	✗

4.4. Summary

In this chapter, the research works conducted in the area of TD and TL were discussed. The review of these studies were conducted to assess the existing work with respect to the requirements and to demarcate the proposed approach in this thesis from the existing approaches. The contributions and drawbacks of each of these approaches are pointed out as per the requirements. To sum up, an improvement over the existing TDL methods is required as the comparison of these methods indicates that most of them do not satisfy the major/must-have requirements.

5. Design of the Topic Detection and Labeling (TDL) System

5.1. Overview

In this chapter, the design of the proposed Topic Detection and Labeling (TDL) system is presented in detail. As depicted in Figure 5.1, the proposed system architecture is composed of two major components: Topic Detection (TD) and Topic Labeling (TL). The TD part is dedicated for designing a component for identifying topics along with their boundaries from transcripts. The result of this component which is a list of topics is used later as input for the TL component in order to generate representative labels for the topics.

The major activities employed in TD are Sentence Embedding (SE), Customized Agglomerative Clustering (CAC), and Determining the Optimal Number of Topics (DONT). SE is used to represent sentences from a transcript with vectors of real numbers. The CAC process involves applying a Sentence Similarity (SS) and Cluster Linkage (CL) function to get hierarchical clusters of sentences of a given transcript. The SS method is used to determine the similarity between each data point used in the clustering process whereas the CL function is applied to decide which two clusters to merge at each level of the hierarchy.

On the other hand, the TL component of the system employs two main tasks; Label Generation (LG) and Label Ranking (LR). In the LG task, a set of semantically enriched terms are chosen as candidate labels with the involvement of an external knowledge base (specifically DBpedia) in order to represent a certain topic. Once a set of candidate labels are generated, the LR task will be undertaken to rank the labels according to their importance to the topic being interpreted.

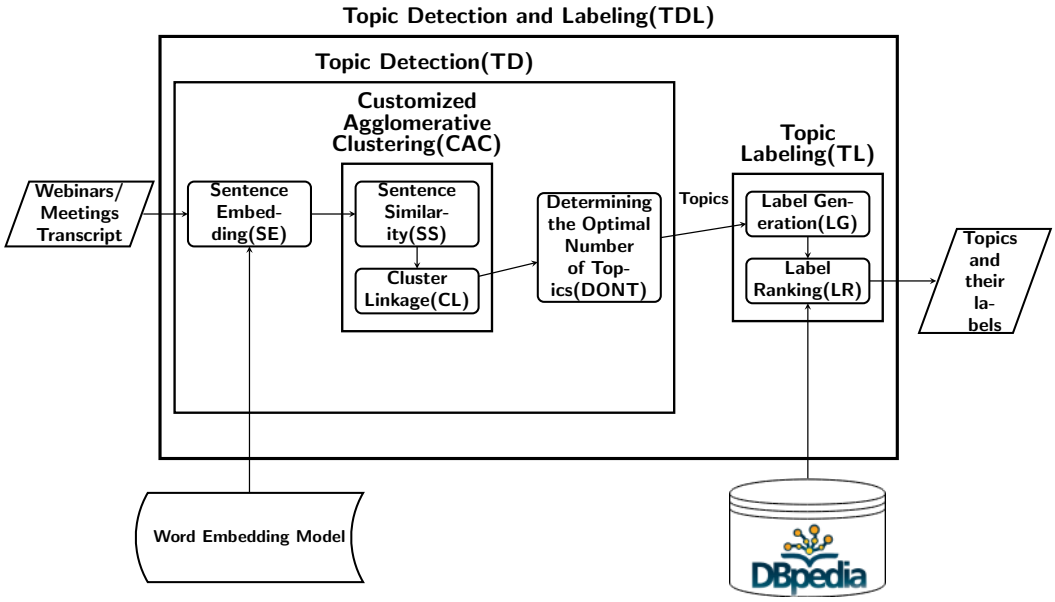


Figure 5.1.: The general framework of the proposed TDL system

5.2. Topic Detection (TD)

The TD component is dedicated to splitting audio/video textual transcriptions into parts or segments based on structural integrity and semantic coherence. Each of these segments is referred to as a topic containing one or more sentences. As depicted in Figure 5.1, the input given to the TD component is a textual transcript which is assumed to be divided in to sentences in advance. This transcript will be passed to the SE algorithm in order to generate embedding for the sentences in the transcript. Then, these sentence embeddings will be given as an input to the CAC algorithm which generates hierarchical clusters of the sentences from the transcript based on a SS measure and a CL function. Once the clustering hierarchy is generated, the DONT algorithm which cuts the hierarchy at some level will be applied to get an optimal number of topics that are present in the transcript. The TD component of the proposed system is mostly designed in order to satisfy the functional requirements FR-1 upto FR-3 discussed in Section 3.3.

5.2.1. Sentence Embedding (SE)

The first step in the TD process is to vectorize the sentences in the input transcript based on a word/text embedding model. It is required for the sentence embedding model to be capable enough to capture the semantics of the sentences of transcripts covering as variety of domain as possible. In Section 2.1.1, the description, advantages, and disadvantages of the different higher level text embedding methods, namely, VSM, Embedding Centroids, Specialized Averaging of Word Embeddings, Doc2Vec, Doc2VecC, Word Movers Distance (WMD), Skip-thoughts, Sent2Vec, and 'Embed, encode, attend,

predict' are discussed.

VSM is used to represent a text object (i.e. word, phrase, sentence, paragraph, and mostly a document) by giving values to each term in the vocabulary of the corpus. If a term occurs in the text, then it will be given a non zero value in the vector otherwise it will take 0. The values can be computed using different methods such as co-occurrence or TF-IDF. However, vsm creates sparse vectors with huge dimensions and it doesn't capture the actual semantics of the text as it works either by checking their existence in the text or applying co-occurrence matrix computation. Therefore, it is not a good option for representing text with the purpose of computing text similarity. As described in Section 2.1.1, the Doc2vec embedding more suits for feature extraction than embedding as it doesn't work well with distance functions whereas WMD can not be used for our approach as it is computationally expensive and is not recommended to use it for applications like clustering that requires computing similarity between each pair of sentences.

Due to the fact that there is shortage of transcripts in generic domain that can be used to train our own model for this thesis work, it is reasonable to use a pretrained word embedding model. Becasue of this reason and those discussed in Section 2.1.1, Doc2VecC, Skip-thoughts, Sent2Vec, and 'Embed, encode, attend, predict' methods, for which there are no pretrianed models on big and and domain generic corpora, can not be used for our approach. On the other hand, embedding centroids and specialized averaging of word embeddings are highly related approaches. The former computes the centroid of all of the words that exist in a text/sentence to get an embedding for the whole text while the later adds wights like TF-IDF to each word embeddings before applying averaging. Even if it is valuable to add weight to the words using TF-IDF, it can not be done for our system due to the fact that only one transcript is processed at a time not a corpus of transcripts (i.e. It is a single-document topic detection). So, in this thesis it makes sense to follow the averaging word embedding approach which uses some word embedding model. Therefore, the first step in this approach, averaging approach, is to choose an appropriate word embedding model that best fits for implementing the SE algorithm. It is required to note two things while choosing this model; It needs to have a pretrained model available for English language and It should work better than the other methods.

Word embedding is a kind of word vectorization which is a way of representing words with vectors. Vectorization can be done in many ways but the most common ones involve word context. However, vectors created based context are sparse and have thousands or milions of dimensions. In order to avoid these issues, they are required to be changed to a dense representation. The dense vectors are refereed to as embeddings. A word embedding model provides a dense representation of words with their relative meanings. The definition, the main concept, and the types of word embeddings are discussed in detail in Section 2.1.1. The two broad types of word embeddings described are namely,

frequency-based, and Prediction based Embeddings. Prediction based Embeddings are way effective than frequency-based embbedins as they are capable of predicting words. The common types of prediction based embeddings are word2vec, Glove, FastText, and WordRank. Word2Vec, Glove, and FastText are more popular embedding methods than WordRank as there are more works using those methods and there are pretrained models for each of them. However, we will consider only Word2vec's and Glove's pretrained models by Stanford and Google respectively because of the limit of time and resource of the thesis work.

Glove is based on a matrix factorization, i.e. by factorizing a word-context matrix where the context is neighboring words. On the other hand, word2vec is based on a neural network which takes word as an input and generates a context of the word as an output where the context it is neighboring words. It is written in Section 2.1.1 and 2.1.1 how word2Vec and Golve works respectively. For our proposed TD approach, we would like to use both Glove and Word2Vec and compare the result to decide which embedding method fits best. Therefore, the experiment will be conducted using both Word2Vec's and Glove's pretrained models in order to show which one works better with the TD component of the proposed system. Both Word2Vec and Glove pretrained models come with different dimensions. As stated in a research [14], the common dimensions for Word2Vec are in the interval 100 - 300 where 300 performs well with bigger vocabulary. Regarding Glove's pretrained models, as indicated in [17], 300 dimension gives better accuracy even with very huge vocabulary. Moreover, for both models, as the dimension increases the possibility to capture high dimensional properties also increases but the training will be extremely slow. For this reason, we will choose 300 as the dimension for both Word2Vec and Glove pretreined models for the experiment.

To this end, a SE algorithm is designed to generate a vector representation for a sentence by taking a colon-wise mean of the word vectors for the words occurring in the sentence. Figure 1 shows the pseudocode for the SE algorithm. This algorithm takes five values as inputs; a set of sentences S, word vectors from a pretrained word embedding model $\{v_w : w \in Vocab\}$, a dimension M of the word vectors, a parameter α with value between 0-1, and a parameter β with value between 0-1. The parameter α is used for deciding if a sentence can be properly vectorized or not; i.e. a sentence is properly vectorized if at least $(\alpha * 100)\%$ of the words in the sentence exist in the word embedding model vocabulary $Vocab$ otherwise a zero vector of dimension M will be used as vector for the sentence. On the other hand, the parameter β is used for deciding if a transcript with a set of sentences S has a proper vectorization or not; i.e. a transcript has a proper or acceptable vectorization if at least $(\beta * 100)\%$ of the sentences in S can be vectorized. If the result of the SE algorithm for a given transcript is not a proper vectorization, then the TD system returns a single topic containing all the sentences in the transcript without applying the CAC process. Based on the way both α and β are defined here it is clear to see that their values are percentages. However, in order to make the formula simpler in the algorithm and to just take small values as inputs, we let

the value to lie in the interval 0 to 1 where 0 and 1 indicate 0% and 100% respectively.

Algorithm 1 Sentence Embedding

Input:

S ▷ The input transcript as a set of sentences S
$\{v_w : w \in Vocab\}$ ▷ word vectors
α ▷ A parameter to check if a sentence can be vectorized
β ▷ A parameter to check if S has proper vectorization
M ▷ The dimension of the word vectors

Output:

Sentence Embedding $\{v_s : s \in S\}$
Proper_vectorizaion

procedure SENTENCEEMBEDDING(S, $\{v_w : w \in Vocab\}$, α, β)
  $Properly\_Unvectorized\_Sentences \leftarrow 0$
  $U \leftarrow ZeroVectors_M$ ▷ Create a vector of dimension M for those sentences with not enough words in Vocab
  for each $s \in S$ **do**
    $v_s \leftarrow \frac{\sum\{v_w | w \in s, w \in Vocab\} \cup \{U \mid w \in s \wedge w \notin Vocab\}}{|s|}$ ▷ Colon-wise mean of the word vectors
    $Available\_words \leftarrow |\{w | w \in s, w \in Vocab\}|$
    if $Available\_words < \alpha * |s|$ **then**
      $Properly\_Unvectorized\_Sentences \leftarrow Unvectorized\_Sentences + 1$
    end if
  end for
  if $|Properly\_Unvectorized\_Sentences| \leq \beta * |S|$ **then**
    $Proper\_vectorizaion \leftarrow true$
  else
    $Proper\_vectorizaion \leftarrow false$
  end if
end procedure

5.2.2. Customized Agglomerative Clustering (CAC)

So as to address research question 1.a, how to adapt an existing clustering algorithm in order to divide a document into segments or topics, stated in Section 1.3, it is required to choose an appropriate clustering technique which best fits the purpose of this research work. Thus, an extensive review of the available clustering techniques, with a description of their strong and weak sides, has been conducted and presented in Section 2.2.1. The comparisons of the most common algorithms (i.e. k means, DBSCAN, and Agglomerative clustering) from the three broad categories of clustering, namely, partitioning methods, density-based clustering, and hierarchical clustering is shown in Table 2.3. According to the functional requirement *FR-1 Topic detection* defined in Section 3.3, the system should be able to generate topics without prior information on

the number of topics that exist in a transcript. On the contrary, K-means clustering requires the number of topics to be given by the user in advance. For this reason, it is not possible to apply k-means clustering for the TD component of the proposed system.

In the case of DBSCAN, the algorithm basically requires two parameters to work, eps (the minimum distance between two points) and minPoints (the minimum number of points to form a dense region). These parameters are assigned values via parameter estimation. It is necessary to have prior knowledge about the data set to be clustered so that a good estimation of the parameters can be achieved. However, it is difficult to use DBSCAN for our approach as it is not possible to have previous knowledge about our transcript data set due to the fact that the transcripts are not domain specific and topics are detected separately for each transcript (i.e. the system employs a single-document topic detection process as mentioned in research question 1 defined in Section 1.3).

On the other hand, agglomerative clustering algorithm is a kind of hierarchical clustering which doesn't require the user to know the number of topics or set any parameter in advance. Moreover, with agglomerative clustering as an alternative to applying a method to determine the number of optimal topics, it is possible to visualize the clustering process using a dendrogram to let users decide on the number of topics themselves. But this is not applicable with k-means and DBSCAN as they do not employ a hierarchical approach. Thus, it is appropriate to choose the agglomerative clustering technique to better satisfy requirement FR-1. Agglomerative clustering works by initially assigning each data point to a singleton cluster and follows by merging together two most similar clusters at a time until either a single cluster with all the data points is left or a certain threshold (a stopping criterion) is met.

Note that there is another hierarchical clustering called divisive clustering its basic idea is the same as that of agglomerative clustering except it starts from a single cluster with all data points and continues by splitting one cluster in to two until every data point has its own cluster. However, agglomerative clustering is chosen instead of divisive clustering due to the reason that it has been implemented by different python libraries such as Scipy and Sci-kit learn and it is easier to reuse the code and customize it than develop it from scratch. Therefore, in this research, the agglomerative clustering method discussed in [78] is adapted for topic detection by customizing its euclidean distance function so as to let it consider the order between the sentences in the transcript. The two main tasks involved in the CAC process are namely SS and CL where the first computes distance between data points whereas the later merges clusters hierarchically. These tasks are discussed in detail in the following subsections.

Sentence Similarity (SS)

In text clustering process, once the sentences are represented in vectors, the first step to perform is to compute the distance between sentences using a chosen text similarity measure. There are various text similarity measures available. The literature conducted

on these measures is presented in Section 2.1.3 with the list of the most common distance functions for clustering is shown in Table 2.2. However, most applied ones with both flat and hierarchical clustering approaches are euclidean and cosine distance metrics. Euclidean is more appropriate for our approach due to the reason that our chosen cluster linkage function, Ward's method, performs well if euclidean is used as a distance function[1].

As discussed in Algorithm 1, to generate an embedding for a sentence the mean of the vectors of the words occurring in the sentence is used. This may lead to assigning an appropriate vector for a sentence if most or all of the words in the sentences do not exist in the vocabulary of the word embedding model that is used by the algorithm. This in turn may affect the result of the euclidean metric applied to get the semantic similarity of sentences. Therefore, there is a necessity to somehow handle the properly unrepresented or unvectorized sentences. Due to this reason, it is required to customize the distance metric used for computing similarity between sentences. One way to do so is to consider the order of the sentences in the transcript. It is persuasive enough to say that the probability that sentences that are neighbors discuss the same topic is much more higher than those that are written far apart.

Thus, the main purpose of the SS module is to customize the euclidean distance function in such a way that the order of the sentences in a transcript can also be taken into consideration as it also plays a great role in the relatedness of the sentences. The formula that customizes the euclidean metrics is shown in Equation 5.1. This formula, $Dist_{sen}$, is defined so as to compute the distance between two sentences by combining euclidean distance (Euc) with a newly defined distance function ($In\_transcript\_dist$). The formula used to compute Euc between two sentence vectors is shown in Equation 5.2. The $In\_transcript\_dist$ between two sentences s_1 and s_2 defines how close these sentences are based on the order they are written in the transcript. As shown in Equation 5.3, the $In\_transcript\_dist$ function works by counting the number of sentences that are in between the two sentences and normalizing the result by dividing it by $n-2$ so as to make the value lie in the range [0,1] where 0 indicates that the sentences are written next to each other (i.e. it is very likely that they are semantically similar) and 1 means one of the sentences is written in the beginning while the other is found at the end of the transcript (i.e. there is a high probability that they are semantically different from one another).

The parameters α and β in the distance function $Dist_{sen}$ are used to give weights for the Euc and $In\_transcript\_dist$ component functions respectively. Both α and β take values between 0 and 1 with $\alpha = 1 - \beta$ this is due to the reason that we need to keep the value of the main distance function between 0 and 1. The values for these parameters can be determined either randomly or estimated by referring to the number

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage`

of sentences that are properly vectorized. If many sentences from a transcript are not properly vectorized, then better to give more weight to the $In\_transcript\_dist$ function (i.e. $\beta > \alpha$). Otherwise, it is possible to give either equal weights which is the default value ($\beta = \alpha = 0.5$) or make one of them bigger by conducting experiments and choosing a better value.

The overall process of the SS method which is based on the $Dist_{sen}$ formula in Equation 5.1 is displayed in Algorithm 2. The algorithm takes a vectorized transcript (i.e. a set of sentence vectors) as an input and returns a condensed distance matrix which has the distance between each pair of sentence vectors in the transcript. The distance matrix is condensed so as to avoid saving repetitive values (i.e. since the distance between s_1 and s_2 is the same as the distance between s_2 and s_1 because of symmetry, it won't be saved twice). Besides repetitive values, the distance a sentence has with it self won't be saved due to the fact that the value is always zero and comparing a sentence with itself is not required for clustering anyway. Therefore, condensing a matrix in such a way helps to save space and also to save time while accessing the stored values.

$$Dist_{sen}(v_{s_i}, v_{s_j}, \alpha, \beta) = \alpha * Euc(v_{s_i}, v_{s_j}) + \beta * In\_transcript\_dist(v_{s_i}, v_{s_j}) \quad (5.1)$$

$$Euc(v_{s_i}, v_{s_j}) = \sqrt{(v_{s_{i1}}, v_{s_{j1}})^2 + (v_{s_{i2}}, v_{s_{j2}})^2 + ... + (v_{s_{in}}, v_{s_{jn}})^2} \quad (5.2)$$

$$In\_transcript\_dist(v_{s_i}, v_{s_j}) = \frac{(j - i - 1)}{n - 2} \quad (5.3)$$

where $s_1, ..., s_n \in S$ and v_{s_i} is an embedding for sentence s_i where $i \in (1, n)$.

Since SE for a given sentence is computed as a mean of its word vectors which are generated from a word embedding model with a vector space created in a way that words that are far from each other are semantically dissimilar and those close to each other are semantically similar, high euclidean distance (Euc) between sentence vectors created in such a manner implies less semantic similarity whereas small distance indicates high similarity. The $In\_transcript\_dist$ distance function is interpreted in a similar fashion as euclidean distance for the reason that mostly sentences which are near to each other are semantically similar than those far apart. Therefore, the $dist$ function which is the combination of these two distance functions (Euc and $In\_transcript\_dist$) is interpreted in the same manner i.e. high $Dist_{sen}$ value between two sentence vectors implies that the sentences are semantically less similar whereas small $Dist_{sen}$ value indicates that they are highly similar.

Algorithm 2 Sentence Similarity

Input:

$SentenceEmbedding \leftarrow \{v_s : s \in S\}$ ▷ The input transcript as a set of sentence embeddings

α ▷ A parameter to assign weight for the Euc function

β ▷ A parameter to assign weight for the $In\_transcript\_dist$ function

Output:

condensed distance matrix *M* as $\{d_{(v_{si},v_{sj})} : v_{si}, v_{sj} \in SentenceEmbedding\}$

1: **procedure** SENTENCESIMILARITY($SentenceEmbedding$, α, β)
2: $k \leftarrow 0$
3: **for each** i from 1 to $|SentenceEmbedding| - 1$ **do**
4: **for each** j from i+1 to $|SentenceEmbedding| - 1$ **do**
5: $M[k] \leftarrow Dist_{sen}(v_{s_i}, v_{s_j}, \alpha, \beta)$ ▷ refer to Equation 5.1, $\alpha = 1 - \beta$
6: $k \leftarrow k + 1$
7: **end for**
8: **end for**
9: **end procedure**

Cluster Linkage (CL)

In agglomerative hierarchical clustering, there are four well known cluster linkage functions namely single-link, complete-link, weighted average-link, and Wards method []. In single-link clustering, the distance between two clusters is the shortest distance between them. The problem with this clustering method is its sensitivity to outliers and incapability in dealing with severe differences in the density of clusters. As defined in requirement FR-1, the proposed system should put together those sentences which are semantically related together to make a topic. However, since the single-link clustering takes into consideration only the two sentences that make the clusters come closer without looking at the rest of the sentences in the cluster, it is not appropriate method to use for the clustering process in the proposed system.

Complete-link clustering considers the distance between two clusters to be the largest distance between them. The difficulty with this clustering approach is its high tendency of breaking large clusters and works less efficiently with convex shaped clusters. This indicates applying complete-link for creating clusters for transcripts with the purpose of detecting topics may be put at risk. This is due to the reason that it is possible to have webinar transcripts that discuss just one or two topics but contain a lot of sentences. In this case, if we apply the complete-link clustering it may break the cluster to create more irrelevant small topics which will lead the system to not satisfy requirement FR-2.

On the other hand, weighted average-link clustering takes the average distance between two clusters as the distance between them. This clustering method is a compromise between single and complete linkage measures which makes it a better choice than them.

Despite that, it has its own drawback which is its sensitivity to the shape and size of clusters. It performs differently with different shape and size of clusters which implies that it is not possible to get a stable result with this approach.

The other common cluster linkage method is Ward's method which chooses two clusters to merge at each iteration/level of the hierarchy based on an optimal value of an objective function. This objective function is referred to as *error sum of squares* and it is used to check how costly it is to merge two clusters. There is no best way to choose the right linkage measure for clustering. In our research, to construct the hierarchical clustering, ideally both weighted average-link clustering and Ward's method can be applied. The appropriate way to pick one among these two linkage measures is to conduct an experiment and choose the one with a better result. However, since there is a limit of time for the thesis work to try out both methods, it is better to compare them and choose one. Thus, the Ward's method is chosen for our proposed system due to the reason that it doesn't have any loose ends as compared to the weighted average linkage metrics (i.e. the probability that clusters with only one or a few data points is very less). The Ward's method [78] defines the distance between two clusters based on the cost obtained using the objective function shown in Equation 5.4.

$$Dist(C_u, C_k) = \begin{cases} Dist_{sen}(C_u, C_k) & \text{if } |C_u| = 1 \ \& \ |C_k| = 1 \\ \sqrt{\frac{|C_k| + |C_i|}{T} \times Dist(C_k, C_i) + \frac{|C_k| + |C_j|}{T} \times Dist(C_k, C_j) - \frac{|C_k|}{T} \times Dist(C_i, C_j)} & \text{otherwise} \end{cases} \tag{5.4}$$

where C_i and C_j are the clusters considered for merging, $C_u = C_i \cup C_j$, C_k is a cluster in the current forest different from C_i and C_j, and T is $|C_k| + |C_i| + |C_j|$.

The CAC Algorithm

The CAC process combining both the SS and CL is shown in Algorithm 3. It constructs the clustering hierarchy iteratively by merging two sentences (data points) at a time. It starts by creating a forest with $|S|$ number of singleton clusters (i.e. each sentence in the transcript making their own cluster). Then, it proceeds to constructing the hierarchy by first merging the two sentences with the lowest distance value in the condensed distance matrix. After doing the first merge, the recursive linkage function defined in Equation 5.4 will be applied to perform the rest of the merges. At every iteration, except the first one, this linkage function is used to compute the distance between the newly merged cluster and the other clusters in the forest. Then, the distance matrix will be updated with the newly computed distance value and the individual clusters used for merging will be removed from the forest and the newly merged cluster will be added instead. This

process continues until exactly one cluster with all the sentences making the root of the hierarchy is left in the forest.

Algorithm 3 CAC

Input: S ▷ a transcript as a set of sentences S
Output: $Tree_{Topics}$
1: $M \leftarrow$ SentenceSimilarity($SentenceEmbedding$, α, β) ▷ condensed distance matrix
2: $Forest \leftarrow \{\{S_i\} \in S\}$ ▷ Take all data points as initial singleton clusters
3: $Tree_{Topics} \leftarrow Forest$ ▷ All sentences at the leaf of the tree(hierarchy)
4: $C_u \leftarrow C_i \cup C_j$ where Dist(C_i, C_j) is the smallest ▷ Merge the two closest data points in the forest
5: Update $Tree_{Topics}$ with C_u ▷ add another level to the hierarchy
6: $Forest \setminus C_i$ and $Forest \setminus C_j$ ▷ remove used clusters
7: Compute $Dist(C_u, C_k)$, for all $C_k \in Forest$
8: Update M with $Dist(C_u, C_k)$
9: Repeat Step 4 - 8 Until $Forest = \emptyset$

5.2.3. Determining the Optimal Number of Topics (DONT)

One of the challenging tasks in TD is finding the optimal number of topics from a transcript. In most TD or topic modeling applications, the desired number of topics has to be given in prior which is inefficient as it is difficult for users to know how many topics exist in a transcript. Since the TD system proposed in this thesis work is a clustering based method, it is possible to address this issue by making use of algorithms which are designed to capture the optimal number of clusters. As mentioned in Section 2.2.2, there are three main automatic approaches that can be used for DONT purpose namely elbow method, average silhouette method, and gap statistic method.

There is no algorithm for DONT which works best in all scenarios; each one of these algorithms has its own benefits and drawbacks. The advantages and disadvantages of these methods are presented in Table 2.4. The key problems of the gap statistics method are the decrease in performance as the clusters separate and the necessity of predetermining the reference distribution by the data analyst. As stated in requirement FR-2 the system should able to determine the optimal number of clusters with no or less involvement from users. However, in gap statistics method the data analyst has to choose a reference distribution and there is no easy and effective way to do so. Therefore, it is not appropriate to apply this method to address the DONT issue.

Average silhouette and Elbow methods are two closely related approaches as mentioned in Section 2.2.2. Despite that, average silhouette method suits more for clustering which uses average linkage measure due to the fact that both average silhouette method and average linkage make use of average proximities [79]. However, as mentioned previously

for our proposed approach Ward's linkage method is used so we should not combine it with average silhouette method to get optimal number of clusters. Rather, we would like to employ a variant of elbow method which is highly related to average silhouette method [80] but doesn't use average proximities.

Note that the variant of the elbow method used in this thesis has its own problems. As it can be seen from the way the formula is defined there is no way that this algorithm returns a single cluster/topic containing all data points or a singleton clusters (each sentence with their own unique topic) because it is required to have a left and right value to compute acceleration. However, despite its incapability of handling the single cluster cases, it is still the most common elbow variant for hierarchical clustering algorithms because of its simplicity and high speed. Moreover, in the case of webinars/meetings, singleton clusters/topics or a transcript with just one topic is not common. So, this would not be much of a concern for our proposed approach. In addition, as mentioned in Table 2.4, it is a fast and simple to compute algorithm which is an advantage for the proposed system.

The algorithm of the employed elbow method works by looking at the largest acceleration in jump sizes or distance growth. Given a set of distances generated by the CAC algorithm, this elbow method chooses the optimal number of clusters by applying Algorithm 4. The first main step in this algorithm is to compute the acceleration of the distance values using the formula depicted in Equation 5.5. Once the acceleration is computed, the number of clusters will be determined by looking at the largest acceleration value and getting the iteration at which this value is obtained ($argmax(Acc)$ in Algorithm 3) and adding 2 (because acceleration is defined by computing twice - recursively) and finally deducting the result from n (the total number of distances considered for merging - #iterations).

$$Acceleration(d_1, ..., d_n) = \bigcup_{i=3}^{n}(d_i - 2d_{i-1} + d_{i-2}) \tag{5.5}$$

where *n* is the number of merges or distance values.

Algorithm 4 DONT algorithm using a variant of the elbow method

Input:

$Distances\ \{d_1, d_2, ..., d_n\}$ ▷ where n is the number of total merges/iterations and d_i is the distance between the clusters merged at i^{th} iteration

$Cutting\_option = "first"$ ▷ takes the values "first" or "second" representing the first and the second cutting points respectively, with "first" as default

Output: $(Cut\_point, Num\_clust) \mid Cut\_point \in Distances \wedge Num\_clust \in (1, n)$▷ a distance where to cut the dedrogram and the number of clusters found at that level

1: $Acc = Acceleration(d_1, ..., d_n)$ ▷ Refer to the formula in Equation 5.5
2: $MaxInd \leftarrow argMax(Acc)$ ▷ The index of the maximum acceleration value where index starts from 1 and goes up to n
3: **if** $Cutting\_option == "first"$ **then**
4: $Cut\_point \leftarrow Distances(MaxInd + 1)$ ▷ The distance with the high jump
5: $Num\_clust = n - MaxInd$
6: **else if** $Cutting\_option == "second"$ **then**
7: $Distances[maxInd] \leftarrow 0$ ▷ Replace the maximum acceleration value with 0 in order to take the next highest value
8: $Cut\_point \leftarrow Distances(MaxInd + 1)$▷ The distance with the next high jump
9: $Num\_clust = n - MaxInd$
10: **end if**

5.2.4. Example Work Through

In order to capture the idea discussed above for the TD component of the system, a simple example is presented as follows. The following textual transcript with 11 sentences is created by taking sentences from other 3 transcripts discussing different topics. Therefore, this transcript has 3 topics each identified with blue, cyan, and red colors. The first topic has 5 sentences from sentence 0 upto 4 colored with blue, the second topic has the next 3 sentences shown in cyan color and the last has the rest 3 sentences in red color.

I work at an artificial intelligence research lab.
We're trying to create technology that you'll want to interact with in the far future.
Not just six months from now, but try years and decades from now.
And we're taking a moonshot that we'll want to be interacting with computers in deeply emotional ways.
So in order to do that, the technology has to be just as much human as it is artificial.
I'm here to talk about congestion, namely road congestion.
Road congestion is a pervasive phenomenon.
It exists in basically all of the cities all around the world, which is a little bit surprising when you think about it.
But as we started to understand the virus more and how it was transmitted, we realized that that risk had increased its territory.
The highly profiled case of Ryan White in 1985, who was a 13-year-old hemophiliac who

had contracted HIV from a contaminated blood treatment, and this marked the most profound shift in America's perception of HIV.
No longer was it restricted to these dark corners of society, to queers and drug users, but now it was affecting people that society deemed worthy of their empathy, to children.

Given this transcript, first the vectors for each of its sentences will be generated using the SE algorithm. Then, the CAC algorithm will be applied to the sentence vectors. It starts by computing a condensed distance matrix using the SS algorithm with input $\alpha = 0.6$ and $\beta = 0.4$ (i.e. these values are chosen after trying different other values) and continues to apply the CL procedure to construct clustering hierarchy. This hierarchy is shown as a dendrogram in Figure 5.2. The x-axis shows the 11 sentences of the transcript by their index value where 0 is to mean the first sentence and 10 the last sentence. The dendrogram indicates that during the first iteration or merge sentence 7 and sentence 8 are merged because the distance between these sentence, 0.172, is the smallest value in the distance matrix. At the second iteration, sentences 9 and 10 are merged and it continues like that until all the sentences are grouped in one cluster.

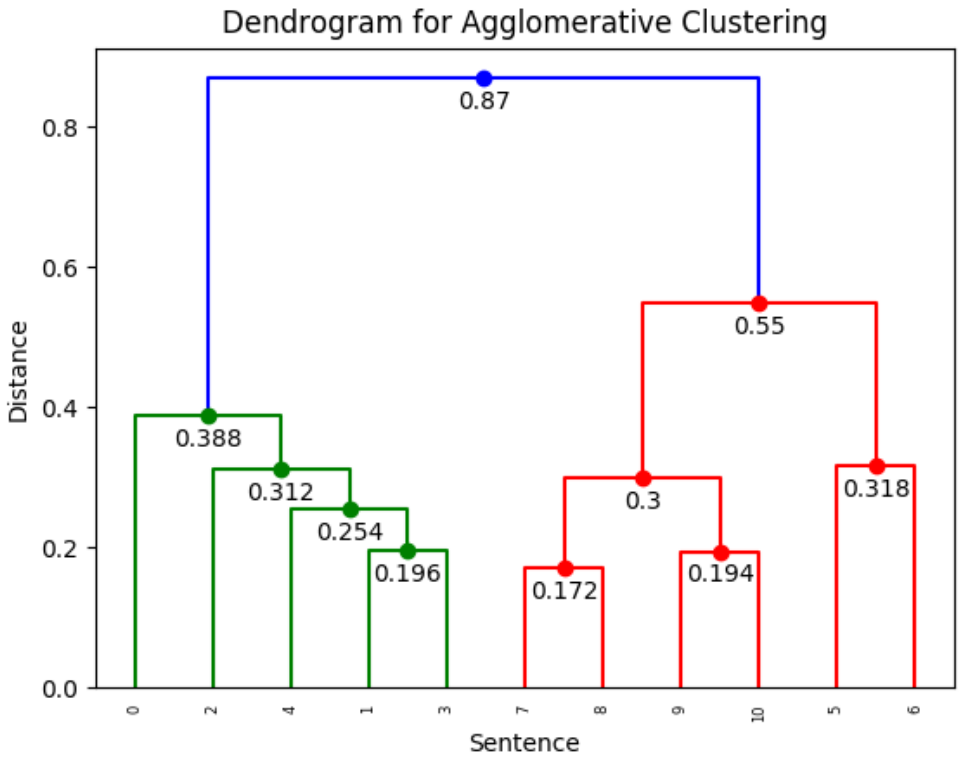


Figure 5.2.: Example dendrogram showing hierarchical clustering

Once the dendrogram is created, the optimal number of topics will be determined us-

ing the DONT algorithm which takes the distances chosen at each iteration/merge (i.e. {0.17177548, 0.19435426, 0.19584274, 0.25428059, 0.29953344, 0.31202124, 0.31774081, 0.38787137, 0.54986513, 0.86985459} which is shown as a label on the dendrogram in Figure 5.2) and produces two alternative values for the optimal number of clusters with their respective cut points or distances. It first computes the acceleration values which is { -0.0210903, 0.05694937, -0.013185, -0.03276505, -0.00676823, 0.06441099, 0.0918632, 0.1579957}. Then, the cut_point or the distance at which the dendrogram will be cut is calculated by taking the index of the maximum acceleration value 0.1579957 (i.e. 8) and adding 1 to it to get the index value of the required cut_point from the distance input which is the 9th value 0.54986513. Finally, the optimal number of clusters 2 is obtained by deducting the index of the maximum acceleration value (MaxInd=8) from the total number of merges $n = 10$. Moreover, as an alternative, another optimal number of clusters *3* at a cut point or distance 0.38787137 is obtained by taking the second maximum acceleration value (0.0918632) and doing the calculation analogously as the way the first optimal number of clusters is obtained.

The result of this process is depicted in Figure 5.3. Note that the acceleration value and the distance values are reversed in order to draw an elbow shaped line on the graph - just to keep the actual meaning of the word elbow. The graph on the left shows the acceleration against the number of clusters in green color and the distance value chosen during each iteration/merge in blue color. The graph on the right is used to visualize how the dendrogram is cut by putting a horizontal straight bar at the second cut point 0.38787137 computed above. Besides, it is possible to see that there are 3 clusters at this cut point by counting the number of vertical lines passing through this horizontal bar.

Looking at the result of the entire TD process for the given transcript, the system generated the correct number of topics (i.e. 3). The first topic is composed of sentences 0 to 4, the second topic has sentences 5 and 6 the third topic consists of sentences 7 to 10. The boundary of the first topic is identified correctly as sentence 0 to 4 but the third topic has one extra sentence (sentence 7) which was supposed to be part of the second topic. Even though sentence 7 has been identified as part of a wrong topic, when you look at the actual sentence it actually is a bit difficult to assign this sentence to any cluster/topic as it doesn't clearly share any concept with any of them. In the ground truth, sentence 7 is considered as part of topic 2 because it was generated from the transcript where the rest of the sentences of topic 2 come from.

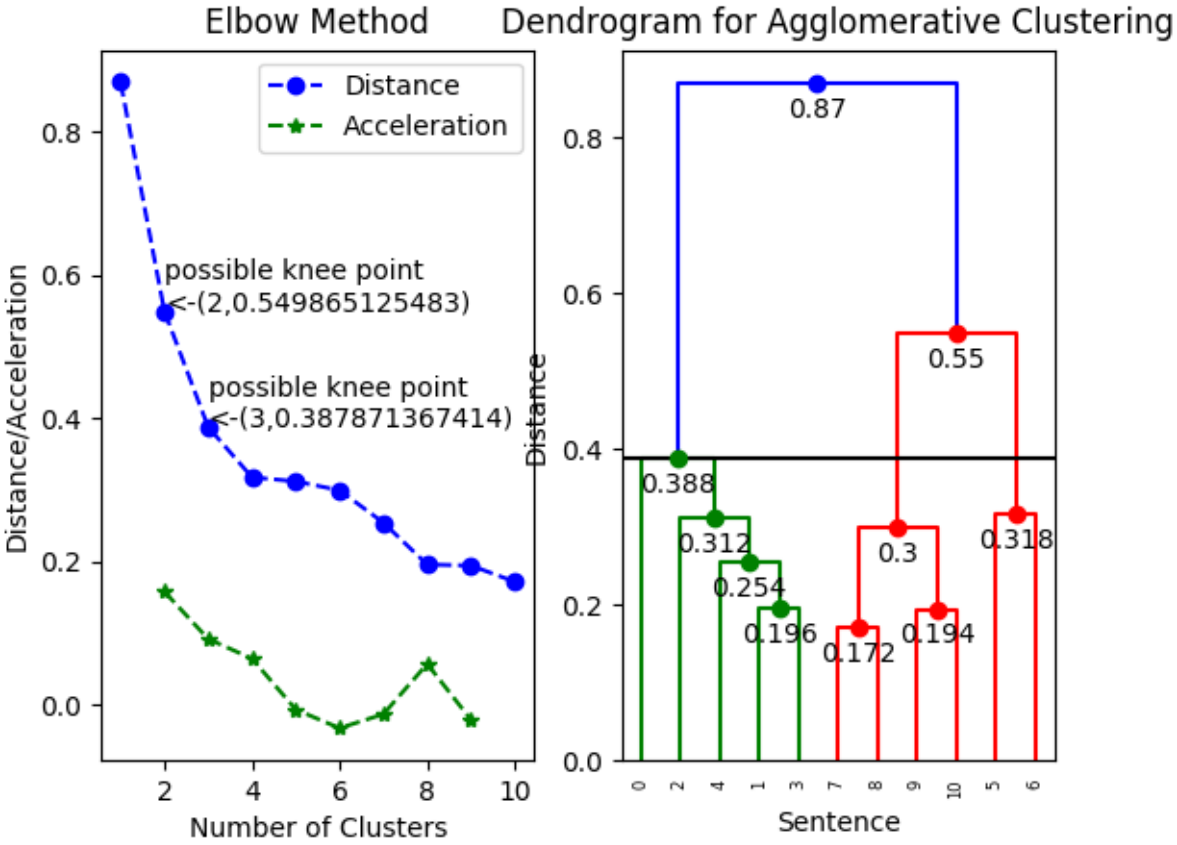


Figure 5.3.: Elbow and Dendrogram

5.3. Topic Labeling (TL)

TL is the process of interpreting topics using representative terms or phrases based on their semantics. Different researches [4, 5, 6, 7, 8] have been conducted so far to propose methods for generating labels for topics as discussed in Section 4.2. The drawbacks of these methods are presented in detail in Table 4.1. Since the research work entitled Automatic Labeling of Topic Models [4] is entirely dependent on Wikipedia, emerging terms for which there is no wikipage can not be used as labels. This may lead to making the system not be able to generate labels for emerging topics. The research which uses summarization for generating topics [5] doesn't check if the summary generated for a topic is coherent or not. If coherence is not checked then, unrelated labels can be generated as options. The ontology-based topic labeling system [6] is dependent on DBpedia to find connection among concepts of a topic. A topic labeling system [7] which generates labels for topics identified using LDA, uses the words in the topic

which impacts the system to not capture the actual underlying meaning of the topics. The labeling system for hierarchical clusters [8] doesn't work well for clusters with few number of data points because it is difficult to find labels unique to such clusters.

In this research, a TL system is designed and implemented so as to address these issues with the existing systems. The first step inTL is choosing the candidate labels from the topic sentences using a LG algorithm. The generated set of candidate terms will be ranked using a ranking method LR which integrates different properties together. The label generation and ranking tasks are discussed in detail in the following subsections.

5.3.1. Label Generation (LG)

The LG task is dedicated to selecting candidate labels from the set of terms occurring in the sentences constituting the topic. In this research, topic labels are terms or phrases that capture the intended interpretation of the topics and have user understandable meaning. An existing method uses n-grams to generate the candidate labels but it removes those n-grams that do not exist in Wikipedia [4]. On the contrary, in our method, we do not enforce the labels to exist in DBpedia rather we make sure only nouns and adjectives with proper meaning are used to construct the labels. A label has a proper meaning if it is just a one word label which is a noun or a multi-word label starting with either an adjective or a noun and followed by nouns. The initial set of candidate labels is created by combining consecutive words provided that they have a proper meaning. Then, the candidate labels set will be extended by clunking the multi-word labels in to more single or multi-word meaningful labels. Checking if a label is meaningful or not helps in avoiding having labels which have no relevance in interpreting topics. For example, giving the adjective "little" alone as a label to a topic doesn't make much sense.

Candidate labels are not filtered out for not existing in DBpedia but ranked instead so that we won't lose labels which are unknown to DBpedia but capable enough to capture the meaning of a topic. For example, the term ''artificial intelligence research lab'' is a meaningful term to be taken as a label for a topic but it doesn't occur in DBpedia and it is made of adjectives and nouns. So, it is better to keep those kind of terms as candidate labels than remove them. To this end, the LG task is designed as shown in Algorithm 5 to extract neighboring nouns from sentences of a topic as candidate labels for the topic.

Algorithm 5 The LG Algorithm

Input: $Cluster \leftarrow \{s|s \in S\}$ ▷ A cluster containing one or more sentences
Output: $CandidateLabels$ ▷ Selected candidate labels for the given cluster
1: $All\_words \leftarrow$ $words\ in\ Cluster$ ▷ Get all words from the sentences in the cluster
2: $Tagged \leftarrow \{(word, pos) \mid word \in All\_words\}$ ▷ Tag each word with their part of speech
3: $CandidateLabels \leftarrow \emptyset$
4: $label \leftarrow$ ""
5: **for each** $(word, pos) \in Tagged$ **do**
6: **if** $pos =$ "NN" or $pos =$ "NNP" or $pos =$ "NNS" or $pos =$ "$NNPS$" or $pos =$ "JJ" **then** ▷ Check if the word is a noun or adjective
7: $label \leftarrow label +$ " " $+ word$ ▷ Concatenate neighboring nouns/adjectives
8: $Continue$
9: **end if**
10: $CandidateLabels \leftarrow CandidateLabels \cup \{label\}\ if\ label\ has\ proper\ meaning$
11: $Chunks \leftarrow ngrams(label)$ ▷ get all ngrams out of label
12: $CandidateLabels \leftarrow CandidateLabels \cup \{chunk\}\ if\ chunk\ has\ proper\ meaning\ for\ chunk \in Chunks$
13: $label \leftarrow$ ""
14: **end for**

5.3.2. Label Ranking (LR)

The candidate terms generated using the LG algorithm needs to be ranked according to their capability in interpreting the given topic. Once the labels are ranked, the top most relevant ones will be chosen and returned to the user. A relevance score is computed for each candidate label of a given topic and the labels will be ranked based on their relevance score. Different properties are considered when computing the relevance of a label for a certain topic; namely, Popularity, Term Specificity, Topic Relevance, and Coherence properties. Each of these properties are discussed in detail as follows:

- **Popularity**: The popularity of the candidate term in a chosen external knowledge base (specifically DBpedia) is considered for validating the meaningfulness of the term. Since candidate label terms are generated by putting together neighboring nouns, it is required to make sure that the terms are not nonsense and have some semantics. Popularity of a candidate label l in DBpedia is measured by looking at the resource r which is labeled with l and counting the number of other resources (other terms in DBpedia) which are connected to r with an outgoing or incoming edges. Once the popularity of a label is counted, the result will be normalized so as to let the value lie between 0 and 1. The normalization is done after counting the popularity of all the labels. Let t be a topic, L_t the candidate label set of t, C_l the result of the popularity of the candidate label $l \in L_t$. The normalized popularity value of l with respect to the topic t is computed by the formula shown

in Equation 5.6.

$$Popularity(l, L_t) = \frac{C_l - min}{max - min} \tag{5.6}$$

where min is the C_l value for the least popular $l \in L_t$ and max is the C_l for the most popular label l.

- **Term Specificity**: Term specificity is defined as how specific the term is regarding its meaning. The specificity of a term is measured by counting the number of words it contains. A term with more words is more specific than a term with less words. Given the frequency of a label *l* in the topic *t* as $Count(l, t)$, the normalized term specificity value of *l* in L_t in the range between 0 and 1 using the formula in Equation 5.7.

$$Term\_Specifity(l, L_t) = \frac{Count(l, t) - min}{max - min} \tag{5.7}$$

where min is the $Count(l, t)$ value for the least frequent $l \in L_t$ and max is the $Count(l, t)$ for the most frequent l.

- **Topic Specificity**: A candidate label is more relevant to a topic if it is unique to the topic (i.e. if it is not used as a label for other topics in the transcript). In order to compute topic specificity, it is required to check the uniqueness or importance of each candidate label to the given topic using an appropriate method. Term frequency and TF-IDF are the common methods to check the expressiveness or relevance of terms to a given text. Term frequency works by counting the occurrence of a term in the text. This doesn't actually tell whether the term is unique to this specific text because the term can also be equally or more frequent in other texts in the corpus as well. However, TF-IDF is a more valid method to check uniqueness since it considers the whole text corpus while computing relevance of a term to a given text. Therefore, in this research the Term Frequency/Inverse Document Frequency (TF-IDF) method is employed to compute topic specificity. The TF-IDF formula stated in Equation 2.1 is used by replacing the concept of documents with sets of candidate labels and corpus with a transcript. Given a set of topics t in a transcript T, a set of candidate labels L_t of t, and $l \in L_t$ the TF-IDF formula is rewritten as in Equation 5.8 for the sake of clarity.

$$Topic\_Specificity(l, L_t) = TF(l, L_t) * IDF(L_t) \tag{5.8}$$

where

$$TF(l, L_t) = \frac{(Number\ of\ times\ label\ l\ appears\ in\ L_t)}{(Number\ of\ candidate\ labels\ in\ L_t)}$$

and

$$IDF(L_t) = log\frac{(number\ of\ topics\ in\ T)}{(number\ of\ topics\ in\ T\ with\ a\ candidate\ label\ l)}$$

- **Coherence**: The coherence property is used to check how semantically connected the candidate labels are with each other. The idea is inspired by Gensim's *'doesn't match'* method which takes out an outlier word from a list of words [81]. This method works by computing a centeroid of the words by taking the mean the of the vectors of all of the words and then calculate the cosine distance from the centeroid to each of the words and take the word with highest cosine distance as an outlier. Thus, based on this method the coherence value of each of the candidate labels is calculated iteratively. First, the outlier candidate label will be taken out from the original candidate set and then the algorithm will be executed against the rest of the candidates and will take out the next outlier. This will continue until two elements are left in the set from which one will be picked randomly for the sake of creating a score difference between them. The first-computed outlier will be given the lowest coherence score (1) and the second-computed the second lowest (2) and similarly until the last picked outlier which will take the highest value(the total number of candidate labels). Finally, the coherence score values will be normalized by dividing each value by the total number of candidates to have the values lie between the range 0 and 1.

  Note that sometimes there can be labels which do not exist in the word embedding model's vocabulary. In such cases, the labels that do not exist in the vocabulary are assigned normalized coherence value of 0 which the lowest value indicating that the label is not important to the topic. It is done like this because if a label is not found in such a huge vocabulary then the probability that it is meaningful, as it is created by using an automated algorithm - Algorithm 5, is very less. Thus, the algorithm shown in Algorithm 6, is used to compute coherence value to those labels that exist in the vocabulary.

The formula in Equation 5.9 is used to compute the relevance score of a candidate label l in a candidate label set L_t generated for a given topic t by combing together the popularity, term specificity, topic specificity, and coherence properties discussed above. Since all properties may not be equally important for computing the relevance score, a weight value is assigned to each of them using the parameters α, β, δ, and σ as shown in the formula.

$$Relevance\_Score(l,t) = \begin{array}{c} \alpha Popularity(l, L_t) + \beta Term\_Specifity(l) + \\ \delta Topic\_Specifity(l, L_t) + \sigma Coherence(l, L_t) \end{array} \tag{5.9}$$

where $\alpha + \beta + \delta + \sigma = 1$

5.3.3. Example Work Through

In the current chapter so far, a detailed discussion on the concept of the proposed TDL system has been given with an example in Section 5.2.4 showing how the TD component works. In this section, the same example will be further extended to show how the TL component creates labels for the topics generated by the system for the example

Algorithm 6 Coherence

Input:
L_t ▷ The candidate label set for a given topic t
Output:
Coherence
procedure COHERENCE(L_t)
  $V_{L_t} \leftarrow \{V_l \mid l \in L_t\}$ ▷ Vectors of all candidate labels
  $N \leftarrow |L_t|$ ▷ Total number of candidate labels
  for each i from 1 to N **do**
    $Centroid \leftarrow Mean(V_{L_t})$ ▷ The mean of all the vectors as the center for all
    $CosDist \leftarrow \{Cosine(V_l, Centroid) \mid V_l \in V_{L_t}\}$ ▷ The cosine distance between each label and the centroid
    $Cohe \leftarrow \frac{i}{N}$ ▷ The coherent score
    $Coherence \leftarrow Coherence + (l : Cohe)\ where\ l \in L_t \wedge Cosine(V_l) = Max(CosDist)$ ▷ Add a label and its coherence result
    $V_{L_t} \leftarrow V_{L_t} \setminus V_l$ ▷ Remove the outlier label
  end for
end procedure

transcript. First the labeling algorithm takes as an input the three topics shown in 5.1 from the result of the TD process presented in Section 5.2.4 and generates candidate labels for each of these topics by using Algorithm 5. Then, for each candidate label the values of the popularity, term specificity, topic specificity, and coherence properties are computed using the formulas defined previously. Finally, the relevance score is computed using the formula in Equation 5.9 with $\alpha = 0.2$, $\beta = 0.5$, $\delta = 0.2$, $\sigma = 0.1$. The ranking of the labels are done using their relevance score i.e. a label with high score being highly relevant. The top 10 candidates generated for the three topics along with their relevance score are presented in Table 5.2.

Table 5.1.: The three topics generated for the transcript given as an example

Topic ID	**Topic**
1	I work at an artificial intelligence research lab. Were trying to create technology that youll want to interact with in the far future. Not just six months from now, but try years and decades from now. And were taking a moonshot that well want to be interacting with computers in deeply emotional ways. So in order to do that, the technology has to be just as much human as it is artificial.
2	I'm here to talk about congestion, namely road congestion. Road congestion is a pervasive phenomenon.
3	It exists in basically all of the cities all around the world, which is a little bit surprising when you think about it. But as we started to understand the virus more and how it was transmitted, we realized that that risk had increased its territory. The highly profiled case of Ryan White in 1985, who was a 13-year-old hemophiliac who had contracted HIV from a contaminated blood treatment, and this marked the most profound shift in Americas perception of HIV. No longer was it restricted to these dark corners of society, to queers and drug users, but now it was affecting people that society deemed worthy of their empathy, to children.

Table 5.2.: Top 10 labels generated by the TL algorithm for the given example

Topic ID	Rank	Label	Relevance Score
1	1	Artificial intelligence research lab	0.41737490345144734
	2	Technology	0.38474980690289473
	3	Year	0.33704871659228441
	4	Artificial intelligence	0.28988315427652983
	5	Research	0.28919008496959919
	6	Computer	0.28519668562966516
	7	Artificial intelligence research	0.28404157011811398
	8	Intelligence research lab	0.28404157011811398
	9	Human	0.25456962292339458
	10	Month	0.24278744470557279
2	1	Road congestion	0.43604426130690987
	2	Pervasive phenomenon	0.41798717402152646
	3	Little bit	0.41798717402152646
	4	City	0.28000000000000003
	5	Bit	0.2244017159804087
	6	Phenomenon	0.18623693915665568
	7	World	0.15498920587575726
	8	Road	0.10927927911294427
	9	Congestion	0.054385371226712512
2	1	Contaminated blood treatment	0.41484837984370276
	2	Drug	0.34468930536792547
	3	Blood	0.29990260977862682
	4	White	0.26484837984370274
	5	Risk	0.23788525619945111
	6	Case	0.23696696263473671
	7	Society	0.23166349866065203
	8	Treatment	0.22797202409532963
	9	Drug user	0.22045213978585748
	10	Perception	0.21853601541854006

5.4. Summary

In this chapter, the main contribution of the research work which is a TDL system composed of a TD and TL components is presented. The general architecture or work flow of the system is depicted in Figure 5.1 showing the interaction among these two main components of the system. The TD component is developed by adapting an agglomerative clustering technique which is customized by modifying the definition of the distance measure used to compute distance between data points or sentences. Moreover, being

one of the research questions, the automatic way of determining the optimal number of clusters is designed and implemented by adopting the elbow method which is chosen for its better quality than the other available methods.

The TL component of the system is designed by taking into consideration the valuable properties of candidate labels namely, popularity, term specificity, topic specificity, and coherence. Each of these properties play a vital role in determining the relevance of a candidate label to a topic. The relevance of a label to a topic is determined by its relevance score which is computed by summing up the weighted values of those 5 properties for the given label. In order to show how the proposed system works, the algorithms developed or customized for every task involved in the process are given. Moreover, an example which illustrates how topics can be detected and then assigned labels is provided for better understanding of the algorithms of the proposed system.

6. Implementation

In chapter 5, the architecture and design of the proposed Topic Detection and Labeling (TDL) system has been presented in detail covering the concept that is behind the main components of the system. In the current chapter, the implementation details of the system will be discussed. In general, the python[1] programming language is used to implement the system architecture, to prepare the ground truth, and to perform the evaluation process. Python is a high-level, general-purpose, dynamically-typed, interpreted programming language which is known for scientific computing, data analysis, AI, and web development. Python is chosen for the implementation of the proposed system in this thesis because it has extensive libraries for data science such as numpy[2], scipy[3], scikit-learn [4], matplotlib[5], gensim[6], and nltk[7].

Besides python being the programming language for the entire system , it is necessary to describe the implementation details for each component of the system. Therefore, in the following sections, the libraries used for the implementation of the main tasks of the components are described.

6.1. Topic Detection (TD) Implementation

Various python libraries have been explored to implement the TD component of the system. As discussed in the previous chapter, the tasks, namley, Sentence Embedding (SE), Sentence Similarity (SS), Cluster Linkage (CL), and Determining the Optimal Number of Topics (DONT), are involved in the process of detecting topics. Therefore,

[1] https://www.python.org/download/releases/2.7/
[2] http://www.numpy.org/
[3] https://www.scipy.org/
[4] http://scikit-learn.org/stable/
[5] https://matplotlib.org
[6] https://radimrehurek.com/gensim/
[7] https://www.nltk.org/

in this section, the implementation details of each of these tasks and the TD evaluation module will be given.

- **Sentence Embedding (SE)**: In order to implement the SE algorithm, Algorithm 1, designed in the previous chapter, the gensim library is used to load the pretrained word embedding model. Moreover, the numpy python library is imported to work with arrays and compute mean of vectors.
- **Sentence Similarity (SS)**: When implementing Algorithm 2, Scipy's $pdist$[8] method from scipy.spatial.distance package is used to compute euclidean distance. The euclidean distance is one component of the distance function defined in Equation 5.1 to find distance between sentences.
- **Cluster Linkage (CL)**:The ward cluster linkage function depicted in Equation 5.4, as described more in Algorithm 3, is implemented using the $linkage$ method from $scipy.cluster.hierarchy$ package. The result of the clustering process, the cluster/topic hierarchies, is depicted as a dendrogram using the $dendrogram$ library from $scipy.cluster.hierarchy$ package.
- **Determining the Optimal Number of Topics (DONT)**: The python libraries matplotlib and numpy are used to implement Algorithm 4 which is used to determine the cutting point of the dendrogram generated by the Customized Agglomerative Clustering (CAC) algorithm, Algorithm 3, to get the optimal number of topics.
- **Evaluation Metrics**: The numpy library and the $v\_measure\_score$[9] method from sklearn.metrics package of the scikit-learn library are used to implement the automated evaluation of the TD system since it saves time to use already available methods from such libraries than write code from scratch.

6.2. Topic Labeling (TL) Implementation

Similar to TD, different python libraries are used to implement the tasks of the TL component. Besides using those libraries, an external knowledge base has also been integrated. The description of the implementation details of this component is presented as follows:

- **LG**: To implement Algorithm 5 which generates a candidate label set for a given topic, the pos_tag method from $nltk$ python library is used to get part of speeches (POS) of words in the topic.

[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html

[9] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html

- **LR**: As discussed in the previous chapter, the ranking of candidate labels involves four properties, namely, popularity, term specificity, topic specificity, and coherence properties. In order to implement the popularity property, it is required to use an external knowledge base which provides a linked data set containing as huge triples as possible so that information on any term/label can be found in the data set. As discussed in Section 2.3.1 , DBpedia is the most widely used domain-generic data set covering variety of topics. Therefore, to implement the popularity property, DBpedia[10]is used as a knowledge base and accessed at virtouso[11] endpoint using SPARQL rdf query language through SPARQLWrapper[12] python library.

6.3. User Interface

A website is provided to give access to the developed TDL system. Even if there are many web frameworks for python, flask belongs to the most popular ones. Flask provides different qualities such as simplicity, flexibility and fine-grained control [82]. Since the webpages required for the proposed system are not too complicated, it is efficient to use Flask. Therefore, the website is designed and implemented using Flask[13] framework for python.

There are three main services provided by the website:

- **Online TDL**: This service is to enable users to access the system online. First, a user adds a transcript directly on the page and then the system detects the topics available in the transcript and returns the topics along with their corresponding labels back to the user. In addition to the actual results, the user can also see the dendrogram generated during the topic detection process. The relevance score computed for each label is also presented with the labels. The webpage dedicated for this purpose is shown in Figure 6.1 with the example presented in the previous chapter, Chapter 5, to explain how the TD and TL components work.

[10] http://wiki.dbpedia.org/
[11] http://dbpedia.org/sparql
[12] https://rdflib.github.io/sparqlwrapper/
[13] http://flask.pocoo.org/

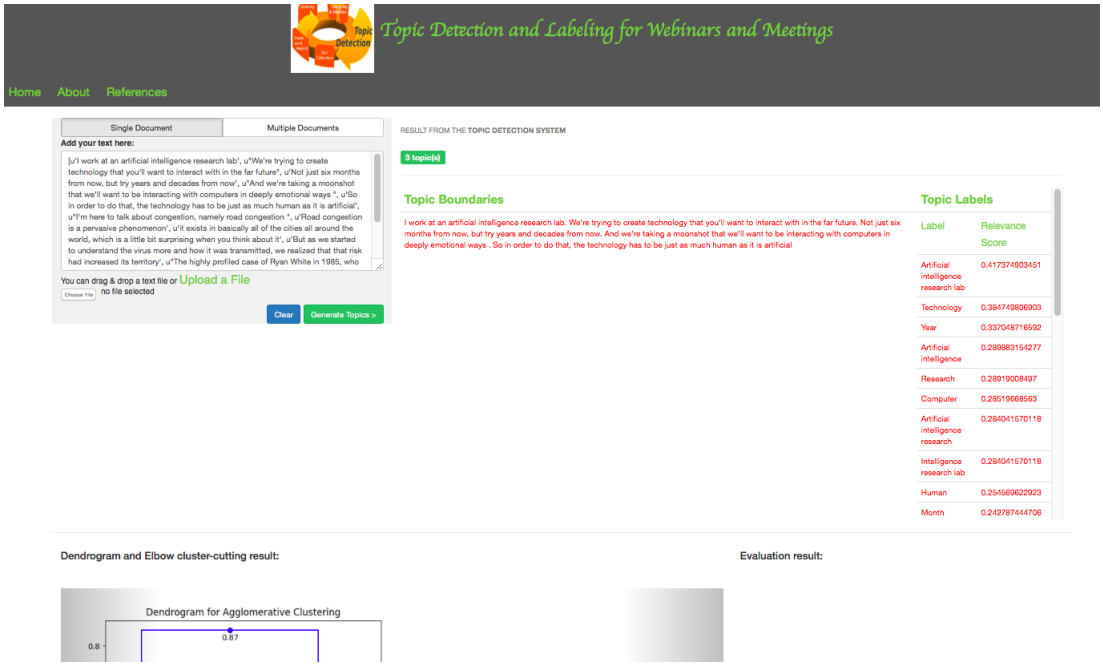


Figure 6.1.: The webpage to use the system online

- **Offline TDL**: The other service is to use the system offline using commands in a terminal and visualize the result on a website. This way a user can use the system to detect topics and generate labels for multiple transcripts at a time and once the process is completed the result can be seen on a website. An example of the webpage which is used to visualize the result offline is shown in Figure 6.2.

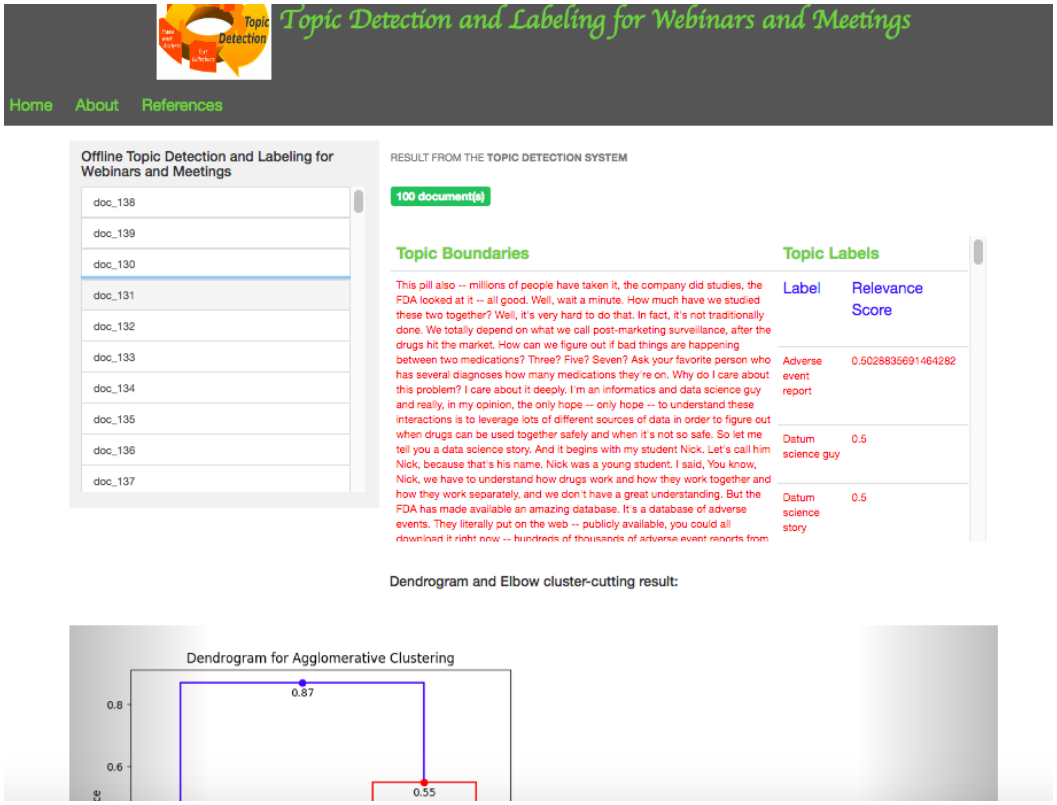


Figure 6.2.: The webpage to use the result offline

- **Evaluation result visualization**: The last main service of the website is to let the developer of the system visualize the result of the experiment conducted and evaluated. At this webage, it is possible to go through each transcript used during the experiment and see the boundaries of the topics in the ground truth and also in the result returned by the system. Moreover, it also provides the result of the evaluation of all of the transcripts involved in the experiment. A screen shot of this webpage is displayed in Figure 6.3.

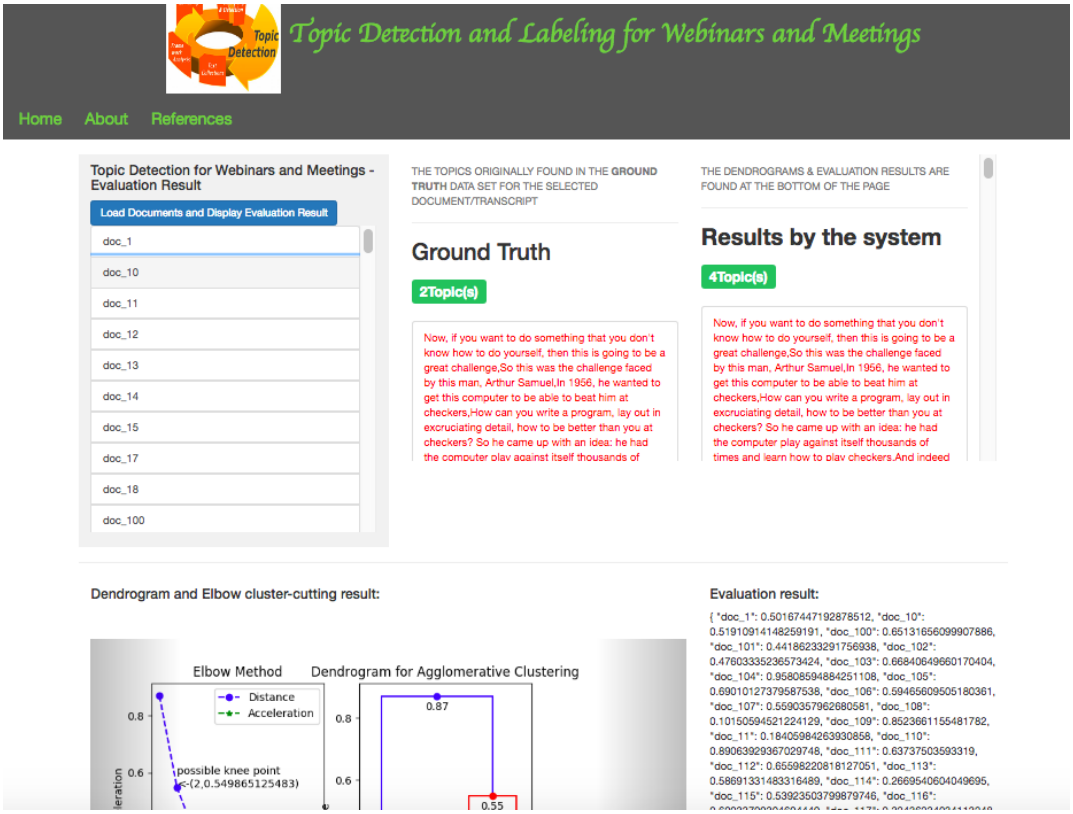


Figure 6.3.: The webpage to visualize the evaluation result

6.4. Summary

In this chapter, the implementation details of the proposed system has been discussed. The Python programming language is used to write the code for the system. Moreover, different technologies, tools, and libraries are used for implementing the various parts of the components of the system. As an optional task, different webpages have been designed and implemented using flask python framework so as to make use of the proposed TDL system.

The implementation has fully covered all designed components of the proposed system. For parts of some of the algorithms such as the CL function of the clustering algorithm, the 'linkage' method provided by the $scipy.cluster.hierarchy$ package from the Scipy python library has been reused. Apart from those already available methods in various python libraries that can be reused, the main part of the system are written from scratch. Besides the system being used as a whole, the two main components of the system, namely, TD and TL can be used by their own; each of them as a separate system. Therefore, any one of these components can be integrated with any other existing system that can make use of them.

7. Experiment and Evaluation

As mentioned in Chapter 1, it is beneficial to have a Topic Detection and Labeling (TDL) system for webinars which can be used in many fields. To this end, this master's thesis study has been undertaken with the main objective of designing and implementing a TDL system for webinars and meetings. In order to actually determine the relevance of the study and check whether it has satisfied its requirements, it is needed to do an experiment and evaluate the findings. Thus, in this chapter, the set of experiments conducted to validate the proposed TDL approach is presented.

First, the experimental procedure which discusses the data set collection, the ground truth, and the technical setup for the experiments is presented followed by the evaluation of the main two components, namely, Topic Detection (TD) and Topic Labeling (TL) of the system separately. The components are evaluated separately due to the fact that they can exist as an application individually by their own and besides different evaluation methods are applied for each of them. Finally, the entire system will be evaluated by combining the results of its components together.

7.1. Experimental Procedure

In order to do the experiment, collection of a test data, preparation of a ground truth, and selection of appropriate evaluation metrics have been required. Different strategies and metrics are used to evaluate the result of the experiment separately for the two main components of the system; TD and TL. In the following sub sections, we will present the source of the test data, the ground truth, and the technical setup separately.

7.1.1. Data Source Collection

Two sets of data sets are used to test and evaluate the system; a set of transcripts and a linked data set. The first is required to test both components of the system whereas

the later is used by the TL system only.

DBpedia data set

DBpedia[1] is a freely available general domain linked data set which contains a set of RDF triples. This data set is used as an external knowledge base to which the proposed system, specifically the algorithm designed for the TL component, can be applied so as to evaluate its performance. This dataset is used online through SPARQL queries during the development of the system.

Test Data

A set of webinar transcripts were required so as to undertake the experiment. The source for these transcripts was TED talks[2]. TED talks have a wide variety of webinars or videos in different domains such as technology, entertainment, politics, health, social life, design, and etc almost covering all topics. For this particular experiment, our data set contains 70 Ted talk transcripts in most of those domains.

7.1.2. Ground Truth

It is quite challenging to find ground truth to test the relevance of the proposed approach as preparing a ground truth involves identifying the actual topics that exist in each document along with their boundaries and also assigning possible labels for the topics. The most common ways to get ground truths are either to prepare them manually or to gather transcripts which are already labeled with ground truth. The former is not applicable because of various reasons. First and foremost, doing it manually for hundreds of transcripts requires high human labor to detect topics and to find appropriate labels for the identified topics for each of these transcripts. Besides, even if we let more than one person prepare the ground truth, it still may not be reliable enough as the sense of topics depend on the eye of the beholder. The later, it is quite difficult and mostly impossible to find webinar transcripts which are already properly tagged with topic and their labels. Therefore, for conducting the experiment in this thesis, the ground truth is prepared in a way which is semi-automatic.

In our data set each TED talk transcript is tagged with its respective classes or categories of the nature of the domain or field on which the talk has been prepared. In order to prepare the ground truth an algorithm has been designed and implemented which takes a set of transcripts as input and generates as an output a set of documents which are tagged with topic boundaries. Given a set of transcripts T and a set of $(\#topic : \#Document)$ pairs, for each $\#topic : \#Document$ pair the algorithms creates $\#Document$ number of documents each of which containing $\#topic$ number

[1] `http://dbpedia.org`
[2] `https://www.ted.com/talks`

of topics. A document d containing $\#topic$ number of topics is created from a randomly selected set of $\#topic$ number of transcripts $T' \subseteq T$. Each topic is generated by randomly choosing n number of consecutive sentences from $t \in T'$. For the evaluation purpose, using this algorithm, 150 documents each containing 2 topics, 75 documents each with 3 topics, 50 documents each containing 4 topics, and 25 documents each one made of 5 topics were created from 70 transcripts gathered from TedTalk. It is easier with documents created this way to see the actual boundaries of the topics in the documents.

7.1.3. Technical Setup

The experiments in this thesis will be conducted using MacBook Air (13-inch, Early 2015) OS X EI Captain. The processor of this machine is 2.2 GHz Intel Core i7 and its Memory is 8 GB. Eclipse framework is used to run the tests.

7.2. Evaluation

Since the two main components of the system can be stand alone systems by themselves, the evaluation process has been divided into 3 parts. First the result of the experiment for the TD component is evaluated with the chosen metrics. Then, the result of that of the TL component of the system has been evaluated. Finally, the system as a whole is evaluated based on the requirements and research questions specified in the beginning of the research.

7.2.1. Evaluation of the TD Component

In order to evaluate the performance of the TD component, a two step process is required. First, a set of experiments has to be undertaken to optimize parameters involved in the algorithms of the TD component. Then, the actual evaluation of the component will be conducted using the optimized parameter values. Moreover, before starting the whole evaluation process, it is required to choose appropriate evaluation measure(s). Therefore, in this section, first the chosen evaluation metrics are discussed followed by the experiments for parameter optimization. Then, a discussion on the result of the experiments will be presented with the purpose of choosing the best values. Finally, the evaluation will be done for a separate set of transcripts using the optimized parameter values.

Evaluation Metrics

Since the approach proposed to solve the TD problem is based on a sentence clustering technique, it is possible to choose appropriate metrics from the cluster evaluation metrics available. In this study, to evaluate the TD component we have used two evaluation metrics, namely purity [83] and v_measure_score[3] cluster evaluation metrics used by

[3] http://scikit-learn.org/stable/modules/clustering.html

clustering algorithms. Purity is an external cluster quality evaluation technique. It is designed in such a way that it focuses on the frequency of the most common category from the reference distribution (i.e. in our case it will be a topic from the ground truth) into each cluster (topic generated by the proposed system). The formula used to compute purity [84] is shown in Equation 7.1.

$$Purity(\Omega, C) = \frac{1}{N} \sum_{k} \underset{j}{max} |w_k \cup c_j| \tag{7.1}$$

where $\Omega = w_1, w_2, ..., w_K$ is the set of clusters to be evaluated, $C = c_1, c_2, ..., c_J$ is the set of categories and N is the total number of data points/sentences to be clustered.

Even if purity considers penalizing the noise found in a cluster, it does not reward putting together data points from the same category; if we simply make all clusters singltons, then we will have a maximum purity value [85]. Therefore, it is required to also use another clustering technique apart from purity which is able to reward grouping together data points from the same category. To this end, the metric called v_measure_-score is used which is designed mainly with the intention of measuring the closeness between the result generated by the system and the ground truth. It is defined as the harmonic mean between the homogeneity and completeness of the clustering result [86]. Homogeneity is satisfied if all of the clusters returned by the system contain only data points which are members of a single class. Specifically, in our case, homogeneity is achieved if all the topics contain sentences from the classes from which their transcripts are collected from. On the other hand, completeness is satisfied if all the data points that are members of a given class are elements of the same cluster. The values returned by all these metrics are in the range from 0.0 to 1.0 where 1.0 indicates a perfect result. The formula used to compute v_measure is defined in Equation 7.2 bellow.

$$V\_measure\_score = 2 * \frac{(homogeneity * completeness)}{(homogeneity + completeness)} \tag{7.2}$$

Experiments for parameter optimization

In order to conduct the actual experiment and evaluate the result for the TD component, the parameters required as an input for the algorithms involved in this component of the system need to be set first. In total there are six parameters that need proper value selection. The first is the word Embedding model used by the sentence embedding algorithm, Algorithm 1, to get word vectors as inputs. The second is the parameter α used by the same algorithm to determine if a sentence is properly vectorizable or not. Similarly, the third parameter β is required by this algorithm as an input to decide if a transcript is vectorizable or not. The forth and fifth ones are α and β in sentence similarity algorithm, Algorithm 2, to give weight to the euclidean distance measure and to the $in\_transcript$ distance functions respectively. The last one, the sixth parameter, is the cutting_option for the DONT algorithm as mentioned in Algorithm 4. In order to optimize the values of those parameters discussed above, it is required to conduct

experiments.

Thus, in this section we discuss the experimentation setup and the actual experimentation result obtained. Note that the the technical setup for the experiment is discussed in section 7.1.3.

- **Experimentation setup**: In order to make the number of experiments as small as possible due to time limitation, the values for the parameters α and β for Algorithm 1 are assigned the value 1. The value 1 for α is interpreted as a sentence is considered as properly vectorizable if only at least one of its words exist in the word embedding model. Similarly, the value 1 for β implies that a transcript is properly vectorizable if only at least one of its sentences is properly vectorizable. This value is chosen due to the fact that even if most of a sentence's words do not exist in the vocabulary of the word embedding model considered, the issue will be avoided by the In_transcript_distance function because it enables the system to assign the sentence to the cluster to which its neighboring sentences belong to.

  The word embedding parameter, as discussed in Chapter 5, has two options Word2Vec and Glove chosen for the experimentation. A summary of the specification of these two word embedding models is presented in Table 7.1. The two parameters α and $\beta = 1 - \alpha$ for the 2 algorithm has been given values in the range 0 to 1 as 0.1, 0.2, ..., 1. For instance, $\alpha = 0.6$ and $\beta = 0.4$ implies that the result of the eculidean function will constitute 60% of the main distance function Dist_sen where as the rest 40% will come from the In_transcript_distance function. As mentioned in Algorithm 4, "first" and "second" are the possible values for the cutting_option parameter and the experiments are conducted using both values.

	Word2Vec	**Glove**
File Size	1.5GB	822 MB
Vocabulary Size	3 million words and phrases	400 K
Training Data set	100 billion words from a Google News dataset	6B tokens from Wikipedia 2014 and Gigaword 5
Dimension	300	300
Loading Time	103.08 seconds	123.6 seconds

Table 7.1.: Information on the pre-trained Word2Vec[4] and Glove[5] word embedding models used for the experiment

  Therefore, the experiment has been conducted using 100 transcripts/documents with the combination of the values discussed above for the 6 parameters.

- **Experimentation result**: The result of the experiment using Word2Vec embed-

ding model is shown in Table 7.2 and also depicted in Figure 7.1. Similarly, Table 7.3 and Figure 7.2 shows the result with Glove word embedding model.

Table 7.2.: Experiment choices and results using word2vec Embedding model for 100 documents TD component

Parameters		**V-measure**			**Purity**		
α in SS (Euclidean)	β in SS ($In\_transcript\_distance$)	With optimal number of topics predefined	Using Elbow		With optimal number of topics predefined	Using Elbow	
			first cut option	second cut option		first cut option	second cut option
0	1	50.61	49.92	49.44	85.80	84.08	89.89
0.1	0.9	50.99	49.22	49.29	85.83	84.70	87.95
0.2	0.8	48.38	49.35	47.6	83.85	84.70	86.33
0.3	0.7	52.18	50.48	48.93	85.50	84.32	88.36
0.4	0.6	52.42	52.24	49.93	85.92	85.90	89.04
0.5	0.5	51.28	52.31	49.80	85.51	85.69	89.48
0.6	0.4	53.90	52.48	51.18	86.08	85.43	89.60
0.7	0.3	52.29	51.61	49.64	85.53	84.36	89.07
0.8	0.2	50.71	50.30	48.44	85.21	84.80	87.89
0.9	0.1	31.83	36.12	36.08	77.59	79.62	83.90
1	0	4.78	4.88	8.87	8.87	66	69.11

- **Discussion on the Result of the Experiments**: The experiments are conducted with the intention of optimizing the parameters against one chosen evaluation metric from the v_measure and purity metrics. Unlike purity, as discussed before, v_measure considers both homogeneity and completeness when computing qualities of clusters. Therefore, for such reason v_measure has been selected as a metric for optimization. Nevertheless, since purity plays a vital role in showing the performance of the boundary detection task of the system, it is also used to measure the quality of the system.

  Looking at Figure 7.1, the evaluation result for Word2Vec, v-measure with the *number of topics predefined option* has a maximum value of 53.90% which is obtained at $\alpha = 0.4$ (and or $\beta = 0.6$). In the same figure, 52.48% is the maximum v-measure value found at $\alpha = 0.6$ (and or $\beta = 0.4$) with the *elbow method using the first cutting option*. In the same figure, the maximum v_measure value 51.18% *for elbow with the second cutting option* is found at $\alpha = 0.6$ (and or $\beta = 0.4$). On the other hand, using Glove in Figure 7.1, the *number of topics predefined option* has the maximum value 52.57% for v_measure obtained at $\alpha = 0.1$ (and or $\beta = 0.9$). In the same figure, 51.30% is the maximum v-measure value with the *Elbow method using the first cutting option* found at $\alpha = 0.3$ (and or $\beta = 0.7$)

Table 7.3.: Experiment choices and results using Glove Embedding model for 100 documents TD component

Parameters		**V-measure**			**Purity**		
α in SS (Eucli dean)	β in SS ($In\_transcript\_distance$)	With optimal number of topics predefined	Using Elbow		With optimal number of topics predefined	Using Elbow	
			first cut option	second cut option		first cut option	second cut option
0	1	50.61	49.92	49.44	85.81	84.08	89.89
0.1	0.9	52.57	51.77	48.99	86.03	85.54	88.57
0.2	0.8	49.47	49.76	47.86	84.69	84.62	87.82
0.3	0.7	50.13	51.30	48.59	85.01	85.52	87.13
0.4	0.6	50.25	50.84	49.8	84.99	84.95	88.31
0.5	0.5	49.78	50.75	49.11	84.74	85.08	88.21
0.6	0.4	51.14	51.08	50.21	85.77	84.58	89.4
0.7	0.3	51.14	50.40	47.4	85.60	84.47	88.15
0.8	0.2	46.29	47.70	43.73	83.19	83.36	86.82
0.9	0.1	29.32	33.05	34.36	76.80	79.23	82.48
1	0	4.27	5.61	11.07	65.23	66.44	74.69

whereas 50.21% is the maximum v_measure value with *elbow algorithm using the second cutting option* at $\alpha = 0.6$ (and or $\beta = 0.21$) .

Since finding an optimal number of topics is one of the research questions discussed in Section 1.3 and in this research the Elbow method is the one proposed as a solution for this problem, more focus is given to the results obtained with the elbow algorithm. Thus, the parameters should be optimized based on the elbow algorithm. To that end, the v_measure values for the elbow algorithm are compared for both glove and word2vec.

In the case of both Word2Vec and Glove, looking at Figure 7.1 and 7.2, you can see that the difference among the v_measure values for α in the range 0 to 0.8 is not that significant for all the three types, namely, number of topic predefined option, elbow with first cutting option, and elbow with second cutting option. On the other hand, the v_measure values rapidly decrease as for $\alpha > 0.8$. Similarly, focusing on the elbow algorithm and comparing the first cutting option with the second cutting option, besides their graph having same shape as mentioned above, there is no significant difference between their values as well. In case of Word2Vec, the mean of the differences of the v_measure scores at each α value for the two cutting options is 1.62. This value is derived by first deducting the v_measure obtained with the second cutting option from that obtained with the first cutting option at each α value and taking their absolute values and computing their mean. Besides

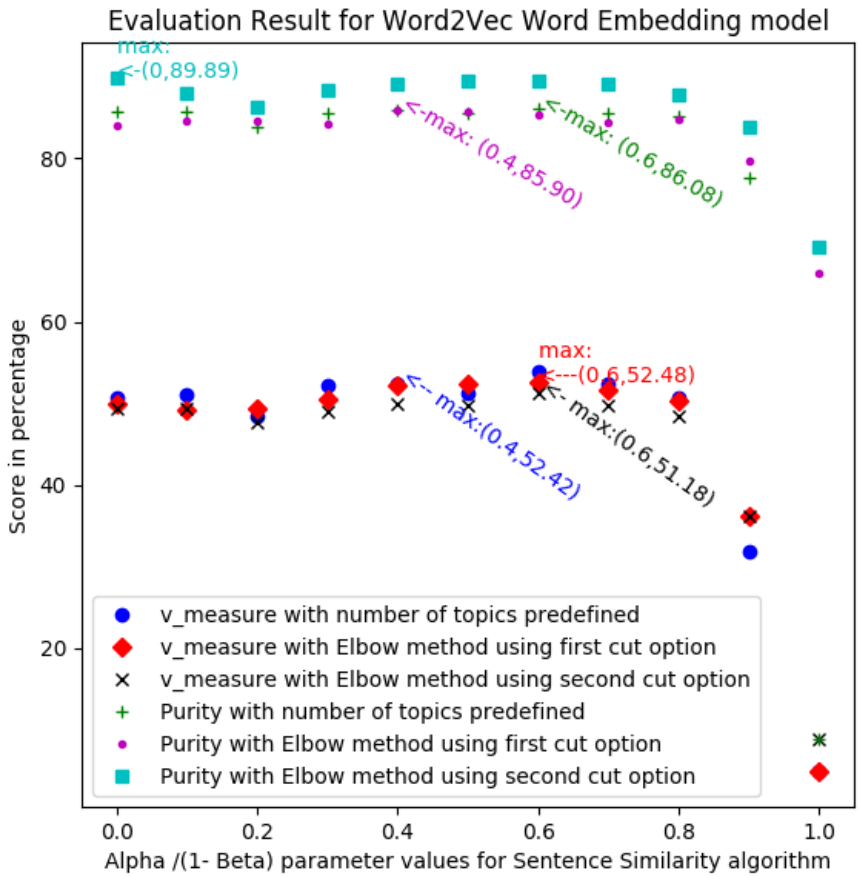


Figure 7.1.: V_measure and Purity evaluation result for TD component with Word2Vec word embedding model

the mean, the maximum difference in v_measure for the two cutting options is 3.99 found at $\alpha = 1.0$ obtained by deducting 4.88 (v_measure score with first cutting option) from 8.87 (v_measure score with first cutting option) whereas the minimum difference is 0.07 computed by deducting 49.22 (v_measure score with first cutting option) from 49.29 (v_measure score with second cutting option) at $\alpha = 0.1$.

Similarly for Glove, the mean of the differences of the v_measure scores at each α value for the two cutting options is 2.287. This value is derived by first deducting the v_measure obtained with the second cutting option from that obtained with the first cutting option at each α value and taking their absolute values and

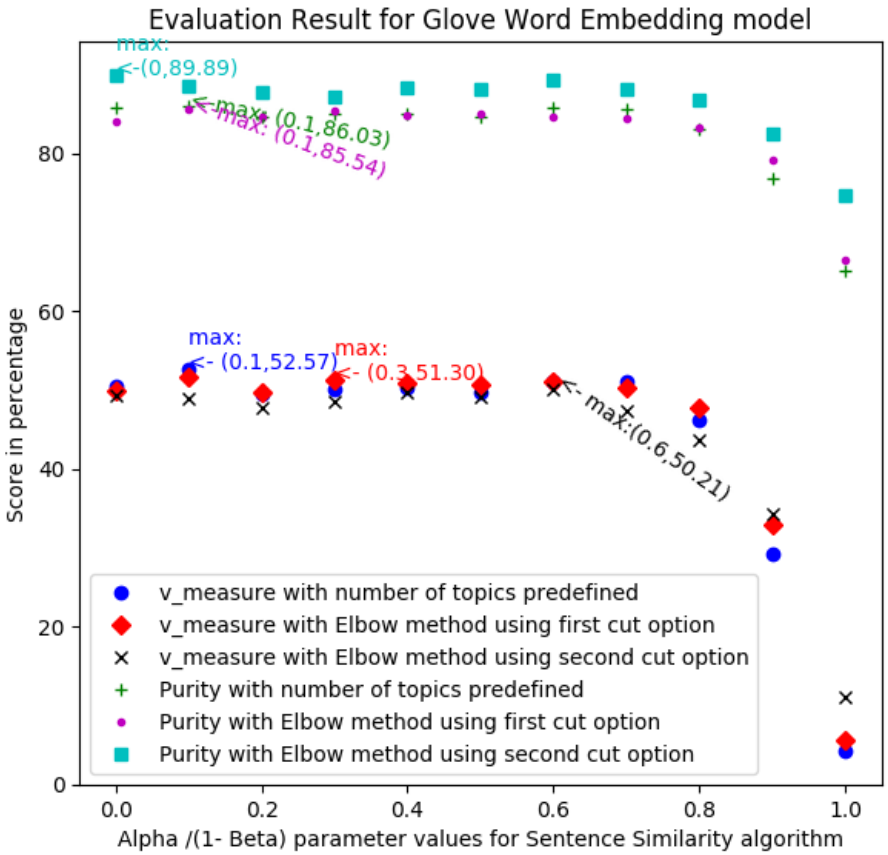


Figure 7.2.: V_measure and Purity evaluation result for TD component with glove word embedding model

computing their mean. Besides the mean, the maximum difference in v_measure for the two cutting options is 5.46 found at $\alpha = 1.0$ obtained by deducting 5.61 (v_measure score with second cutting option) from 11.07 (v_measure score with second cutting option) whereas the minimum difference is 0.48 by deducting 49.44 (v_measure score with second cutting option) from 49.92 (v_measure score with second cutting option) at $\alpha = 0$. This analysis shows that regardless of the word embedding model used, either glove or word2vec, there is no significant difference between the two cutting options as the mean values (1.62 for word2vec) and (2.287 for Glove) are very less.

Despite this analysis, since it is required to choose optimal values for the parameters, for word2vec the maximum v_score measure obtained is 52.48% (with

the first cutting option) which is better than that of glove's, which is 51.30%, by 0.2%. Therefore, the parameter values that are used to obtain the value 52.48% are considered as the best/optimized values. This indicates that, the system performs slightly better if the parameters α and β are assigned the values 0.6 and 0.4 respectively given Word2Vec as the word embedding model and the *first cutting option* is chosen for the elbow algorithm. Note that due to the fact that only one data set is used for the experimentation it may seem that the result is dependent on the data source used. However, we have tried to make the data set as diverse as possible by generating transcripts from different tedtalk categories. Besides this, we have generated transcripts of different sizes and number of topics. Thus, we argue that the parameter values are not dependent on the data set used.

Apart from using v_measure for optimization, purity has also been used to show the performance of the system regarding the purity of the clusters/topics (i.e. topic boundaries). When we look at these purity values for the word2vec word embedding model in Figure 7.1, the same case as in v_measure, purity shows no significant difference for alpha between 0 and 0.8 and it decreases as alpha gets bigger than 0.8. This holds true in all the three cases namely, number of topics predefined option, the elbow algorithm using the first cutting option, and the elbow algorithm using the second cutting option. However, in case of both word2vec and Glove in general all the three options together, the maximum purity value of 89.89% has been achieved at $\alpha = 0$ (and or $\beta = 1$).

Note that this value is interpreted as using in_transcript_distance function alone to determine the distance between sentences. This doesn't guarantee the best performance always as topics in transcripts can be mixed up with not only one single boundary i.e. a topic may come in between sentences of one topic. Therefore, it is not reliable to just use the in_transcript_distance function alone ($\beta = 1$). Rather it is required to also consider the actual similarity or distance between sentences. On the other hand, it is not also a good idea to totally ignore in_transcript_distance as it can be seen from the result of the experiments that the smallest values for both purity and v_measure in all cases are obtained with $\alpha = 1$ (using only euclidean as the distance function). Due to this fact, you can notice on the graph that, in case of v-measure, the better values are found in the middle where the values for the parameters α and β are nearly equal. This is the prove for the reason, which is v_measure considers both completeness and homogeneity while purity doesn't, to be taken in to consideration when choosing v_measure as the parameter optimization function instead of purity.

Besides evaluating the quality of the TD component, it is sensible to also consider the speed to discuss the performance of the system. When it comes to speed, it takes 113.3 seconds to run, including the loading time 103.08 seconds mentioned in Table 7.1 for word2vec training model, the TD component against a randomly

chosen input transcript of the average size 482 sentences calculated in Section 3.4. On the other hand, with glove, it takes 128.34 seconds including the loading time 123.6 seconds to run with the same transcript. For the chosen 100 transcripts/documents for the experiment, in total it took 169.08 seconds with Glove and 154.8 seconds with Word2Vec model. This indicates that TD takes slightly less time with word2Vec than with glove. Thus, according to the result of the evaluation process, Word2vec is preferable both in quality and speed.

Evaluation with the Selected Parameter Values

Once the parameters are optimized, the next step is to evaluate the performance of the TD component against a separate set of transcripts using the values chosen for the parameters. As discussed above, the values for those 6 parameters are determined by conducting a set of experiments. These parameters with their respective optimized values are presented in Table 7.4.

Table 7.4.: Parameter assignments for TD evaluation

Parameter	**Values**
Word Embedding Model	Word2Vec
α in Algorithm 1	1
β in Algorithm 1	1
α in Algorithm 2	0.6
β in Algorithm 2	0.4
Cutting option	first

- **Test Setup**: The evaluation has been conducted by running the TD component against a data set of 10,000 documents/transcripts which are separate from the transcripts used during parameter optimization. These transcripts are generated using the same algorithm designed to generate transcripts for parameter optimization. The transcripts are of random sizes covering variety of topics. The technical setup for this evaluation process is the same as what is mentioned in Section 7.1.3. The evaluation is measured by taking the v_measure score derived for each transcript and computing the average (summing up each score and dividing the result by number of transcripts in the data set) and multiplying the final result by 100.
- **Evaluation Result**: The result of the evaluation indicates that 76.5% and 69.56% v_measure scores are obtained using the *number of topics predefined* option and with the *elbow algorithm with the first cutting option* respectively. On the other hand, 87.56% purity is achieved with *number of topics predefined* option whereas 77.92% is obtained using the *elbow algorithm with the first cutting option*.

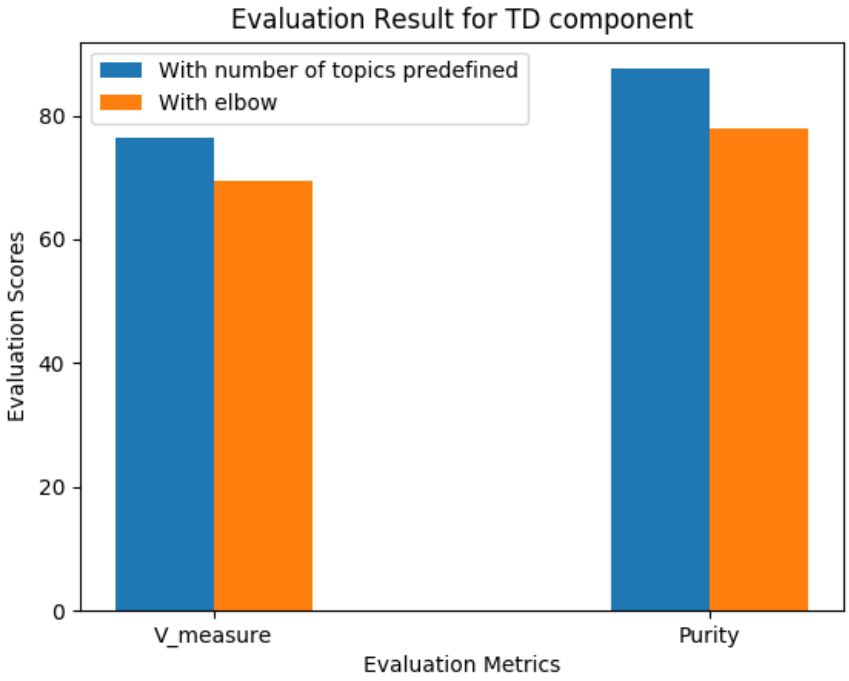


Figure 7.3.: The TD component evaluation result using v_measure and purity

- **Discussion on the result obtained**: As depicted in Figure 7.3, both v_measure and purity evaluation results indicate that the result obtained using the elbow algorithm (to find the optimal number of topics) is close to the result achieved with number of topics predefined option. This proves that the elbow variant algorithm used in this thesis performs well in determining optimal dumber of topics. Besides, as discussed before, the data sets used in these tests contain different transcripts which are of random sizes from small to large and most importantly discussing variety of topics. Given this fact (i.e., the transcripts being diverse) and also both v_measure and purity showing the same behavior, it can be concluded that the performance of the TD component is stable regardless of the data set used.

  The run time of this component is mentioned above while discussing the result of the parameter optimization experiment and it is proven that the component takes considerable and appropriate time to run given the technical set up used for evaluation. To sum up, the fact that the proposed TD system runs in considerable time, possess stability in performance, satisfies the derived requirements, and shows more than average performance as measured using v_measure and purity (i.e., v_measure = 69.56% and purity = 77.92% with elbow algorithm), proves that the component is capable enough to be used for the purposes that it is intended for which are discussed in detail in Section 1.7.

7.2.2. Evaluation of the TL Component

Similar to the the evaluation process for the TD component, a two step process is also required here. First, a set of experiments has to be undertaken to optimize parameters involved in the algorithms of the TL component. Then, the actual evaluation of the component will be conducted using the optimized parameter values. As it has been done for the TD evaluation, before starting the whole evaluation process, it is required to choose appropriate evaluation measure(s). Therefore, in this section, first the chosen evaluation metrics are discussed followed by the experiments for parameter optimization. Then, a discussion on the result of the experiments will be presented with the purpose of choosing the best values. Finally, the evaluation will be done for a separate set of topics using the optimized parameter values.

Evaluation Metrics

For testing the relevance of the TL component of the system, it is required to have a ground truth which contains a set of topics and their respective labels. Since it is very difficult to find such ground truth, it was necessary to find a way to prepare it. One of the common and more efficient way to do so is to generate topics and to let humans label it manually. However, due to the fact that it is time consuming for users to label topics by themselves, we find the idea of presenting users with candidate labels much more convincing. To that end, a cross-sectional survey using a questionnaire with closed-ended questions is prepared. Each question has a topic text and a set of candidate labels as options to let users choose the ones that they believe better interpret the given topic.

The questionnaire is made using Google forms[6] which is an internet-based program to gather data. In the questionnaire, a topic is presented as a question and its candidate labels are listed as check-boxes. Part of the first page of the questionnaire is shown in Annex A.1. We have tried to include as much variety of users as possible depending on their area of expertise/profession. Having users from different profession to involve in the survey helps to get human judgment from different classes of potential users of the system. For instance, given a topic (i.e. a set of sentences) and its respective shuffled list of candidate labels, the human judgment is to choose those labels which are capable enough to annotate the topic from the list.

Once the results of the survey are collected, the evaluation of the component has to be computed using an appropriate evaluation metric. F-measure is the standard measure for evaluating IR and also clustering methods by combining recall and precision techniques. Moreover, for those systems which return top K values like search engine and recommendation systems, it makes more sense to compute precision and recall metrics for the first K items instead of all the items. Since a topic labeling system is one of those applications which return top-K items to the user, it is sensible to compute precision and

[6] `https://www.google.com/forms/about/`

recall for the first K elements. Recall is the ratio of the number of labels assigned to the topic correctly to the total number of labels that the topic can actually have whereas precision is the ratio of the number of labels assigned to the topic correctly to the total number of labels returned by the system. However, recall is not considered for evaluating the topic labeling system in this research due to the fact that it is required to have all the topics manually labeled by human and doing so is time consuming. Besides, since both recall and precision are necessary to compute f-measure, it won't also be used. This leads to the conclusion that Precision at K is the only feasible evaluation metric that can be used for measuring the quality of the topic labeling system as it doesn't require manual labeling. Thus, after gathering the human judgment, we have used precision at k to evaluate the performance of this particular component of the system. Precision is calculated using the formulas shown in Equations 7.3.

$$Precison = \frac{\#CorrectLabels}{\#SystemLabels} \tag{7.3}$$

where *#CorrectLabels* is the total number of labels which are assigned to the topic correctly and *#SystemLabels* is the total number of labels returned by the system.

Experiments for parameter optimization

As in the case of TD, in order to evaluate the TL component, the parameters required as an input for the algorithms involved in the component are need to be set first. One of those parameters is the word embedding model parameter required by the coherence algorithm. The rest of the parameters namely, α, β, δ, and σ are used by the LR algorithm to assign values for *Popularity*, *Term Specificity*, *Topic Specificity*, and *Coherence* properties respectively. In order to assign/choose optimal values for these parameters, experiment has been conducted. Thus, in this section we discuss the experimentation setup and the actual result obtained in the experiment. Note that the the technical setup for the experiment is discussed in section 7.1.3.

- **Experimentation setup**: The word embedding model parameter is set as Word2Vec due to the result of the experiment in TD evaluation which shows that Word2Vec performs better than Glove. The parameters α, β, δ, and σ are configured to take values in the range 0 to 1 as 0.1, 0.2, ..., 1 where the sum of their values must sum up to 1 (i.e. $\alpha + \beta + \delta + \sigma = 1$). For instance, an assignment of $\alpha = 0.2$, $\beta = 0.3$, $\delta = 0.2$, and $\sigma = 0.3$ indicates that Popularity is given 20% weight in the ranking formula , Term Specificity takes 30%, Topic Specificity is given 20%, and the rest 30% of weight to Coherence. Given this set of values, 256 combinations are created which sum up to 1 (100%). Therefore, the experiment is set up to find the best combination from those 256 combinations which optimizes the precision values at each $k \in [1, 5]$ (i.e. k is from precision at k).

  To this end, the survey which is described in detail in the evaluation metrics section above has been used to gather human judgment for the purpose of evaluating

the topic labeling result. Even though having a lot of people to participate in the survey may seem beneficial, due to the average precision used for measuring the evaluation, the result doesn't change much as the number of participants grow. Thus, we have had twelve people from different area of expertise to participate in the survey. Given the fact that their profession is diverse, we considered the number to be good enough.

The questionnaire contains eighteen questions each with a topic and a set of labels to choose from. Clearly it is an advantage to have more set of questions but eighteen is already a lot for the users as it is observed during the survey collection. These eighteen questions are divided into two groups; one with ten questions for parameter optimization and the rest eight are for evaluation. More number of questions (ten as compared to eight) is allocated for optimization because it is better to have as much ground truth as possible to have a stable result for optimization. The eighteen topics included in the survey are generated from the tedtalk transcripts using the ground truth preparation algorithm used in the TD component. Their respective candidate labels are generated using the LG algorithm, Algorithm 5. Since it is not possible to present all candidate labels, we had to run the LR algorithm against all list of candidates giving equal weights to all the parameters α, β, δ, and σ. Finally, the top fifteen labels were taken as candidate labels for each topic added in the questionnaire.

In parameter optimization, the idea is to find the assignment/s that optimizes the values of the parameters such that the possible maximum precision at k is obtained for most k values in K where $K = \{1, 2, 3, 4, 5\}$. Given n number of topics, a set of 256 combinations/assignments for the 4 input parameters C, and k, total number of users m, the first step is to compute precision at k for each topic t, for each $c \in C$ by using the formula in 7.4. This formula works by first computing precision at k per user response and then computes the average by summing up the values computed and diving the result by m. Then, the next step is to find those parameter assignments that give maximum precision value by using the formula given in Equation 7.6. It starts by computing the average precision for each assignment c, by adding up the *precision at k* values for each topic t and dividing the sum by n by using the formula in Equation 7.5. At the end, it returns the maximum averaged precision value for k and all the assignments that gives that values.

Note that the experiment is conducted only once because it is difficult to find more people to have multiple surveys and it is time consuming.

$$AvgP_t(t, c, k) = \frac{\sum_{j=1}^{m} P(t, c, k, j)}{m} \tag{7.4}$$

where $P(t, c, k, j)$ is precision at k computed using the formula in 7.3 with #correctlabels being labels chosen by user j for the topic t and #systemlabels being the labels generated by the system with the parameter values c for the topic t.

$$AvP_c(k,c) = \frac{\sum_{j=1}^{n} AvgP_t(t_j,c,k)}{n} \tag{7.5}$$

$$Assignments_{P_k}(k) = \{AvP_{kc}, c \mid \max_{c,AvP_{kc}} \bigcup_{c=1}^{C} AvP_c\} \tag{7.6}$$

where k is the level at which the precision is computed, C is a collection of parameter assignments, c is a single assignment from C, and n is the total number of topics labeled.

- **Experimentation result**: Given 256 assignments ($|C| = 256$) and 10 topics ($n = 10$), applying the formula in Equation 7.6, we get 36 combinations/assignments which give the maximum value 52.5% precision at $k = 1$, 2 assignments with precision 42.5% for $k = 2$, 2 assignments with 36.94% for $k = 3$, 1 assignment with 34.17% for $k = 4$, and 1 assignment with 32.67% precision for $k = 5$ has been obtained. In total, 42 (36 + 2 + 2 + 1 + 1) assignments are found which maximizes precision at least at one k value.

- **Discussion on the Result of the Experiments**: Having 4 number of parameters with 11 possible values (0 to 1 as 0.1, 0.2, ..., 1) creating 256 assignments/combination, it is difficult to plot their corresponding precision values at each k in a graph to compare them and interpret the graph. Thus, it is found to be efficient to just take out those assignments that maximize precision at least at 1 k-values and choose one from those that maximize precision at more number of k values. To this end, we ranked the assignments using a two step procedure. The first step is to give a score to each assignment by counting the number of k values at which they maximize precision where high score indicates high relevance. The second step is it to take all those assignments which scored the highest value in the first step (i.e. those which maximize as much number of *k*s as possible) and rank them by looking at the values of k that they maximize precision at, where an assignment which maximizes precision at k=i is more valuable than an assignment which maximizes precision at k=i+1. Finally, we randomly take one among those which have the highest rank.

  In this experiment, given all the 42 assignments that give the maximum precision values, only one assignment has the highest rank using the procedure explained above. The assignment maximizes precision at k = 1 and k=3. This assignment gives the parameters α, β, δ, and σ the values 0.1, 0.5, 0.2, and 0.2 respectively. Thus, these numbers will be considered as the optimized values for the parameters used in the LR algorithm. As mentioned above, due to the difficulty in plotting the results achieved using the different parameter assignments, the procedure followed in this step doesn't allow to discuss the actual difference that the assignments bring in the result of the precision values.

Evaluation with the Selected Parameter Values

The technical setup for the evaluation is the same as that of the parameter optimization experiment. The actual evaluation of the TL component has been done using the optimized values of the four parameters of the LR algorithm against the second group of 8 questions from the survey. The result of the evaluation using precision at k=1, upto k=5 is given in Table 7.5 and depicted in Figure 7.4. As you can see in the figure, the precision has highest value possible at k=1 and it continuously decreases as k goes up indicating that precision and the value of k are indirectly proportional. This is due to the way the LR algorithm is designed i.e. when it ranks topics it may give high rank to those labels which are not of importance to the topic but are either very popular in DBpedia, gaining high TF-IDF value, very specific obtaining high term specificity value, or highly topic specific.

You may notice that the maximum precision value (precision at 1) is just a bit more than average for various reasons. One reason is some of the users participated in the survey have selected irrelevant terms as labels to some of the topics. It is observed that some users do not read the topics text entirely rather they just scam it and choose what ever word they see in the list that they can find in the corresponding topic text. Some gets tired after answering some questions and choose "None of the above" as answers to the questions that come at the end. The other problem is users do not read the instructions that says "choose all that apply" and choose only one label for all the topics. For instance, the responding rate for one of the topics is depicted in Appendix A.2. If those situations has not happened, the result would have been better than it is now.

Similar to the process of TD evaluation, besides evaluating the quality of the TL component, it is required to also consider the run time to discuss the performance of the system. Thus, when it comes to speed, it takes 108.93 seconds, including the Word2Vec loading time 103.08 seconds mentioned in Table 7.1, to run the TL component against a randomly chosen input transcript of the average size 482 sentences (as calculated in Section 3.4) which contains 5 topics to be labeled. Given the technical setup used for evaluation, as shown in Section 7.1.3, the run time is promising to use the component separately.

To sum up, despite the evaluation result of the component being little more than average, which happened due to the reasons that are discussed above, the proposed TL component still is usable for the purposes it is designed for which are discussed in the Problem Analysis part in Section 1.2. Thus, it can either be used as a stand alone system or integrated with other systems such as TD.

Table 7.5.: Precision at k values for $k \in 1, 2, 3, 4, 5$ for the TL component evaluation

k	Precision at k
1	54.17
2	41.67
3	37.15
4	33.07
5	30.45

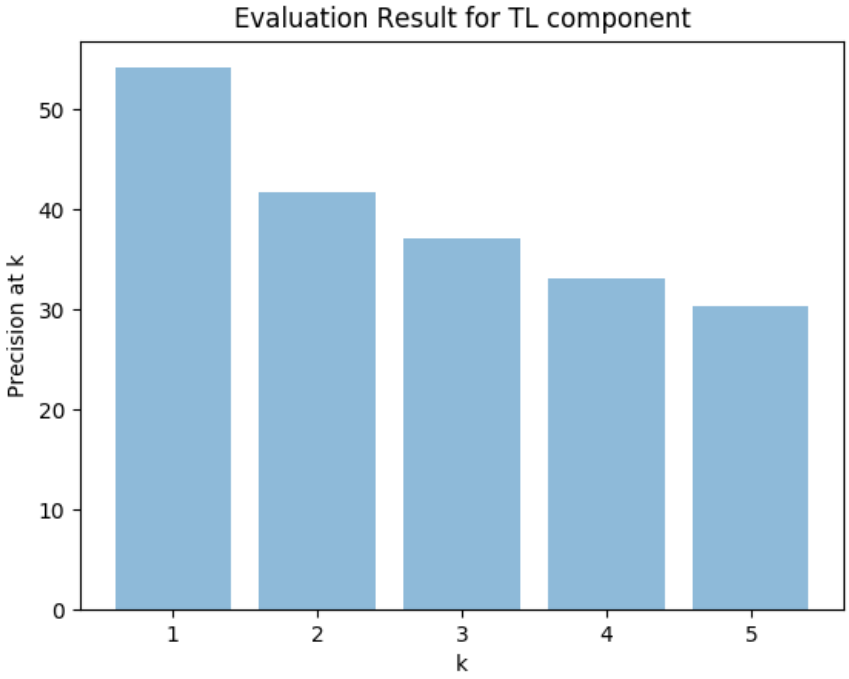


Figure 7.4.: Evaluation of the TL component using precision

7.2.3. Evaluation of the Whole TDL System

Once the main components of the proposed system are evaluated, it is required to check if the TDL system as a whole satisfies the research questions devised in Section 1.3 and the requirements specified in Chapter 3. To that end, in this section, the answers provided for each of the research questions and the description on how the proposed system satisfies the requirements are presented.

The following research questions are answered during this thesis work:

1. How to use sentence clustering algorithm for single-document topic detection?

- How to adapt an existing clustering algorithm in order to divide a document into segments or topics ?

  A review of the available clustering algorithms have been conducted as shown in Section 2.2.1 and the agglomerative hierarchical clustering algorithm is found to be fitting to achieve the purpose of the topic detection process for the reasons discussed in Section 5.2.2. Therefore, this algorithm is adapted by customizing the distance function used to generate the distance matrix used in the clustering process. The distance function is developed by combining an euclidean distance function shown in Equation 5.2 metric with a newly defined distance function shown in Equation 5.3, which determines the distance between data points or sentences based on their appearance in the transcript, together as in Equation 5.1.

- How to determine the optimal number of topics?

  As defined in the functional requirements section, Section 3.3, optimal number of topics in a transcript means the desirable number of topics that should be extracted from the transcript. That said, the possible ways available to determine the optimal number of topics are investigated in Section 2.2.2 and the elbow method is found to be sensible to use for the purpose due to the reason stated in Section sub:dont. Therefore, a variant of the elbow method shown in algorithm 4 is adopted and used for the DONT task. As discussed in the TD evaluation section, Section 7.2.1, it has been proved that in general the elbow method performs well with no significant difference with the topics predefined option.

2. How to use a Knowledge-base to give semantic labels to identified topics?

   - How to generate candidate terms/phrases to represent an identified topic?

     It showed that taking only nouns as labels is not effective as it leads to missing important labels. Besides, creating n-grams of all words is not an option either due to the issue that it will lead to having irrelevant n-grams. Thus, in this research, candidate labels for topics are generated by employing an algorithm which makes use of nouns and adjectives that exist in the topic sentences. First, it takes all the nouns that occur in the topic sentences as initial candidate labels and then concatenates those neighboring nouns and adjectives together to create multi-word labels. It makes sure that meaning less labels are not created by checking the order in which nouns and adjectives are created.
   - How to assign weight to each candidate term used as a label to a topic in order to rank them based on importance?

> In this research, we have shown that it is required to check the availability and popularity of a term in an external knowledge base for the purpose of labeling a topic with terms which are capable enough to represent the semantics of the topic. It is experimentally demonstrated that one way to remove senseless labels from the candidate set (assigning less relevance score) is by checking if it exists in DBpeda. As discussed in Section 7.2.1, the result of the evaluation of the TL component proved that checking the popularity of candidate labels in external knowledge bases along with the rest 3 ranking properties, namely, term specificity, topic specificity, and coherence has significant advantage for ranking candidate labels. This is indicated by the optimized parameters values i.e. popularity is given 0.1 weight, term specificity 0.5, topic specificity 0.2, and coherence 0.2. This entails that the LR algorithm performs well when all of these properties are considered during label ranking with their respective weight values.

Besides analyzing whether the proposed system has answered the research questions, it is important to investigate to what extent that the system has fulfilled the functional and non-functional requirements devised in chapter 3 as well. Therefore, the degree to which these requirements are fulfilled is described as follows with a summary of the evaluation status of the requirements in Table.

- FR-1 **Topic Detection**: The TD component of the system is designed and implemented by customizing a clustering technique which enables to create topics by grouping together sentences which have similar semantics. Therefore, the design of the system makes sure that a topic is made of sentences and has meaningful interpretation.
- FR-2 **Optimal Number of Topics**: As it was discussed in detail in Section 5.2.3, the proposed system employed a variant of the elbow algorithm, Algorithm 4, to automatically determine the number of optimal topics. The algorithm returns a couple of options to choose from where to cut a dendrogram to get an optimal number. Experiments have been conducted to choose the best one from these two options. As a result of the experiments, the first cutting option is selected and used as an optimized value for where to cut a dendrogram. Therefore, this requirement is fully satisfied with design, implementation and evaluation.
- FR-3 **Boundary Detection**: The TD component is designed in such a way that the boundary of every topic where it starts and ends is clearly identified from the transcript. This is achieved by involving a sentence clustering algorithm for topic detection which uses sentences as data points or observations such that the boundaries can be detected simultaneously while detecting topics. The quality of the boundary detection task of the system is evaluated along with topic detection evaluation using purity and v_measure.

- FR-4 **Topic Labeling**: A TL component has been designed and implemented to generate maximum of 5 labels for topics detected by the TL component. Four different properties, namely, term specificity, coherence, topic specificity, and popularity are analyzed to make sure the labels are important to the topic being labeled regrading to their semantics. Thus, this requirement has been addressed fully with experiments and evaluation which uses human judgment and precision evaluation metric.

- FR-5 **Utilizing External Knowledge-base**: During TL, the system uses an external knowledge base, specifically DBpedia, to check if candidate labels extracted from a topic text are meaningful or not. The popularity of the labels in DBpedia is used as one parameter to rank the labels according to relevance to the given topic. This requirement is evaluated as part of the parameter optimization of the TL component.

- NFR-1 **Response Time**: The system as a whole, including both topic detection and topic labeling, takes 121.47 seconds to return topics and their labels for a randomly chosen input transcript of the average size 482 sentences, as calculated in Section 3.4, using an operating system with 8 GB 2.2 GHz Intel Core i7 specification. It is more than appropriate to say the requirement has been satisfied given the specification detail.

- NFR-2 **Scalability**: The system should be capable enough to work regardless of the size of an input transcript. Based on the algorithms designed for both TD and TL components, it is theoretically guaranteed that the quality of the detection and labeling process affected by the increase in size of transcript but it may get slower as the method employed for TD is a hierarchical clustering technique. Even if it is theoretically known that size doesn't affect quality in hierarchical clustering, the data sets used for the experiments and evaluation are made sure to have different size of transcripts starting from small to large. This makes the data sets diverse in size of transcripts.

  In the current evaluation, the data set used for TD evaluation contain all type of transcripts i.e. small-sized, mid-sized, and large-sized transcripts. Thus, the evaluation is done by taking the evaluation score of each transcript together and computing their average. However, it is not clear to analyze the result based on size due to the reason the data set contains all size of transcripts. Despite this, the performance of the system given the type of transcripts used indicates that the component performs well regardless of the size of the transcripts.

  Nevertheless, the better way to clearly assess the evaluation result based on size could be to split the data set which contains 10000 transcripts into groups based on their size range i.e. to generate one data set with only small-size transcripts, another data set with mid-size transcripts, and one more data set with large-size

transcripts. This way it would have been either to show the stability of the system regardless of transcript size.

- NFR-3 **The system should always return a result**: As it can be understood from the design chapter, Chapter 5, the algorithms are designed to return at least a single topic. For instance, in a case where none of the words in the transcript exist in the word embedding model applied, a topic containing all the sentences in the transcript is returned.

Table 7.6.: TDL system requirements evaluation status

	Requirements	**Evaluation Status**
Functional	FR-1: Topic Detection	✓
	FR-2: Optimal Number of Topics	✓
	FR-3: Boundary Detection	✓
	FR-4: Topic Labeling	✓
	FR-5: Utilizing External Knowledge base	✓
Non-Functional	NFR-1: Response Time	✓
	NFR-2: Scalability	Not fully evaluated
	NFR-3: The system should always return a result	✓

7.3. Summary

In this chapter, the evaluation of the TDL system which is designed and implemented in this thesis has been presented with the purpose of showing the findings of the research. In order to do the evaluation, first it was required to undertake experiments to optimize parameter values. The experiments and evaluations are done separately for each component of the system. Then, the system as a whole is analyzed based on the research questions defined in Section 1.3 and the requirements specified in Section 3.

In the case of the TD component, the result of these experiments demonstrated that the Word2Vec word embedding model performs slightly better than Glove given their details in Table 7.1. In addition to quality, when they are compared based on speed, Word2Vec is again slightly better than Glove as it runs faster. Apart from the word embedding model parameter, the other parameters α (weight for the euclidean distance) and β (weight for the in_transcript distance) were optimized as well. The experiment indicated that using either euclidean distance or in_transcript distance alone does not give a better performance. However, combining them together to determine the distance between sentences has been proved to be better. Thus, according to the experiment result α is assigned weight 0.6 and β is given weight 0.4. Once the parameters are optimized,

the actual evaluation of the system has been undertaken. This evaluation has proved that this component works properly regardless of the size of transcripts. Besides this, it has been proved that the component can be used for the purposes that it is proposed for due to the evaluations results it showed with the elbow algorithm i.e., v_measure = 69.56% and purity = 77.92%.

Similarly for TL, experiments are conducted to optimize the parameters α (popularity), β (term specificity), σ (coherence), and δ (topic specificity). A survey was prepared to collect users judgment on the topic labeling component. 18 questions in the survey were prepared by pairing topics with their respective list of candidate labels so that users can choose those labels that can well interpret the meaning of the corresponding topic. Then the responses collected for the 10 questions were used as a ground truth to do the experiment and the rest 8 for the the evaluation purpose. The result of the experiment has shown that all these properties are required in order to achieve better performance. The value 0.1 is assigned for α, 0.5 for β, 0.2 for δ and 0.2 for σ. These optimized parameter values are used to evaluate the component with precision as an evaluation metrics. Precision at k for $k \in [1, 5]$ was computed for each question and the average precision at k showed that the system has better performance as k goes down.

Finally, the TDL system is assessed as a whole against the research questions and requirements defined for the system. This assessment proved that the system answered all the research questions as discussed in detail in 7.2.3. In addition, all the requirements which are specified in the early stage of the system are proved to be satisfied.

8. Conclusion and Future Work

In this chapter, we provide the summary of the whole thesis work showing its limitations, findings, and further recommendations that can be pursued as future works. Thus, in the following sections the conclusion, limitations, contribution, and future works will be presented.

8.1. Conclusion

It has become a common practice to use webinars for various purposes in different sectors due to their invaluable benefits. Webinars enable companies to have direct contact with their customers, to reach their target group live and afterwards, to have a more interactive sessions with audiences, to sell their product without investing a lot, and most importantly to save time and money. Given those benefits, webinars are now widely used by companies to reach their customers because of those qualities it provides. Besides webinars, online meetings are also a very important, almost a day to day, activity in various companies. However, once the webinars are provided it is required to make sure that those webinars are actually reaching the intended users. Thus, for users, accessing vital information out of these huge amount of webinars needs to be as simple and as effective as possible. In order achieve this, it is desirable to have automated systems which enable users to search using text. Therefore, having a TDL system which detects topics discussed in webinars/ meetings with a method to label the topics with relevant terms/phrases would bring enormous advantages in making information retrieval easy and effective. Having this as a purpose, in this master's thesis, a Topic Detection and Labeling (TDL) system for online webinars and meetings has been proposed.

The main goal of this study is to explore the role of clustering algorithms and external knowledge bases in TDL. The proposed system is composed of two main components, namely, TD and TL. The TD component was designed and implemented to detect topics from transcripts which are related in a hierarchical structure by using an agglomerative

clustering technique. This component of the system has SE, CAC, and DONT modules. The SE module is intended to compute embeddings (vector representations) for sentences of a transcript whereas the CAC and DONT are applied to create a dendrogram of topics and to find an optimal number of topics respectively. In order to implement the SE algorithm, Word2Vec and Glove pretrained models are compared and Word2Vec is found to be better in providing quality embeddings. The CAC algorithm customizes the agglomerative clustering method used in Scipy by modifying the distance function used. A variant of the elbow algorithm, which is found to be effective for the DONT purpose, is implemented and evaluated.

The TL component was built to generate labels and to rank them as per their relevance to the given topic using LG and LR modules respectively. Meaningful nouns and adjective-noun combinations are considered as candidate labels by the LG algorithm. Taking into considerations how semantically close a label is to all other candidate labels (*coherence*), the number of occurrences of the label in an external knowledge base (*popularity*), the number of tokens/words of a label (term specificity), and the TF-IDF value of a label in the given topic (topic specificity) during ranking candidate labels is experimentally proven to be better than just ranking with only one of these parameters. For instance, ranking with just the popularity value gives very less precision as compared to combining all parameters together.

Two set of experiments are conducted for each of the two components of the system. The first one is to train the system so as to optimize parameters required as an input for the algorithms used in the system. The other is to evaluate the system using evaluation metrics. For the TD component, 100 transcripts with a ground truth were prepared to optimize the parameters of the algorithms used in this component. In addition, the same component was evaluated using the optimized parameter values against a separate set of 10,000 transcripts and ground truth. V_measure and purity are used as evaluation metrics for the TD component whereas Precision is used to evaluate the TL component. The result of the evaluation process shows that the proposed system answers the research question and satisfies the requirement of the system better than the existing systems.

8.2. Limitation of the Study

- **TL evaluation**: During the evaluation of the TL component, the *topic specificity* property is not properly evaluated. This is due to the way the questions in the survey are prepared. In order to properly examine the role of *topic specificity*, it would be more efficient if the topics that belong to the same transcript were grouped together so that users may take that into consideration when they choose labels for topics. However, the topics are presented in rather flat structure showing no relationship among each other. It has been done this way to avoid making the questions too complicated for the users who participate in the survey.

- **User interface**: Even though user interface is not part of the requirements of the system, a website has been developed in order to use the TDL system. As discussed in Section 6.3, this website provides three main services, namely, online TDL, offline TDL, and evaluation result visualization services. As you may notice on the webpage in Figure 6.1, which gives the online TDL service, there is a button to let users upload a transcript. However, this button is not active currently. Thus, in order to use the online TDL service users have to copy and paste a transcript text into the text area. The website is not given much focus because it is optional and not part of the requirement of the system.

8.3. Contribution

The main contributions of this master's thesis are summarized as follows:

- Applying hierarchical clustering for topic detection by customizing its distance function. The study experimentally proved that taking the location of sentences in a transcript (i..e the $in\_transcript\_distance$ function) as a parameter for computing their similarity gives better result than just using euclidean distance alone.
- Identifying an algorithm which efficiently determines the number of optimal topics so as to stop requiring users to provide the number of topics in advance. A variant of the elbow algorithm has been implemented and evaluated showing a promising result.
- Exploring the advantages of using knowledge bases in developing a topic labeling system. It is demonstrated that checking the popularity of candidate labels in DBpedia helps to determine if the label is not nonsense and have semantics.
- A topic labeling system which considers different characteristics to rank candidate labels for a topic. The study proved that taking in to considerations TF-IDF values, coherence computed using cosine distance, term specificity, and popularity has positive impact on the quality of a topic labeling system.

8.4. Future work

Even though the proposed TDL system meets all requirements, it can be further improved in order to achieve better quality. Thus, the following directions are pointed out so that this research can be further pursued.

- **Improving the sentence embedding algorithm**: In this research, Algorithm 1 has been used to generate embedding for sentences. This algorithms assigns an embedding by computing the mean of the word vectors of the words in the sentence. One way to get a better sentence embedding is to train and apply a deep learning algorithm.

- **Improving the coherence algorithm**: The coherence algorithm, Algorithm 6 which is designed to determine the coherence of candidate labels of a given topic, is dependent on the word embedding model used. This is due to the fact that the algorithm works by calculating cosine distance which requires words of the labels to exist in the word embedding model's vocabulary. If an embedding cannot be generated for a label, then it will be assigned a coherence value of 0 which is the lowest coherence value making the candidate label less important to the given topic. However, not having a label in a word embedding model's vocabulary doesn't always guarantee that the label is the least important. It may be the case that the label is a newly emerging term or the vocabulary is limited. Thus, it is one direction to further do a research on to design a better coherence algorithm that avoids this issue.

- **Creating correlation among topics**: The system proposed in this thesis work is a single-transcript/document topic detection. If a user uploads a set of transcripts, it processes each transcript separately and returns the topics generated for each of them. However, sometimes users may get interested in finding the correlation among the topics generated from the entire corpus. Therefore, it may seem vital to find some kind of hierarchical or flat semantic relation between the topics of the transcripts.

- **Generate topic labels hierarchically**: The current topic labeling algorithm generates labels with no relationship between them. Finding labels with hierarchical relations helps to identify more generic and specific labels. This can be one area to do further research in topic labeling.

- **Considering terms outside the topic sentences as labels for the topic**: The algorithm designed to generate candidate labels considers only the terms that exist in the topic sentences. This may lead to missing important labels which do not exist in the topic sentences but capture the intended meaning of the topic. Thus, it will be more efficient to extract terms from the topic sentences and find their common super concepts from external knowledge bases to have them as additional candidate labels.

- **Using topic labels to further optimize topic detection**: Once topic labels are identified, using these labels to further optimize the topic labeling algorithm, specifically the DONT algorithm, will improve the quality of the topics that can be detected by the system. One way to achieve this is to group together topics that share similar labels.

References

[1] Dipanjan Sarkar. *Text Analytics with Python: A Practical Real-world Approach to Gaining Actionable Insights from Your Data*. Apress, 2016.

[2] Shixia Liu et al. "Interactive, topic-based visual text summarization and analysis". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pp. 543–552.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[4] Jey Han Lau et al. "Automatic labelling of topic models". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 1536–1545.

[5] Xiaojun Wan and Tianming Wang. "Automatic Labeling of Topic Models Using Text Summaries." In: *ACL (1)*. 2016.

[6] Mehdi Allahyari and Krys Kochut. "Automatic topic labeling using ontology-based topic models". In: *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE. 2015, pp. 259–264.

[7] Xianling Mao et al. "A Novel Fast Framework for Topic Labeling Based on Similarity-preserved Hashing." In: *COLING*. 2016, pp. 3339–3348.

[8] Pucktada Treeratpituk and Jamie Callan. "Automatically labeling hierarchical clusters". In: *Proceedings of the 2006 international conference on Digital government research*. Digital Government Society of North America. 2006, pp. 167–176.

[9] Pierre Ficamos and Yan Liu. "A Topic based Approach for Sentiment Analysis on Twitter Data". In: *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS* 7.12 (2016), pp. 201–205.

[10] Ehsaneddin Asgari and Jean-Cédric Chappelier. "Linguistic Resources and Topic Models for the Analysis of Persian Poems." In: *CLfL@ NAACL-HLT*. 2013, pp. 23–31.

[11] Joseph Turian, Lev Ratinov, and Yoshua Bengio. "Word representations: a simple and general method for semi-supervised learning". In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 384–394.

[12] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 2003, pp. 133–142.

[13] Amit Mandelbaum and Adi Shalev. "Word Embeddings and Their Use In Sentence Classification Tasks". In: *arXiv preprint arXiv:1610.08229* (2016).

[14] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[15] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[16] Roy Schwartz, Roi Reichart, and Ari Rappoport. "Symmetric Pattern Based Word Embeddings for Improved Word Similarity Prediction." In: *CoNLL*. Vol. 2015. 2015, pp. 258–267.

[17] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[18] Jon Ezeiza Alvarez and Hannah Bast. "A review of word embedding and document similarity algorithms applied to academic text". PhD thesis. University OF Freiburg, 2017.

[19] Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *arXiv preprint arXiv:1607.04606* (2016).

[20] Shihao Ji et al. "Wordrank: Learning word embeddings via robust ranking". In: *arXiv preprint arXiv:1506.02761* (2015).

[21] Manaal Faruqui et al. "Retrofitting word vectors to semantic lexicons". In: *arXiv preprint arXiv:1411.4166* (2014).

[22] Wen-tau Yih et al. "Learning discriminative projections for text similarity measures". In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics. 2011, pp. 247–256.

[23] Samuel Gershman and Joshua B Tenenbaum. "Phrase similarity in humans and machines." In: *CogSci*. 2015.

[24] Tom Kenter and Maarten De Rijke. "Short text similarity with word embeddings". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM. 2015, pp. 1411–1420.

[25] Tom Kenter, Alexey Borisov, and Maarten de Rijke. "Siamese cbow: Optimizing word embeddings for sentence representations". In: *arXiv preprint arXiv:1606.04640* (2016).

[26] Lei Guo, Chong Li, and Haiming Tian. "Duplicate Quora Questions Detection". In: (2017).

[27] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014, pp. 1188–1196.

[28] Minmin Chen. "Efficient vector representation for documents through corruption". In: *arXiv preprint arXiv:1707.02377* (2017).

[29] Matt Kusner et al. "From word embeddings to document distances". In: *International Conference on Machine Learning*. 2015, pp. 957–966.

[30] Ryan Kiros et al. "Skip-thought vectors". In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[31] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. "Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features". In: *arXiv preprint arXiv:1703.02507* (2017).

[32] Ankur P Parikh et al. "A decomposable attention model for natural language inference". In: *arXiv preprint arXiv:1606.01933* (2016).

[33] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification." In: *HLT-NAACL*. 2016, pp. 1480–1489.

[34] Wael H Gomaa and Aly A Fahmy. "A survey of text similarity approaches". In: *International Journal of Computer Applications* 68.13 (2013).

[35] Archana Singh, Avantika Yadav, and Ajay Rana. "K-means with Three different Distance Metrics". In: *International Journal of Computer Applications* 67.10 (2013).

[36] Shraddha K Popat and M Emmanuel. "Review and comparative study of clustering techniques". In: *International journal of computer science and information technologies* 5.1 (2014), pp. 805–812.

[37] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[38] Leonard Kaufman and Peter J Rousseeuw. "Partitioning around medoids (program pam)". In: *Finding groups in data: an introduction to cluster analysis* (1990), pp. 68–125.

[39] Leonard Kaufman and Peter J Rousseeuw. "Clustering large applications (Program CLARA)". In: *Finding groups in data: an introduction to cluster analysis* (2008), pp. 126–163.

[40] Saurabh Arora and Inderveer Chana. "A survey of clustering techniques for big data analysis". In: *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-*. IEEE. 2014, pp. 59–65.

[41] Martin Ester and Hans-Peter Kriegel. "Jörg SANDER a Xiaowei XU". In: *A Density-Based Algorithm for Discovering Clusters: in Large Spatial Databases with Noise* (1996).

[42] Mihael Ankerst et al. "OPTICS: ordering points to identify the clustering structure". In: *ACM Sigmod record*. Vol. 28. 2. ACM. 1999, pp. 49–60.

[43] Alexander Hinneburg and Hans-Henning Gabriel. "Denclue 2.0: Fast clustering based on kernel density estimation". In: *IDA*. Vol. 7. Springer. 2007, pp. 70–80.

[44] Alexander Hinneburg, Daniel A Keim, et al. "An efficient approach to clustering in large multimedia databases with noise". In: *KDD*. Vol. 98. 1998, pp. 58–65.

[45] M Kuchaki Rafsanjani, Zahra Asghari Varzaneh, and N Emami Chukanlo. "A survey of hierarchical clustering algorithms". In: *The Journal of Mathematics and Computer Science* 5.3 (2012), pp. 229–240.

[46] Fionn Murtagh and Pierre Legendre. "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?" In: *Journal of Classification* 31.3 (2014), pp. 274–295. ISSN: 1432-1343. DOI: 10.1007/s00357-014-9161-z. URL: https://doi.org/10.1007/s00357-014-9161-z.

[47] Robert L Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (1953), pp. 267–276.

[48] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 2009.

[49] Robert Tibshirani, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[50] LV Bijuraj. "Clustering and its Applications". In: *Proceedings of National Conference on New Horizons in IT-NCNHIT*. 2013, p. 169.

[51] Young-Woo Seo and Katia Sycara. *Text clustering for topic detection*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2004.

[52] Xiaohui Huang et al. "A topic detection approach through hierarchical clustering on concept graph". In: *Applied Mathematics & Information Sciences* 7.6 (2013), p. 2285.

[53] Nicola Guarino et al. "Formal ontology and information systems". In: *Proceedings of FOIS*. Vol. 98. 1998. 1998, pp. 81–97.

[54] Tom Gruber. "What is an Ontology". In: *WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)* (1993).

[55] W3C. *SPARQL Query Language for RDF*. `http://www.w3.org/TR/rdf-sparql-query/`. [Online; accessed 10-January-2018]. 2008.

[56] Dejing Dou, Hao Wang, and Haishan Liu. "Semantic data mining: A survey of ontology-based approaches". In: *Semantic Computing (ICSC), 2015 IEEE International Conference on*. IEEE. 2015, pp. 244–251.

[57] M Allahyari and K Kochut. *OntoLDA: An Ontology-based Topic Model for Automatic Topic Labeling*. 2009.

[58] Kino Coursey and Rada Mihalcea. "Topic identification using Wikipedia graph centrality". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*. Association for Computational Linguistics. 2009, pp. 117–120.

[59] James Allan et al. "Topic detection and tracking pilot study final report". In: (1998).

[60] Frederick Walls et al. "Topic detection in broadcast news". In: *Proceedings of the DARPA broadcast news workshop*. Morgan Kaufmann Publishers, Inc. 1999, pp. 193–198.

[61] Juan Cigarrán, Ángel Castellanos, and Ana García-Serrano. "A step forward for Topic Detection in Twitter: An FCA-based approach". In: *Expert Systems with Applications* 57 (2016), pp. 21–36.

[62] Thomas Hofmann. "Probabilistic latent semantic indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 50–57.

[63] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), p. 391.

[64] J Michael Schultz and Mark Liberman. "Topic detection and tracking using idf-weighted cosine coefficient". In: *Proceedings of the DARPA broadcast news workshop*. San Francisco: Morgan Kaufmann. 1999, pp. 189–192.

[65] Shu-Wei Liu and Hsien-Tsung Chang. "A topic detection and tracking system with TF-Density". In: *Recent Progress in Data Engineering and Internet Technology*. Springer, 2013, pp. 115–120.

[66] Carlos N Silla Jr, Celso AA Kaestner, and Alex A Freitas. "A non-linear topic detection method for text summarization using wordnet". In: *Workshop of Technology Information Language Human (TIL'2003)*. Vol. 24. 2003.

[67] Christian Wartena and Rogier Brussee. "Topic detection by clustering keywords". In: *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*. IEEE. 2008, pp. 54–58.

[68] A Castellanos, J Cigarrán, and A García-Serrano. "Formal concept analysis for topic detection: a clustering quality experimental analysis". In: *Information Systems* 66 (2017), pp. 24–42.

[69] Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. "Simple semantics in topic detection and tracking". In: *Information retrieval* 7.3-4 (2004), pp. 347–368.

[70] Masaki Mori, Takao Miura, and Isamu Shioya. "Topic detection and tracking for news web pages". In: *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*. IEEE. 2006, pp. 338–342.

[71] Qian Chen, Xin Guo, and Hexiang Bai. "Semantic-based topic detection using Markov decision processes". In: *Neurocomputing* 242 (2017), pp. 40–50.

[72] Tengfei Liu, Nevin L Zhang, and Peixian Chen. "Hierarchical latent tree analysis for topic detection". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pp. 256–272.

[73] Peixian Chen et al. "Latent tree models for hierarchical topic detection". In: *Artificial Intelligence* 250 (2017), pp. 105–124.

[74] Peixian Chen et al. "Progressive EM for Latent Tree Models and Hierarchical Topic Detection." In: *AAAI*. 2016, pp. 1498–1504.

[75] Damiano Spina, Julio Gonzalo, and Enrique Amigó. "Learning similarity functions for topic detection in online reputation monitoring". In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM. 2014, pp. 527–536.

[76] Yan Chen et al. "Emerging topic detection for organizations from microblogs". In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2013, pp. 43–52.

[77] Karl Grieser et al. "Using ontological and document similarity to estimate museum exhibit relatedness". In: *Journal on Computing and Cultural Heritage (JOCCH)* 3.3 (2011), p. 10.

[78] Fionn Murtagh and Pierre Legendre. "Wards hierarchical agglomerative clustering method: which algorithms implement Wards criterion?" In: *Journal of classification* 31.3 (2014), pp. 274–295.

[79] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[80] Jrn Hees. *SciPy Hierarchical Clustering and Dendrogram Tutorial*. `https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/`. 2015 (accessed October 10, 2017).

[81] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[82] Gareth Dwyer. *Flask vs. Django: Why Flask Might Be Better*. https://www.codementor.io/garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v. 2017 (accessed March 20, 2017).

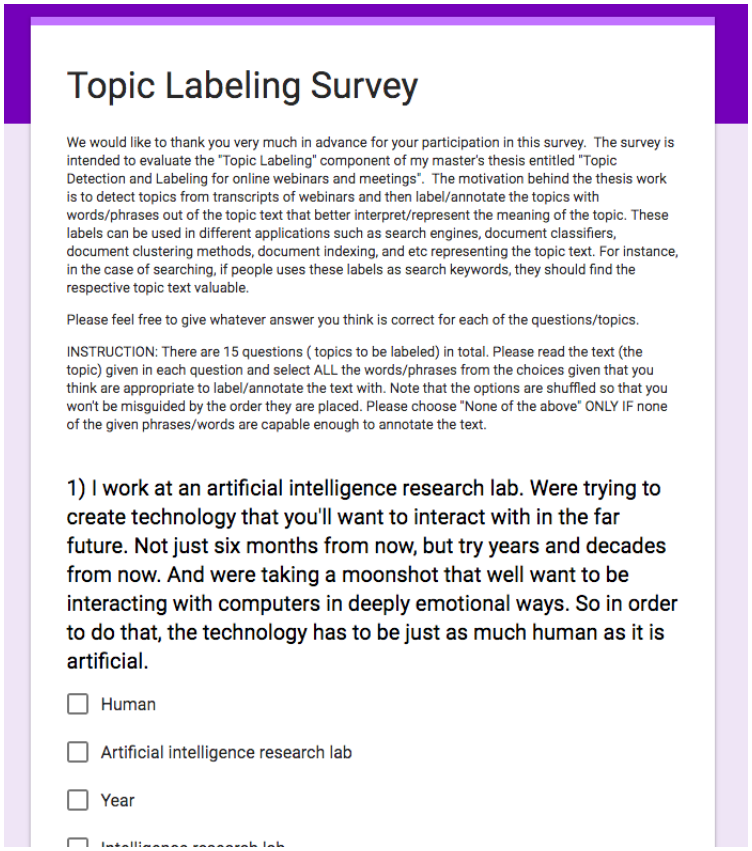
[83] Ying Zhao, George Karypis, and Ding-Zhu Du. *Criterion functions for document clustering*. University of Minnesota, 2005.

[84] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press, 2008.

[85] Enrique Amigó et al. "A comparison of extrinsic clustering evaluation metrics based on formal constraints". In: *Information retrieval* 12.4 (2009), pp. 461–486.

[86] Andrew Rosenberg and Julia Hirschberg. "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure." In: *EMNLP-CoNLL*. Vol. 7. 2007, pp. 410–420.

A. Appendix



Topic Labeling Survey

We would like to thank you very much in advance for your participation in this survey. The survey is intended to evaluate the "Topic Labeling" component of my master's thesis entitled "Topic Detection and Labeling for online webinars and meetings". The motivation behind the thesis work is to detect topics from transcripts of webinars and then label/annotate the topics with words/phrases out of the topic text that better interpret/represent the meaning of the topic. These labels can be used in different applications such as search engines, document classifiers, document clustering methods, document indexing, and etc representing the topic text. For instance, in the case of searching, if people uses these labels as search keywords, they should find the respective topic text valuable.

Please feel free to give whatever answer you think is correct for each of the questions/topics.

INSTRUCTION: There are 15 questions (topics to be labeled) in total. Please read the text (the topic) given in each question and select ALL the words/phrases from the choices given that you think are appropriate to label/annotate the text with. Note that the options are shuffled so that you won't be misguided by the order they are placed. Please choose "None of the above" ONLY IF none of the given phrases/words are capable enough to annotate the text.

1) I work at an artificial intelligence research lab. Were trying to create technology that you'll want to interact with in the far future. Not just six months from now, but try years and decades from now. And were taking a moonshot that well want to be interacting with computers in deeply emotional ways. So in order to do that, the technology has to be just as much human as it is artificial.

- [ ] Human
- [ ] Artificial intelligence research lab
- [ ] Year

Figure A.1.: Part of the first page of the survey used to collect human judgment for topic labeling evaluation

14) Former UN Secretary Kofi Annan has spoken candidly about his personal failure leading to the Rwandan genocide. In 1994, he was head of the UN peacekeeping department. At a 10-year memorial for the genocide, he reflected, I believed at the time I was doing my best, but I realized after the genocide that there was more I could and should have done to sound the alarm and rally support. The AIDS epidemic caught the health community unprepared, and today, when the World Health Organization estimates that 39 million people have lost their lives to this disease, I'm not alone in feeling remorse and regret at not having done more earlier. But while living in Zimbabwe, I didn't see my role as an advocacy or a political one. I was there for my technical skills, both my clinical and my research epidemiology skills. And in my mind, my job was to take care of patients and to do research to better understand the population patterns of transmission, and I hoped that we'd slow the spread of the virus.

15 responses

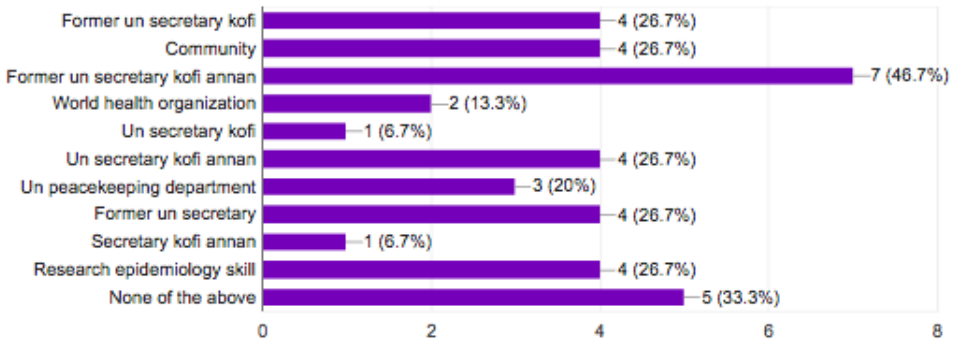


Figure A.2.: The user responding rate for topic number fourteen