





MASTER'S THESIS IN COMPUTER SCIENCE: ALGORITHMS  
LANGUAGES AND LOGIC

**Topic Modeling and Clustering for Analysis of  
Road Traffic Accidents**

AGAZI MEKONNEN  
SHAMSI ABDULLAYEV



Department of Applied Mechanics  
Division of Vehicle Safety  
Accident Prevention Group  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Göteborg, Sweden 2017

Topic Modeling and Clustering for Analysis of Road Traffic Accidents  
AGAZI MEKONNEN  
SHAMSI ABDULLAYEV

© AGAZI MEKONNEN, SHAMSI ABDULLAYEV, 2017

Supervisor: Selpi, Department of Applied Mechanics  
Supervisor: Jordanka Kovaceva, Department of Applied Mechanics  
Examiner: Selpi, Department of Applied Mechanics

Master's Thesis 2017:65  
ISSN 1652-8557  
Department of Applied Mechanics  
Division of Vehicle Safety  
Accident Prevention Group  
Chalmers University of Technology  
SE-412 96 Göteborg  
Sweden  
Telephone +46 (0)31-772 1000

Cover:  
Visualisation of word embeddings.

Department of Applied Mechanics  
Göteborg, Sweden 2017

Topic Modeling and Clustering for Analysis of Road Traffic Accidents

AGAZI MEKONNEN

SHAMSI ABDULLAYEV

Division of Vehicle Safety

Department of Applied Mechanics

Chalmers University of Technology

## **Abstract**

In this thesis, we examined different approaches on how to cluster, summarise and search accident descriptions in Swedish Traffic Accident Data Acquisition (STRADA) dataset. One of the central questions in this project was that how to retrieve similar documents if a query does not have any common words with relevant documents. Another question is how to increase similarity between documents which describe the same or similar scenarios in different words. We designed a new pre-processing technique using keyword extraction and word embeddings to address these issues. Theoretical and empirical results show the pre-processing technique employed improved the results of the examined topic modeling, clustering and document ranking methods.

Keywords: Machine Learning, Latent Dirichlet Allocation, Clustering, Probabilistic Topic Models, Text Mining, Traffic Safety, Accident database

## Acknowledgements

Foremost, we would like to thank our supervisor and examiner Dr. Selpi for her continuous support, motivation, patience and recommendations throughout the project. Besides, we are thankful to our supervisor Jordanka Kovaceva who helped us understand more regarding traffic safety and for her major input in shaping the report.

We would like to thank SAFER for providing us with a suitable working environment for carrying out our thesis and presenting the opportunity to extend our knowledge of traffic and vehicle safety.

Our deepest gratitude goes to our family, friends and everyone who was there supporting us.

Last but not the least, we would like to thank the Swedish Institute and Chalmers University of Technology for providing us with a scholarship opportunity to this Master's program.

Agazi Mekonnen, Shamsi Abdullayev  
June 2017



# Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Goals and Challenges . . . . .	2
1.3 Scope . . . . .	2
1.4 Thesis Outline . . . . .	2
<b>2 Text Data</b>	<b>3</b>
2.1 The Datasets . . . . .	3
2.2 From Text to Vectors . . . . .	3
2.2.1 Term Frequency . . . . .	3
2.2.2 Inverse Document Frequency . . . . .	4
2.2.3 TFIDF Weighting . . . . .	4
2.3 Word Embeddings . . . . .	4
2.3.1 Co-occurrence Matrix . . . . .	5
2.3.2 Word2Vec: Distributed Representation of Words . . . . .	5
2.4 Similarity Measures . . . . .	7
2.4.1 Euclidean Distance . . . . .	7
2.4.2 Cosine Similarity . . . . .	8
2.4.3 Word Mover’s Distance . . . . .	8
2.5 Pre-processing . . . . .	9
2.5.1 Stop-word Removal . . . . .	9
2.5.2 Stemming . . . . .	9
2.5.3 Lemmatisation . . . . .	10
2.5.4 Pruning Rare Terms . . . . .	10
<b>3 Methods</b>	<b>11</b>
3.1 K-means . . . . .	11
3.1.1 K-means++ . . . . .	12
3.1.2 Choosing the Number of Clusters . . . . .	13
3.2 Probabilistic Topic Models . . . . .	14
3.2.1 Latent Dirichlet Allocation . . . . .	14
3.3 Keyword Expansion Approach . . . . .	15



---

3.4	Dimensionality Reduction . . . . .	16
3.5	Evaluation Metrics . . . . .	17
3.5.1	F-Measure . . . . .	17
3.5.2	Rand Index . . . . .	18
3.5.3	Silhouette Index . . . . .	18
3.5.4	Calinski-Harabasz index . . . . .	18
3.5.5	Evaluation of Probabilistic Topic Models . . . . .	19
<b>4</b>	<b>Experiments and Results</b>	<b>20</b>
4.1	The Datasets . . . . .	20
4.2	Pre-processing . . . . .	21
4.2.1	Stop-word Removal . . . . .	21
4.2.2	Lemmatisation . . . . .	21
4.2.3	Pruning Rare Terms . . . . .	21
4.2.4	How Pre-processing Affects Dimensionality . . . . .	21
4.3	Algorithms . . . . .	22
4.3.1	Latent Dirichlet Allocation . . . . .	22
4.3.2	K-means . . . . .	23
4.3.3	Word Embeddings . . . . .	23
4.3.4	Keyword Extraction . . . . .	24
4.3.5	Enriching the Data . . . . .	24
4.4	Selecting the Number of Clusters and Topics . . . . .	25
4.5	Evaluation . . . . .	26
4.5.1	Clustering Evaluation . . . . .	26
4.5.2	LDA Evaluation . . . . .	28
4.5.3	Document Ranking Evaluation . . . . .	32
4.6	Discussion . . . . .	33
<b>5</b>	<b>Conclusion</b>	<b>35</b>
5.1	Future work . . . . .	36
<b>A</b>	<b>LDA topics</b>	<b>41</b>
<b>B</b>	<b>Code Listings</b>	<b>47</b>
<b>C</b>	<b>Perplexity values for different topics</b>	<b>52</b>

# List of Figures

2.1	The skip-gram model with window size $\mathcal{C}$ . . . . .	6
2.2	Euclidean distance between document vectors . . . . .	8
3.1	K-means at local optimum. The different shapes (circle, square, triangle) indicate different clusters and the centroids are denoted by '+'. . . . . .	12
3.2	Elbow Curve . . . . .	13
3.3	Data enrichment pipe-line . . . . .	15
3.4	Projecting 2D data onto one-dimensional space . . . . .	17
4.1	PCA reduced test data before and after enrichment . . . . .	25
4.2	Elbow curves for dataset $P$ and $H$ . . . . .	25
4.3	Perplexity values for choosing the number of topics . . . . .	26
4.4	Confusion matrix for k-means with bigrams on $T$ . . . . .	27
4.5	Confusion matrix for k-means on enriched $T$ . . . . .	28
4.6	Perplexity results on $H$ and $H_E$ . . . . .	29
4.7	Perplexity results on $P$ and $P_E$ . . . . .	29

# List of Tables

2.1	Co-occurrence Matrix . . . . .	5
4.1	The number of documents and unique words in dataset $P$ and $H$ . .	20
4.2	Gold standard clusters from $T$ . . . . .	20
4.3	Dimensionality reduction with different pre-processing techniques . .	22
4.4	K-means parameters . . . . .	23
4.5	Words that are identified as related to "korsning" ('intersection') and "seriekrock" ('pileup') by word embeddings . . . . .	23
4.6	Parameters for word embedding . . . . .	24
4.7	Parameters keyword extraction . . . . .	24
4.8	Evaluation of different algorithms on test set $T$ . . . . .	27
4.9	Evaluation with Calinski-Harabasz index on Dataset $P$ and $H$ . . . .	28
4.10	Sample topics for $P_E$ . . . . .	30
4.11	Sample topics for $H_E$ . . . . .	31
4.12	Evaluation of different ranking algorithms on test set $T$ . . . . .	32
4.13	Evaluation of different ranking algorithms with no occurrence of query words in relevant documents . . . . .	32
4.14	Running time of different ranking algorithms for a single query in MacBook Air with processor 2,2 GHz Intel Core i7 and Memory 8GB	33
A.1	LDA topics for $H$ . Top 10 most probable words are provided for each topic . . . . .	41
A.2	LDA topics for $H_E$ . Top 10 most probable words are provided for each topic . . . . .	42
A.3	LDA topics for $P$ . Top 10 most probable words are provided for each topic . . . . .	43
A.4	LDA topics for $P_E$ . Top 10 most probable words are provided for each topic . . . . .	45
C.1	Perplexity for $P$ , $H$ , $P_E$ , and $H_E$ . . . . .	52

# 1

## Introduction

### 1.1 Motivation

The ever-increasing digital data available today in the form of text has created a gap between availability and usage. This vastness of the data makes finding, searching, retrieving and summarising of information very cumbersome and challenging. For instance, given a massive collection of documents, one might ask for the main idea or theme of the text data. Therefore tools capable of searching, organising, and summarising large collection of text documents are in demand.

Swedish Traffic Accident Data Acquisition (STRADA) is a relational database system which is used by Swedish Transport Administration (STA) starting from the year 2003 [17], to keep a record of all traffic accidents in Sweden. Every accident registered in STRADA contains an accident description. Accident description attribute consists of information about specific road traffic accident by the police and hospital. It is inevitable that different people will describe similar traffic accidents in different words and use different expressions. For instance, to write accident description about a traffic accident that occurred because of an icy road, various persons could write it differently. One person might write, "an icy road was the cause of the accident", another person might comment "last night's massive snowfall" as the the cause traffic accident, yet another could write, "frozen road" as an accident description. These text data describe similar traffic accidents and can be categorised under the same topic, say, icy road. However, if we try to search an accident that occurred because of an icy road in the current system, we will only hit one result instead of three.

Currently, one way of searching for relevant accidents is string matching using keywords. This way of searching is not effective because rich data is lost [17]. According to [17], finding the desired traffic accidents is challenging. Therefore, the problem of finding similar kind of traffic accidents has created a challenge for traffic accident analysis and research [17]. One cannot help in reducing traffic accident without having the full picture of the accidents taking place in the country. To lessen the frequency of road traffic accidents a thorough data analysis and acting upon the result of the data analysis is vital.

## 1.2 Goals and Challenges

This project aims at exploring the recent advancements in the area of text mining and finding out how these advancements can be used for a short text collection.

The goals of this project include:

- Finding a suitable way of clustering accident descriptions.
- Summarising the data using topic models.
- Finding a suitable document ranking algorithm which can work well on short text collection.

The central challenge of this project is grouping similar scenarios to the same cluster although they have been expressed in different ways or retrieving the correct document set even though the query does not have any common word with relevant documents. The short documents make it harder for document ranking, topic modeling and clustering algorithms to produce an acceptable result due to lack of information or sparsity. Hence, dealing with the sparsity of document vectors is another challenge.

## 1.3 Scope

The scope includes using the existing algorithms to achieve the goals described in the previous section. If the existing algorithms do not produce good results, some adjustment may be made either in pre-processing step or to the algorithms. However, the thesis does not include developing a new algorithm from scratch.

The data for experiments are limited to only accident descriptions of STRADA database. The accident descriptions include some grammatical and spelling errors. The correction of these errors will not be considered here.

## 1.4 Thesis Outline

The rest of this thesis is outlined as follows. Chapter 2 describes the datasets, vector representation of documents, word embeddings, document level similarities and finally, it provides information about pre-processing techniques. Chapter 3 describes topic modelling, clustering algorithms and evaluation techniques for these algorithms. Moreover, we describe our approach in this chapter. In Chapter 4, we present the test results with different pre-processing techniques. This chapter also includes discussion of the results. Finally, in Chapter 5 we summarise our work and discuss future work.

# 2

## Text Data

In this chapter, we discuss how to represent documents and words as a vector, pre-processing techniques and similarity measures between a document and word pairs.

### 2.1 The Datasets

The datasets used for this project are acquired from STRADA information system. Every recorded accident in STRADA has attributes such as report id, type of accident, the location of the accident by coordinates, accident description, date of registration, etc. These accident reports are recorded by police and hospital from the year 2003 to the year 2015.

There are two datasets which will be analysed in this project. The first dataset is accident descriptions recorded by the police. It contains 223574 descriptions. The second dataset is accident description recorded by hospitals, and it contains 448476 descriptions. All these descriptions are very short ranging from 1 to 5 sentences. The accident descriptions will be referred as documents throughout this report.

### 2.2 From Text to Vectors

In pre-processing phase, documents have to be transformed from raw text into an understandable form for algorithms. The trivial idea is constructing a unique vocabulary from the text corpus. The vectorized document representation is a vector which has a length of vocabulary, and each value in the vector is the number of occurrences of a word corresponding to its index. However, this simple model cannot say anything about the importance of words in a document. The old yet powerful technique is TFIDF weighting [12] which penalises insignificant words in documents. The detailed explanation about TFIDF is given in Sections 2.2.1 - 2.2.2.

#### 2.2.1 Term Frequency

Term frequency calculates how frequent a word occurs in a document. It is a vector which has a size of vocabulary and contains a statistics about the number of occurrences of each word in a document. The Equation for computing term frequency of term  $t$  in document  $d$  is given below.  $tf_{t,d}$  is a number of occurrences of term  $t$  in

document  $d$  [11].

$$tf(t, d) = \begin{cases} 1 + \log(tf_{t,d}) & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

Another technique for term frequency is Boolean frequency where  $tf(t, d) = 1$  when word occurs in a document, 0 otherwise. This method is usually used for short documents [24]. The logarithm function is applied to  $tf_{t,d}$  because the relevance is not linearly related to the number of occurrences of a word in a document.

### 2.2.2 Inverse Document Frequency

Inverse document frequency is computed by the following equation [11].

$$idf_t = \log(\mathcal{N}/df_t) \quad (2.2)$$

where  $df_t$  is the number of documents which contain word  $t$  and  $\mathcal{N}$  is the size of the corpus. The words which occur in most of documents are going to have low IDF score, because these words are not specific to a particular document. Words occurring in small number of documents will have high IDF score. Without the logarithmic scale, word occurring only in one document is going to have inverse document frequency of  $\mathcal{N}$  which is not correct relation. Therefore this value is scaled using logarithm function.

### 2.2.3 TFIDF Weighting

After computing the term frequency and inverse document frequency, the final weight of the term  $t$  in document  $d$  is computed by Equation 2.3 which is called TFIDF weight of term  $t$  in document  $d$

$$w(t, d) = tf(t, d) \times idf_t \quad (2.3)$$

In order to compute the whole TFIDF matrix, the computation 2.3 is basically done for each document and for each word in the vocabulary.

## 2.3 Word Embeddings

*You shall know the word by the company it keeps*

— John Rupert Firth

Machine learning algorithms require inputs as a fixed length feature. One of most popular fixed-length feature for text is bag-of-words. However, this method completely ignores order and semantics of the words. According to the bag-of-words approach, the words "crash", "accident" and "car" are equally distinct [16]. However, in the word embedding each word is expressed by a vector and similar words end up with similar vectors. Word embedding algorithms capture this similarity from the context of the word. Therefore, this approach is much more powerful than traditional representation such as TFIDF.

### 2.3.1 Co-occurrence Matrix

The co-occurrence matrix contains co-occurrence values for each word in a corpus according to a window size. The window size of  $k$  will take  $k$  words before and  $k$  words after the given word as a context. For a text corpus and a window size, the co-occurrence matrix can be computed by looking at context words of the given word <sup>1</sup>. Assume a corpus contains the following sentences:

("I like Machine Learning", "I enjoy NLP", "I like deep learning")

Table 2.1 shows the co-occurrence matrix for above corpus with windows size 2.

**Table 2.1:** Co-occurrence Matrix

Vocabulary	i	like	enjoy	machine	learning	NLP	deep
i	0	2	1	1	0	1	1
like	2	0	0	1	2	0	1
enjoy	1	0	0	0	0	0	0
machine	1	1	0	0	1	0	0
learning	0	2	0	1	0	0	1
NLP	1	0	1	0	0	0	0
deep	1	1	0	0	1	0	0

From the co-occurrence matrix we can see that similar words end up with similar vectors. However, computing the co-occurrence matrix is very inefficient for large corpus. Another disadvantage is that the dimensionality of the matrix is very high and machine learning algorithms cannot handle this. Therefore dimensionality reduction (e.g. Singular Value Decomposition) is used on this matrix. If a new document is added to the corpus, the whole dimensionality reduction has to be applied to the new corpus which has the complexity of  $O(mn^2)$  for an  $m \times n$  matrix. Therefore, this kind of approach is not practical for large text corpus.

### 2.3.2 Word2Vec: Distributed Representation of Words

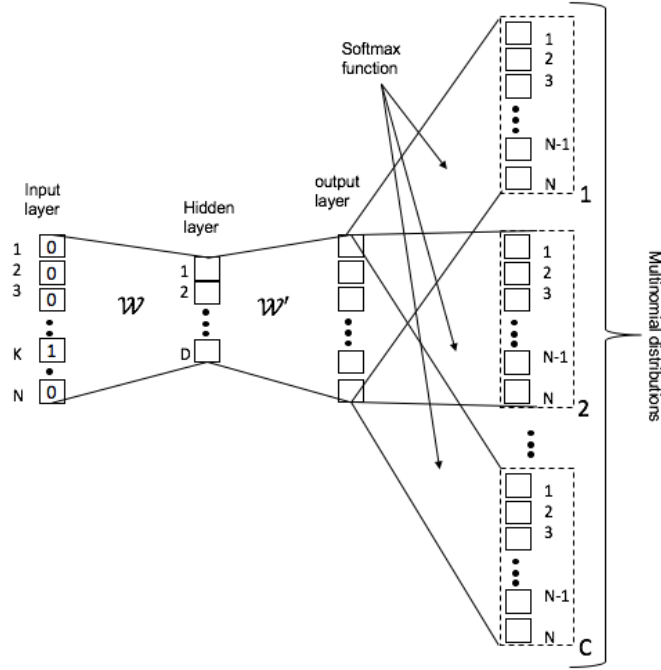
Word2Vec is a model which learns word vectors from a large text corpus using two-layer shallow neural network [28]. CBOW (Continuous bag-of-words model) and skip-gram are two types of Word2Vec. The skip-gram model takes a word as an input and predicts the context words. However, the CBOW model takes context words and predicts a word which is most likely to be in that context. This is used in search engine predictions. The input and output words are represented as one-hot vector. For instance, the unique vocabulary of text corpus contains  $N$  words. The  $k$ th word is represented as an  $N$  dimensional vector with all zeros except the position which corresponds to  $k$ th word.

In skip-gram model, a one-hot vector with  $N$  dimensions is fed to two layers neural network which has  $D$  nodes in hidden layer and  $C$  output layer, where  $C$  is a window size (the algorithm takes context words as  $C$  words before or after the central word). The weight vector from the input layer to the hidden layer is  $N \times K$  dimensional

<sup>1</sup>[https://cs224d.stanford.edu/lecture\\_notes/notes1.pdf](https://cs224d.stanford.edu/lecture_notes/notes1.pdf)



matrix. Each row of this matrix contains vector representation of the corresponding word. These vectors are learned step-by-step using gradient descent. Thus, the model learns context words and the vector representation of words at the same time. The similar words can be found using cosine similarity on the weight matrix. Figure 2.1 shows the skip-gram model with  $\mathcal{C}$  words in context. There are two weight



**Figure 2.1:** The skip-gram model with window size  $\mathcal{C}$

matrices  $\mathcal{W}$  and  $\mathcal{W}'$ . The weight matrices are initialised to very small random values before training the algorithm. The vector for hidden layer is computed by Equation 2.4.

$$h = \mathcal{W}^T x \quad (2.4)$$

where  $x$  is  $\mathcal{N}$  dimensional one-hot encoded vector which represents the input word. Since all values of  $x$  are zero except the one which corresponds to the  $k^{th}$  position, the vector  $h$  will contain the  $k^{th}$  row of  $\mathcal{W}$ . In the next step, a score for each word in vocabulary is computed by Equation 2.5.

$$u = \mathcal{W}'^T h \quad (2.5)$$

The  $\mathcal{N}$  dimensional vector  $u$  contains a score for each word in the vocabulary. Equations 2.4 and 2.5 describe the forward-propagation process of neural network. After forward-propagation, the values in vector  $u$  are fed to soft-max function to turn the scores into probabilities.

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{j=1}^{\mathcal{N}} \exp(u_j^T v_c)} \quad (2.6)$$

where  $o$  is the index of outside word within a window size,  $c$  is the index of center or the input word,  $u_o$  is the vector representation of word in index  $o$  and  $v_c$  is the vector

representation of center word. The objective of skip-gram model is maximising the probability of a word in any context. This probability is calculated by Equation 2.7.

$$\prod_{t=1}^T \prod_{-k \leq j \leq k} p(w_{t+j}|w_t) \quad (2.7)$$

where  $w_i$  is any word in the vocabulary. Let us now simplify Equation 2.7 using log likelihood. Equation 2.8 is the simplified version of 2.7 and maximising 2.7 is the same as minimising 2.8. [18].

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-k \leq j \leq k} \log p(w_{t+j}|w_t) \quad (2.8)$$

Now we have the cost function. In back-propagation we have to use derivatives to optimise word vectors. If we get the derivative of 2.6 we will end up with Equation 2.9.

$$u_o - \sum_{x=1}^T \frac{\exp(u_x^T v_c)}{\sum_{w=1}^T \exp(u_w^T v_c)} u_x \quad (2.9)$$

Equation 2.9 has to be computed in each gradient descent update. Therefore, it is computationally impractical. In practice, negative sampling and hierarchical softmax is used to approximate the actual value.

## 2.4 Similarity Measures

The similarity measure is a metric for calculating the similarity of objects. Sometimes this can be Euclidean distance, the similarity between images or similarity between two sentences. For latter case, the similarity is calculated using term frequency and inverse document frequency or more advanced technique which is described in Section 2.4.3.

### 2.4.1 Euclidean Distance

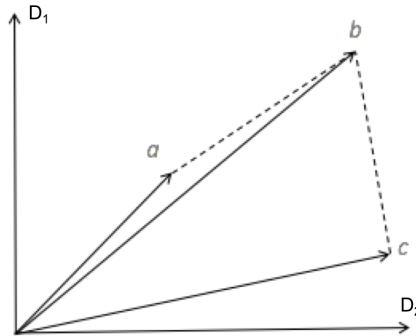
Euclidean distance is a distance between two points in space. It is also used extensively to compute the distance between two text documents. In this case, the distance is computed using TFIDF vectors of documents. Given two text documents  $d_a$  and  $d_b$  and their TFIDF vectors  $t_a$  and  $t_b$  the Euclidean distance is computed by the following equation [11].

$$\mathcal{D}_e(t_a, t_b) = \sqrt{\sum_{t=1}^m |t_a - t_b|^2} \quad (2.10)$$

where  $m$  is vocabulary size.

The figure 2.2 shows distance between document vectors  $a$ ,  $b$  and  $c$ . In this distance metric the vector  $b$  is equally distinct from vector  $a$  and  $c$ . But the documents  $a$  and  $b$  are related to  $D_1$ . On the other hand, the document  $c$  is related to  $D_2$ . Therefore,

the distance between  $a$  and  $b$  should not be equal to the distance between  $b$  and  $c$ . This shows the weakness of Euclidean distance.



**Figure 2.2:** Euclidean distance between document vectors

### 2.4.2 Cosine Similarity

Cosine similarity is used extensively in text mining to compute the similarity between text documents where the documents are represented as a weight vector. The weight vector, in this case, is TFIDF. Given two vectors  $t_1$  and  $t_2$ , their similarity is computed by the following equation [11].

$$\text{similarity}(t_1, t_2) = \cos(\theta) = \frac{t_1 \cdot t_2}{|t_1| \times |t_2|} \quad (2.11)$$

where  $\theta$  is angle between vectors  $t_1$  and  $t_2$

### 2.4.3 Word Mover's Distance

The main drawback of conventional distance measures is that they cannot capture word level similarities. If two documents express the same meaning without any common word, the similarity between these documents will be zero. These affects the similarity of short text documents. Because it is more likely that documents which contain a sentence do not have common words even if they talk about the same issue. Another problem with traditional distance measures is, even documents contain same words, there may be other similar words which is not taken into account while computing the similarity. Therefore these distance measures are poor.

A metric called Word Mover's Distance (WMD) overcomes the problems related to traditional distance metrics. It uses word embedding to understand the similarity or the distance between words. The distance between two words  $w$  and  $w'$  is computed by Equation 2.12.

$$c(w, w') = \|\mathcal{V}(w) - \mathcal{V}(w')\| \quad (2.12)$$

where the function  $\mathcal{V}$  takes a word and returns the corresponding vector from given word embeddings matrix. This distance is called word travel cost which is a building block for document distances [14]. Let  $T$  be a flow matrix where  $T_{ij}$  denotes how

much word  $i$  in  $d$  flows to word  $j$  in  $d'$ . Computing the WMD is equivalent to the following optimisation problem:

$$\begin{aligned} \min_{T \geq 0} \quad & \sum_{i,j=1}^n T_{ij} c(i, j) \\ \text{s.t.} \quad & \sum_{j=1}^n T_{ij} = d_i, \forall i \in \{1 \dots, n\} \\ & \sum_{i=1}^n T_{ij} = d'_j, \forall j \in \{1 \dots, n\} \end{aligned} \quad (2.13)$$

where  $d_k = \frac{tf(k,d)}{\sum_{i=1}^n tf(i,d)}$  and  $tf(k, d)$  is term frequency of word  $k$  in document  $d$ .

## 2.5 Pre-processing

Pre-processing is a crucial step in text mining. It is applied to the text data before vectorization. In this section, we discuss the pre-processing techniques such as stop-word removal, stemming, lemmatisation and pruning. All these pre-processing methods aim to remove noise or meaningless data from a corpus, and they all decrease the term space [29].

### 2.5.1 Stop-word Removal

Stop-words are often removed from documents in the pre-processing phase. These are words which have no lexical meaning and occur in documents very frequently. The common words in documents are articles, pronouns, prepositions which do not add any meaning to the sentences [22]. Usually, the previously defined stop list is used to eliminate stop words from documents. However, in addition to pre-defined stop-list corpus specific stop-words can be constructed based on word occurrence statistics. This is done by adding words to stop-list which has document frequency value more than a specified threshold [19]. This process is called automatic stop-word generation.

### 2.5.2 Stemming

In documents, same words which have different suffixes are taken as entirely different words. To avoid this stemming can be used. Stemming is eliminating the suffix of the words, so that inflected words are reduced to the same term. For example, if stemming is applied to the words "drives", "driving" and "driver", all of them will be transformed to "drive". No dictionary lookups are needed in stemming. Therefore, sometimes inflected words may end up with a meaningless term. For instance, some stemming algorithms reduce the word "navy" to "navi" which has no meaning<sup>2</sup>.

---

<sup>2</sup>[www.nltk.org](http://www.nltk.org)

### 2.5.3 Lemmatisation

Lemmatisation is reducing inflected words to their base using dictionary lookups. In this case, the keys in the dictionary are inflected words and the value is the base of that word. Therefore, this technique is more powerful than stemming. Unlike stemming, the resulting word always has a meaning. However, applying lemmatisation may be very costly for large text corpus because one dictionary look-up needed for each word in the corpus. Therefore, the number of lookups is  $N \times K$  where  $N$  is the number of words in the corpus, and  $K$  is the length of the dictionary. Hash-maps can be used to speed-up this since one look-up in hashmap is close to  $O(1)$ .

### 2.5.4 Pruning Rare Terms

Pruning is eliminating rare words if their document frequency is less than some pre-defined threshold. The idea behind pruning is rare terms do not represent the clusters. However, they may add noise to the similarity measures [10]. Besides noise removal, pruning rare words decreases the dimensionality of term space more than other pre-processing techniques. The reason is a corpus usually contain many rare words.

# 3

## Methods

In the previous chapter, we have seen vector representation of words and documents, pre-processing techniques, and similarity measures. However, we still lack the data organisation or grouping. In this chapter, we present clustering and topic modelling algorithms. Furthermore, we present our method that improves the performance of existing algorithms. We also include a dimensionality reduction technique and evaluation measures for clustering and topic modeling.

### 3.1 K-means

K-means is the most popular clustering algorithm. It is very efficient and easy to implement. The k-means algorithm belongs to a category of prototype-based clustering. Prototype-based clustering means that each cluster has a representative called centroid [25].

The algorithm contains the following steps. First, the cluster centroids are initialized randomly and data points are assigned to the closest centroid. Next, The mean of current clusters is computed and the chosen centroids are moved to the mean point. This procedure is repeated until the centroids do not change. This state is called convergence. Algorithm 1 describes this process.

---

**Algorithm 1:** K-means algorithm

---

Randomly initialize  $k$  centroids;

**repeat**

**foreach** *data point*  $i$  **do**

        | assign  $i$  to the closest centroid ;

**end**

**foreach** *centroid*  $\mu$  **do**

        |  $\mu =$  the mean of data points assigned to  $\mu$ ;

**end**

**until** *Convergence*;

---

The k-means algorithm can be described as simple optimization problem where the best clustering corresponds to minimized intra-cluster sum of squared errors. This error is computed by Equation 3.1.

$$Error = \sum_{i=1}^n \sum_{j=1}^k w^{(i,j)} \|x_i - \mu_j\|^2 \quad (3.1)$$

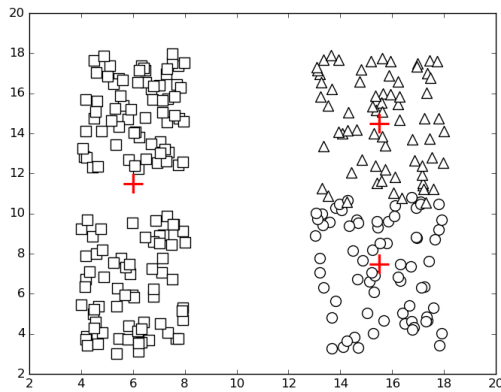
where  $n$  is the number of data points,  $k$  is the number of clusters,  $\mu_j$  is the centroid for the cluster  $j$ , and  $w^{(i,j)} = 1$  if  $x_i \in j$ ,  $w^{(i,j)} = 0$  otherwise [25].

### 3.1.1 K-means++

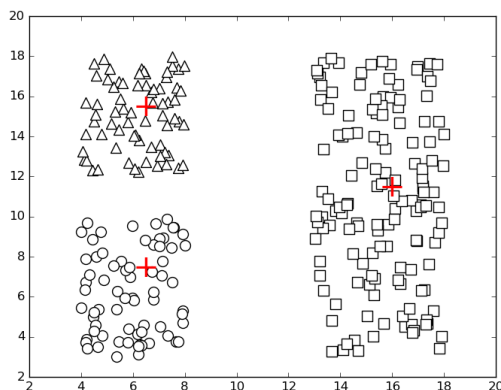
One drawback of k-means algorithm is that its initialisation step is done uniformly at random. This sometimes may lead to bad initialisation and thereby k-means may result in wrong local optimum as shown in Figure 3.1a. K-means++ addresses this issue. In K-means++ the centroids are chosen one by one. After selecting a centroid the next one is selected with the following probability:

$$\mathcal{P} = \frac{\mathcal{D}(x)^2}{\sum_{x \in \mathcal{X}} \mathcal{D}(x)^2} \quad (3.2)$$

here  $\mathcal{D}(x)$  is the shortest distance from a data point to already chosen centroid and  $\mathcal{X}$  is the set of data points. Therefore the data points which are close to the already selected centroid is going to have a low probability of being selected as a next centroid than data points which are far [2].



(a) K-means at wrong local optimum



(b) K-means at correct local optimum

**Figure 3.1:** K-means at local optimum. The different shapes (circle, square, triangle) indicate different clusters and the centroids are denoted by '+'.

**Algorithm 2:** K-means++ algorithm

---

Take one center  $c_1$ , chosen uniformly at random from the dataset;  
 Take new center  $c_i$ , choosing  $x \in \mathcal{X}$  with probability  $\mathcal{P}$ ;  
 Repeat the previous step until we have  $k$  centers;  
 Proceed as the standard k-means algorithm.

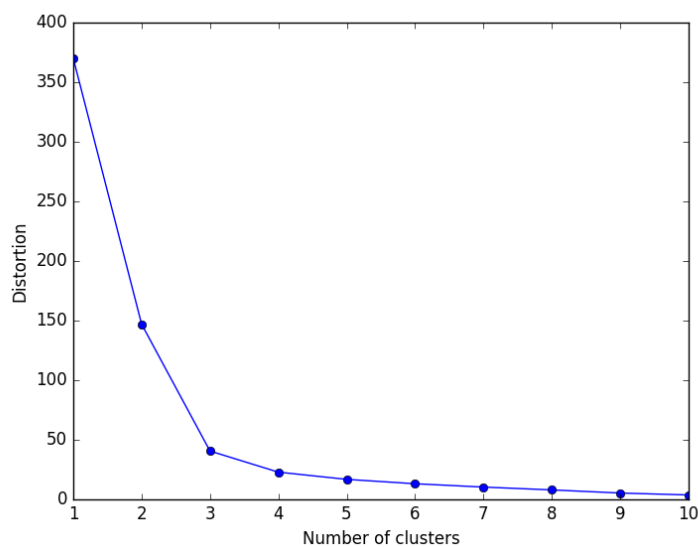
---

According to [2] k-means++ outperforms k-means algorithm both in time and accuracy.

### 3.1.2 Choosing the Number of Clusters

Finding the number of clusters in a dataset is the main challenge in clustering. To identify a value which is very close to an exact number of clusters is an open question until now. However, to get a reasonable number of clusters internal clustering validation techniques such as Calinski-Harabasz index and Silhouette index are used [13]. This validation measures are based on within and between cluster distances. The most famous method is elbow which uses within cluster sum of squared error (distortion) to identify the number of clusters [25].

Figure 3.2 shows a curve which indicates how k-means error function changes when we increase the number of clusters. By looking at Figure 3.2 we can say that 3 is good choice for number of clusters. According to the figure, the error function is decreasing slightly (for clusters  $> 3$ ) when number of clusters is increased. That is because, if we assign each data point to a cluster the error will be 0. However, sometimes this method ends up with an ambiguous curve where it is hard to say what is good number of clusters.



**Figure 3.2:** Elbow Curve



## 3.2 Probabilistic Topic Models

Probabilistic Topic Models are widely used in information retrieval and Natural Language Processing. They can infer the hidden topic structure in a large text corpus. Topic modeling algorithms does not need any prior labelling and the topics emerge from analysis of the entire documents [5]. Unlike, clustering one document can belong to many topics with different probabilities.

### 3.2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a type of topic model which assumes multiple topics for a document. Topic is a distribution over a fixed vocabulary [5]. In each topic the distribution of words are different. First, lets assume the topics are specified before documents are generated. The documents are generated by the following process. Initially random distribution of topics ( $K$ ) are selected. For each word in the document a random topic  $T$  is selected from  $K$ . Finally, a word is selected from the topic  $T$ . This generative process is described by Algorithm 3 and Equation 3.3.

$$p(\beta_{1:k}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^K p(\beta_i) \prod_{d=1}^D p(\theta_d) \left( \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right) \quad (3.3)$$

where,  $\beta_{1:K}$  are the topics (each  $\beta_k$  is a distribution over the vocabulary),  $\theta_d$  is the topic proportions of document  $d$  ( $\theta_{d,k}$  is topic proportion of topic  $k$  in document  $d$ ),  $z_d$  is the topic assignment of  $d$ th document and  $z_{d,n}$  is topic assignment of  $n$ th word in the document  $d$ ,  $w_{d,n}$  is  $n$ th word in document  $d$ .

---

**Algorithm 3:** The generative process of each document

---

```

foreach Document in the corpus do
  Randomly choose a distribution over topics
  foreach word in the document do
    Randomly choose a topic from the topic distribution
    Randomly choose a word from the topic
  end
end

```

---

The goal of topic modeling is to discover the topics from a collection of documents automatically. The topics, per document topic distribution, per-word topic assignment are all latent variables which can be inferred from documents. To compute the hidden topic structure from documents the probability of the conditional distribution of the hidden variables given the documents must be computed [5].

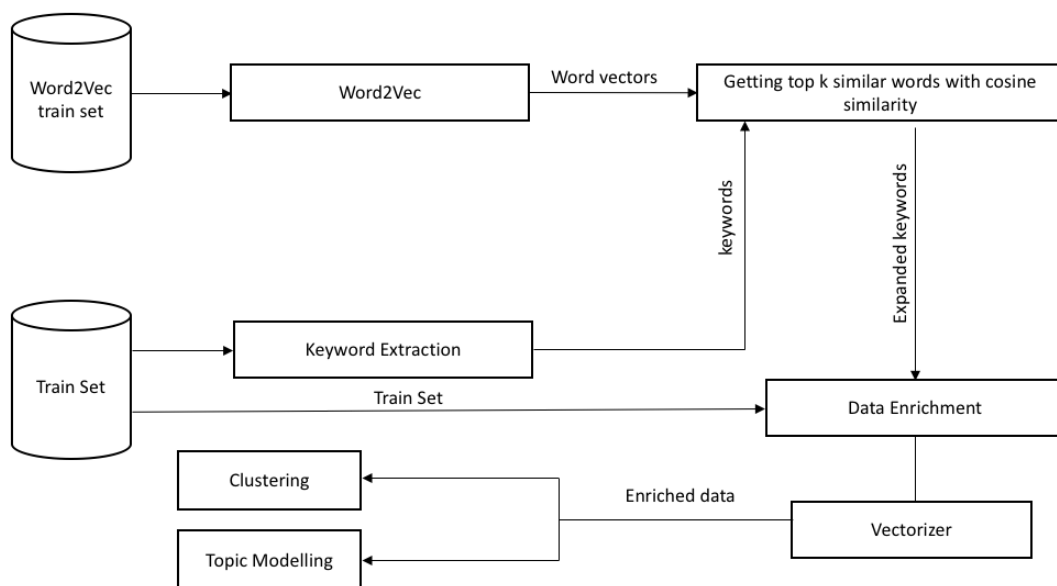
The computational problem of inferring the latent topics is equivalent to computing the following probability.

$$p(\beta_{1:k}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:k}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \quad (3.4)$$

In Formula 3.4 the numerator can be calculated straightforwardly using 3.3. However, computing the exact value of the denominator is a very difficult problem. It can be computed by summing the joint distribution over every possible instantiation of hidden topic structure which makes the problem intractable [5]. This probability can be approximated using Gibbs sampling [5].

### 3.3 Keyword Expansion Approach

It is not a good idea to apply clustering directly to TFIDF matrix especially when the documents are short. Therefore we need a more sophisticated pre-processing technique. We designed new approach which is based on keyword expansion to improve the clustering and topic modeling result. This method is a combination of keyword extraction, word embeddings and clustering or topic modeling algorithm. The overall procedure is as follows. First, the word embedding is trained on the Word2Vec dataset. We call this dataset  $D_w$ . Then we extract keywords from the train set using keyword extraction algorithm. After getting the keywords, we get top  $n$  similar words to each input keyword using cosine similarity on word vectors which are produced by word embeddings. Finally, the enrichment is done by appending the similar words to each keyword which produces a new enriched dataset. Diagram 3.3 shows the overall pipeline of this approach.



**Figure 3.3:** Data enrichment pipe-line

For a better illustration, let us look at how the enrichment function works. Let  $\mathcal{F}$  be the function which finds  $n$  similar words using word embeddings where  $n$  is specified as a parameter. Equation 3.5 shows the output of the function  $\mathcal{F}$  for a keyword  $k$ .

$$\mathcal{F}(k, n) = \{w_1, w_2, w_3, \dots, w_n\} \quad (3.5)$$

If two keywords,  $k_1$  and  $k_2$  are related then  $\mathcal{F}(k_1, n)$  and  $\mathcal{F}(k_2, n)$  will have big overlap. Therefore, the documents enriched by these keywords will have high similarity. The overall procedure is summarised by Algorithm 4.

---

**Algorithm 4:** Enrichment procedure of documents

---

**Input** :  $\mathcal{K}, \mathcal{D}, n$

**Output:**  $\mathcal{D}_e$

```

foreach  $d \in \mathcal{D}$  do
  foreach  $k \in \mathcal{K}$  do
    if  $k \in d$  then
       $d = d + \text{""} + \mathcal{F}(k, n)$ 
    end
  end
  add  $d$  to  $\mathcal{D}_e$ 
end

```

---

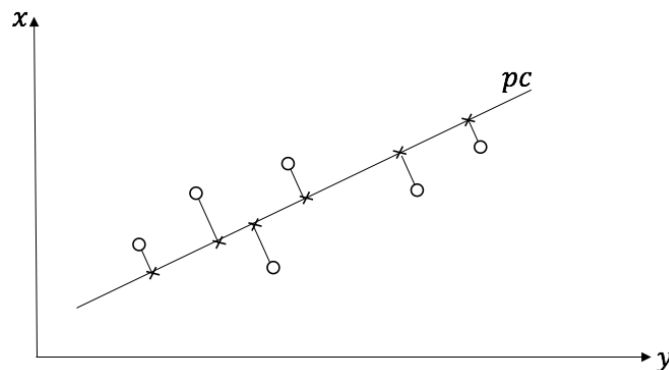
In Algorithm 4,  $\mathcal{K}$  is keyword list,  $\mathcal{D}$  is original document corpus and  $n$  is the parameter to select top  $n$  similar words for each keyword. The dataset  $D_w$  can exactly be the same as train set if train set contains enough data for word embeddings to learn good word vectors. However, it can also be other data with similar context. The accuracy of word embeddings increases as the dataset gets bigger. The idea of enrichment is inspired by the paper [31]. However, it is the first time the enrichment is done by word embeddings instead of a synonym dictionary. There are a couple of advantages in using word embeddings instead of a synonym dictionary. Synonym dictionary only contains a word which has the same meaning, not related words. For instance, if some animal is a keyword, word embedding can find other animals as related words which are not the case for synonym dictionary. In our case, this will help to find all animal-related accidents. Another advantage is that the data is expanded by the words which are occurred already in the text and hence it does not increase the dimensionality of feature space. Moreover, this method decreases the sparsity of vectors, and after enriching the documents, the data becomes more separable. Finally, it helps to reduce the distance between documents which describes the same situation in different ways.

### 3.4 Dimensionality Reduction

The dimensionality reduction is projecting high dimensional data to lower subspace and preserving the informativeness of the data. Many machine learning algorithms cannot deal with high dimensional data. Therefore dimensionality reduction is used to speed up the running time of these algorithms. Another application of dimensionality reduction is for data visualisation. The data is projected into 2D or 3D space; this helps to understand the data. If the data is well-separated, then one can even identify the number of clusters using this method.

Principal components analysis or PCA is by far the most widely used approach of dimensionality reduction. In [4] PCA is defined as the orthogonal projection of the data onto a lower dimensional subspace, such that the variance of the projected data

is maximised. This means that PCA is an optimisation problem of finding the line such that the variance of projected points is maximum (Figure 3.4).



**Figure 3.4:** Projecting 2D data onto one-dimensional space

The first step in PCA algorithm computes a covariance matrix for a given data. This matrix  $d \times d$  matrix where  $d$  is the number of features contains pairwise covariances for each feature in a dataset. The covariance between two features vectors is computed by Equation 3.6.

$$\sigma_{jk} = \frac{1}{n} \sum_i^n (x_j^i - \mu_j)(x_k^i - \mu_k) \quad (3.6)$$

where  $\mu_j$  and  $\mu_k$  are the means of feature  $j$  and  $k$ . To compute the covariance matrix Equation 3.6 has to be computed for each pairwise feature vectors. After computing the covariance matrix, it is decomposed into eigenvectors and eigenvalues. In order to get  $k$  principal components, we have to get  $k$  eigenvectors which correspond to top  $k$  eigenvalues.

## 3.5 Evaluation Metrics

Cluster evaluation measures the correctness of clustering result. There are two kinds of cluster evaluation metrics which are called external and internal validation. External validation measures the quality based on already labelled data called gold standard clusters. Internal validity evaluates the result on information intrinsic to the data alone. The latter is useful when there is no gold standard clusters or ground truth available [27].

### 3.5.1 F-Measure

F-Measure is mapping based measure where each cluster  $k_i \in K$  is mapped to  $c_j \in C$ .  $K$  is set of system-generated clusters, and  $C$  is a set of naturally grouped classes. Since F-Measure uses true labelling to evaluate clustering, it is an external validation metric.

$$F(C) = \sum_{c_i \in C} \frac{|c_i|}{S} \max_{k_j \in K} F(c_i, k_j) \quad (3.7)$$

$$Recall(c_i, k_j) = \frac{|c_i \cap k_j|}{|c_i|} \quad (3.8)$$

$$Precision(c_i, k_j) = \frac{|c_i \cap k_j|}{|k_j|} \quad (3.9)$$

$$F(c_i, k_j) = \frac{2 \times Recall(c_i, k_j) \times Precision(c_i, k_j)}{Recall(c_i, k_j) + Precision(c_i, k_j)} \quad (3.10)$$

where  $S$  is the number of objects,  $Precision$  is the number of correct cluster results divided by the number of all results,  $Recall$  is the number of correct cluster results divided by the number of naturally correct results [30].

### 3.5.2 Rand Index

The Rand index is external validation scheme which computes the similarity between system generated clusters and gold standard clusters and it is computed by the following equation.

$$RI = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} \quad (3.11)$$

where  $TP$  is a decision which assigns similar documents to the same cluster,  $TN$  is assignment of dissimilar documents to different cluster.  $FP$  is assignment of dissimilar documents to the same cluster and  $FN$  is assigning similar documents to distinct clusters [23].

### 3.5.3 Silhouette Index

Silhouette index (SI) is an intrinsic cluster evaluation method. It measures between cluster separation and within cluster similarity. SI for a single sample  $i$  is computed by Equation 3.12.

$$s(x_i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (3.12)$$

where  $a(i)$  is the average distance of data point  $i$  to all data points of the same cluster, and  $b(i)$  denotes the average distance of data point  $i$  to the data points of closest cluster [1]. Based on Formula 3.12, the silhouette scores can be computed for each individual cluster  $C_j$ . This is computed by Equation 3.13 [1].

$$s(C_j) = \frac{1}{|C_j|} \sum_{x_i \in C_j} s(x_i) \quad (3.13)$$

Evaluating clusters is not the only application of SI. It is also used as a visualisation tool for finding the number of clusters [25].

### 3.5.4 Calinski-Harabasz index

Calinski-Harabasz index is also an internal clustering validation technique. This clustering technique is based on between and within cluster variance. Unlike Silhouette index, this evaluation metric can have any positive value. It is computed

by Equation 3.14 [27].

$$CH = \frac{\text{trace}(S_B)}{\text{trace}(S_w)} \cdot \frac{n-1}{n-k} \quad (3.14)$$

where  $S_B$  is between cluster scatter matrix and  $S_w$  the within cluster scatter matrix,  $n$  is the number of data points and  $k$  is the number of clusters and the function  $\text{trace}(A)$  sums up the diagonal elements of matrix  $A$ .

### 3.5.5 Evaluation of Probabilistic Topic Models

Most widely used measure to evaluate the performance of probabilistic topic models is by computing perplexity. Perplexity is the probability of the test set, normalised by the number of words [4]. The perplexity is used in text mining, natural language processing and many other information retrieval areas. For a test set of  $M$  documents, the perplexity is defined in [6] as:

$$\text{perplexity}(D_{\text{test}}) = \exp \left\{ - \frac{\sum_{d=1}^M \log p(w_d)}{\sum_{d=1}^M N_d} \right\} \quad (3.15)$$

where  $N_d$  is the number of words in the  $d^{\text{th}}$  document of held-out test corpus  $D_{\text{test}}$  and  $w_d$  is  $d^{\text{th}}$  document in the corpus. Lower perplexity score indicates a better model.

Another widely used evaluation technique for machine learning models is cross-validation. The intuition behind cross-validation as stated in [20] is, to split the training data into  $K$  folds; then, for each fold  $k \in \{1, \dots, K\}$ , we train on all the folds but the  $k^{\text{th}}$ , and test on the  $k^{\text{th}}$ , in a round-robin fashion. Cross-validation can be applied with perplexity to estimate the error in a given model.

# 4

## Experiments and Results

This chapter presents the evaluation of clustering and Latent Dirichlet Allocation with different pre-processing methods. We also evaluate the methods for document ranking and provide information about the pre-processing parameters and how we deal with sparse document vectors.

### 4.1 The Datasets

In this project we analysed two text corpus, the accident descriptions provided by police and hospital. The detailed information of the text corpus is given in Section 2.1. The statistics of both datasets is depicted in Table 4.1.

**Table 4.1:** The number of documents and unique words in dataset  $P$  and  $H$

Datasets	#documents	#unique words
$P$	222375	71767
$H$	448476	81348

To test the quality of clusters we have constructed a test set containing 5485 documents from the dataset  $P$ . Then these 5485 documents are labelled into eight different clusters. This is done by carefully examining each document and searching words or combinations such as "red light". The documents which can be put into two or more clusters are deleted. Table 4.2 shows the cluster labels and number of elements in each cluster. We will refer to this dataset as  $T$ .

**Table 4.2:** Gold standard clusters from  $T$

Labels	Cluster descriptions	# documents
1	Collision with objects near the road (e.g. tree, fence)	855
2	Red light violation	665
3	Animal involved accidents	817
4	Accidents while turning	586
5	Chain accidents	321
6	Pedestrian involved accidents	748
7	Vehicle spinning	936
8	Accidents at intersections	557
	Total	5485

## 4.2 Pre-processing

This section contains information about how the pre-processing part is implemented and how it affects dimensionality of document vectors.

### 4.2.1 Stop-word Removal

Stop-word removal is implemented using nltk stop-words [3]. The removal is done by simply eliminating all words in documents which are in stop-list. Some words which do not add any meaning to crash scenarios are also added to stop-list manually.

The corpus specific stop-words are eliminated by removing frequent terms. Frequent terms affect the result badly although inverse document frequency penalises the frequent terms. The intuition behind this is that half of the dataset cannot be about the same situation. Therefore words occurring in more than half of the document should not be important. For both datasets, we eliminated words which have document frequency at least 25%. If we increase this value, the number of unique words does not change.

### 4.2.2 Lemmatisation

For lemmatisation purposes, we used Saldo [15] which is an extensive electronic lexicon resource for modern Swedish written language <sup>1</sup>. Saldo provides inflected Swedish words and their reference root. The implementation is done by reducing the inflected words to their root form. Stemming can also be used to reduce inflected words to their stem. However, we could not find a good stemming package for Swedish.

### 4.2.3 Pruning Rare Terms

Pruning is implemented by eliminating words which occurred less than some threshold  $k$ . We evaluated the k-means algorithm with different pruning values ranging from one to ten. We did not observe a major improvement by trying different pruning values. However, with pruning value five we observed a dramatic reduction of unique words on dataset  $P$  and  $H$ . The words which are removed by pruning mostly contain spelling errors or they describe specific street names such as "Johanssgatan", "Hästgatan", "Tromtögatan".

### 4.2.4 How Pre-processing Affects Dimensionality

The dimensionality plays a major role in time performance of algorithms. The pre-processing methods reduce term space and thereby the dimensionality. There are

---

<sup>1</sup><https://spraakbanken.gu.se/eng/resource/saldo>  
<http://www.lexiconista.com/datasets/lemmatization/>



71767 and 81348 unique words without pre-processing in dataset  $P$  and  $H$  respectively. The different pre-processing techniques affect this value differently (refer Table 4.3). After applying all pre-processing techniques described in previous sections, we can decrease this value considerably and speed up the convergence of the algorithms.

**Table 4.3:** Dimensionality reduction with different pre-processing techniques

pre-processing method	#Unique words ( $P$ )	#Unique words ( $H$ )
Stop-words	71733	81317
Lemmatisation	71721	76746
Pruning	14218	9746
Eliminating Frequent Words	71753	81345
After all pre-processing methods	12534	9746

The number of unique words is reduced to 12534 for dataset  $P$  and 9746 for  $H$  after applying stop-words removal, lemmatisation, pruning and frequent word elimination. As Table 4.3 indicates, the pruning affects dimensionality reduction more than any other pre-processing method. The dimensionality after applying all pre-processing methods is already reasonable regarding running time. Therefore, we will not apply PCA to reduce the dimensionality.

## 4.3 Algorithms

We have used k-means on TFIDF term weighting scheme with unigrams, bigrams and topic modeling. Furthermore, we have tried a combination of keyword extraction, word embeddings and clustering or topic modeling. For each of them, we get different performance.

### 4.3.1 Latent Dirichlet Allocation

The LDA algorithm uses initial hyper-parameters  $\alpha$  and  $\beta$  as described in Chapter 3. In addition to those hyperparameters, we need to specify the number of topics and iterations for the algorithm to converge. We have experimented by varying the value of  $\alpha$ ,  $\beta$  and the number of iterations. By setting the value of  $\alpha$ , to be 0.5,  $\beta$  to be 0.1, and the number of iteration of 1500 gave a good result regarding perplexity and manual analysis.

Before applying the LDA algorithm documents are pre-processed. The preprocessing phase includes changing to lowercase, stopword removal, punctuation and numbers. Also, the Swedish lemmatisation is used for all datasets.

We have used python LDA package <sup>2</sup>. This implementation of LDA uses Gibbs sampling, which is the most commonly used sampling algorithm for topic modeling [5].

---

<sup>2</sup><https://pypi.python.org/pypi/lda>

### 4.3.2 K-means

We have tried k-means with unigrams and n-grams.  $n$ -grams are co-occurred words within window size  $n$ . If  $n$  is taken one and two, the model is called unigram and bigram respectively. The bigrams are used to capture the phrases like "red light". We also used k-means on the data which is processed differently. The processing technique is discussed in Section 4.3.5. The selected parameters for k-means are given in Table 4.4.

**Table 4.4:** K-means parameters

parameter	value
initialization method	k-means++
max_iter	500
n_init	20

- *k-means++* - A smart initialisation method in order to speed up the convergence of the algorithm is discussed in 3.1.1.
- *max\_iter* - The number of iterations of k-means algorithm in a single run.
- *n\_init* - The number of initialisation of k-means with different centroids.

K-means is implemented using python scikit-learn package [7].

### 4.3.3 Word Embeddings

Given a set of documents, word embeddings can learn the word vectors for each word in the document set. From the vectors, similar words can be identified using cosine similarity. We used word embeddings to identify the similar words. For the dataset  $P$ , it can capture similarity between words. However, if the dataset is larger, the model can produce much better result. Therefore we decide to combine the datasets  $P$  and  $H$  and give combined documents as an input to word embedding algorithm. The word embedding is implemented using gensim<sup>3</sup> package [26]. For instance, for the words "korsning" ('intersection') and "seriekrock" ('pileup') the word embedding identified related words are very close to the given words (Table 4.5).

**Table 4.5:** Words that are identified as related to "korsning" ('intersection') and "seriekrock" ('pileup') by word embeddings

Word	related words according to word embeddings
korsning	väggkorsning, kors, korsn, cirkulationsplats, väjningsplikten
seriekrock	kökrock, seriekollision, köolycka, kedjereaktion, kedjekrock

<sup>3</sup><https://radimrehurek.com/gensim/models/word2vec.html>

**Table 4.6:** Parameters for word embedding

parameter	value
Number of neurons in the hidden layer	300
min-count	5
Windows size	6

Table 4.6 shows the selected parameters for word embedding. The number of neurons in the hidden layer is selected based on the original paper (Mikolov et.al [18]). The words will be ignored which occur less than *min-count*. This helps to decrease the computation time of word embeddings. Since the documents in the dataset are very short the window size 6 is enough and by some manual inspection, we can see that the result is reasonable in terms of word similarity.

### 4.3.4 Keyword Extraction

The keywords are essential in understanding the set of documents. We used keywords in the approach described in 4.3.5. The keywords are extracted based on document frequency values of terms. The words which have document frequency between some specified thresholds are taken as keywords. This range for the datasets  $P, H, T$  is specified in Table 4.7.

**Table 4.7:** Parameters keyword extraction

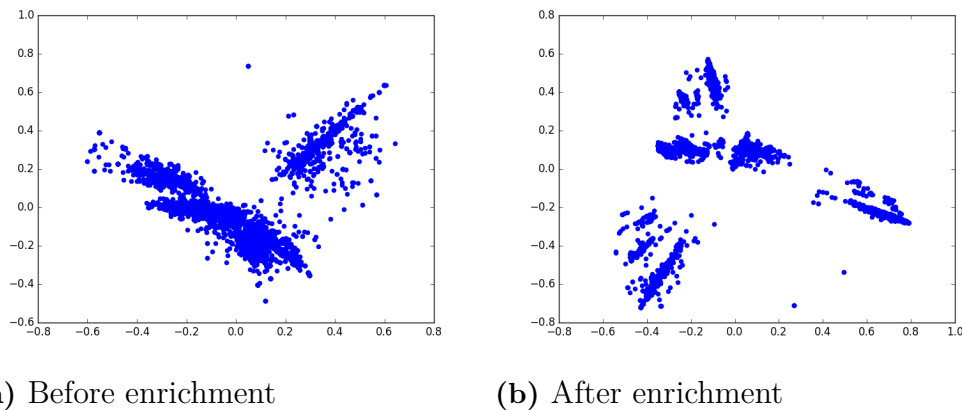
Dataset	min df	max df
T	300	1150
P	700	46698
H	700	94179

where *min df* and *max df* stands for minimum and maximum document frequency. This specified range for dataset  $T$  provides the following keywords "ljus" ("light"), "okänd" ("unknown"), "fotgängare" ("pedestrian"), "röd" ("red"), "hare" ("rabbit"), "korsning" ("intersection"), "rådjur" ("deer"), "seriekrock" ("pileup"), "bakifrån" ("rear"), "träd" ("tree"), "fåra" ("few"), "u-sväng" ("u-turn"), "rund" ("round"), "snurra" ("spin"), "sladda" ("slide"), "djur" ("animal") which contains all of the cluster labels. In fact if we increase the *max df* value the keyword list does not change. This is because the stop words are removed before keyword extraction and words which occur more than 21% of documents are not very important.

### 4.3.5 Enriching the Data

To enrich the data we first extracted the keywords as described in Section 4.3.4. Next, we trained Word2Vec on the combined dataset  $P$  and  $H$ . The parameters are taken exactly as stated in Section 4.3.3. The rest of the enrichment procedure is done as described in Section 3.3. Furthermore, we took top 5 related words to enrich the data.

Figure 4.1a shows visualisation of PCA reduced TFIDF vectors of non-enriched data and 4.1b is visualisation of enriched data. The separation in an enriched data is much more explicit; one can even identify the number of clusters from the figure.

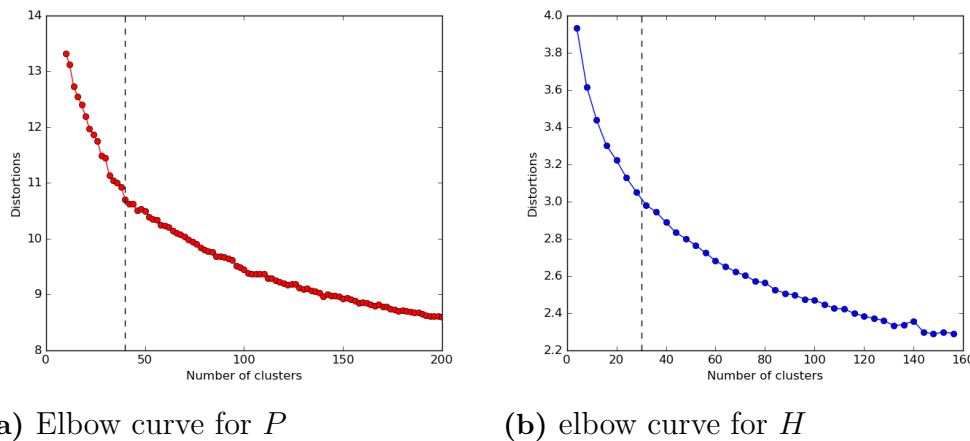


**Figure 4.1:** PCA reduced test data before and after enrichment

We will refer to enriched dataset for  $P$  and  $H$  as  $P_E$  and  $H_E$ .

## 4.4 Selecting the Number of Clusters and Topics

We used elbow method to identify the number of clusters in dataset  $P$  and  $H$ . Figure 4.2 shows the elbow curves for this dataset. Both curves are close to ambiguous, where one cannot easily find the number of clusters. However, for dataset  $P$  (4.2a) the decrease in distortion becomes more steady after 40 clusters. Therefore we can take 40 as a number of clusters for this dataset. For dataset  $H$ , we take a naive approach. Already for 30 clusters, there are some repetitions in groups, and therefore we have taken 30 clusters for this dataset.



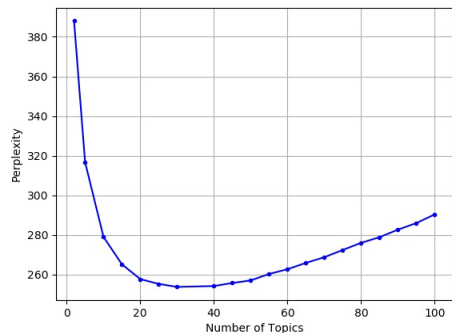
**Figure 4.2:** Elbow curves for dataset  $P$  and  $H$

## 4. Experiments and Results

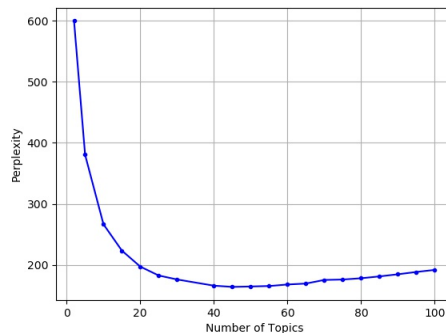
---

To select the number of topics for topic modeling we used perplexity with cross-validation. The tool used for this is the topic modeling package of R <sup>4</sup>.

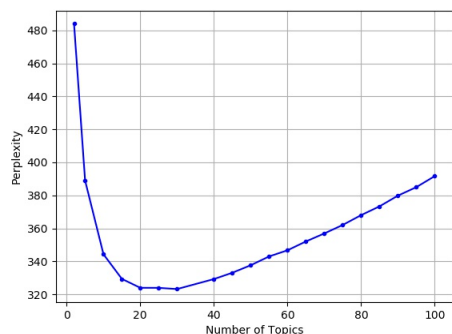
Figure 4.3 shows how perplexity changes when number of topics is increased. As it can be seen from Figure 30 is a reasonable number of topic for all four datasets.



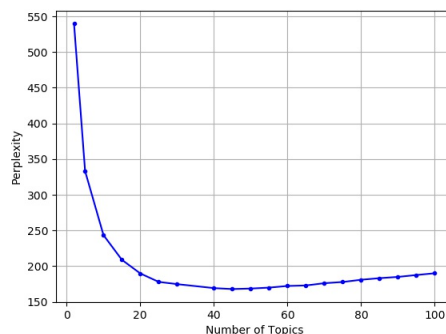
(a) Perplexity for  $P$



(b) Perplexity for  $P_E$



(c) Perplexity for  $H$



(d) Perplexity for  $H_E$

**Figure 4.3:** Perplexity values for choosing the number of topics

## 4.5 Evaluation

This section includes the evaluation of clustering, topic modeling and document ranking algorithms. Calinski-Harabasz, F-measure and ARI are used for the clustering evaluation. The evaluation for document ranking is done using F-measure. Furthermore, the evaluation of topic models is done by manual analysis and perplexity.

### 4.5.1 Clustering Evaluation

We have used external evaluation metrics such as F-measure, and Adjusted Rand Index for the test set. However, since  $P$  and  $H$  are unlabelled, it is not possible

---

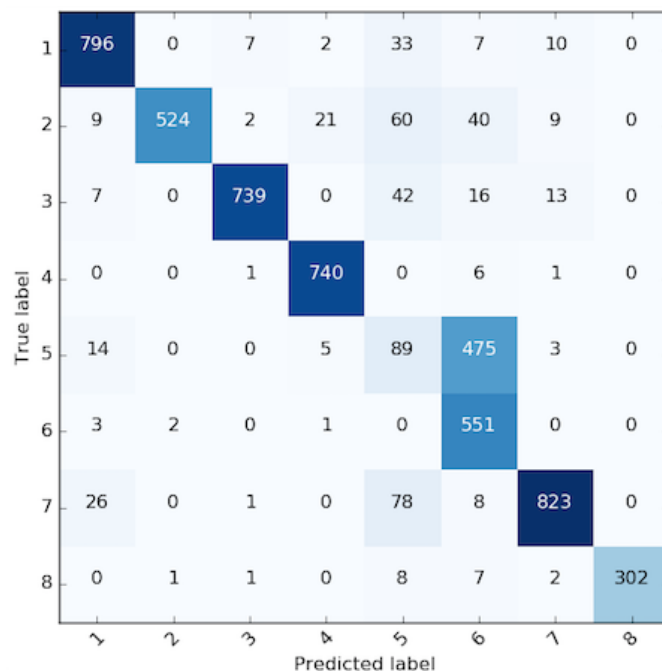
<sup>4</sup><https://cran.r-project.org/web/packages/lda/>

to evaluate the results on both datasets in an external manner. Therefore, we used Calinski-Harabasz index to check the quality of the clusters.

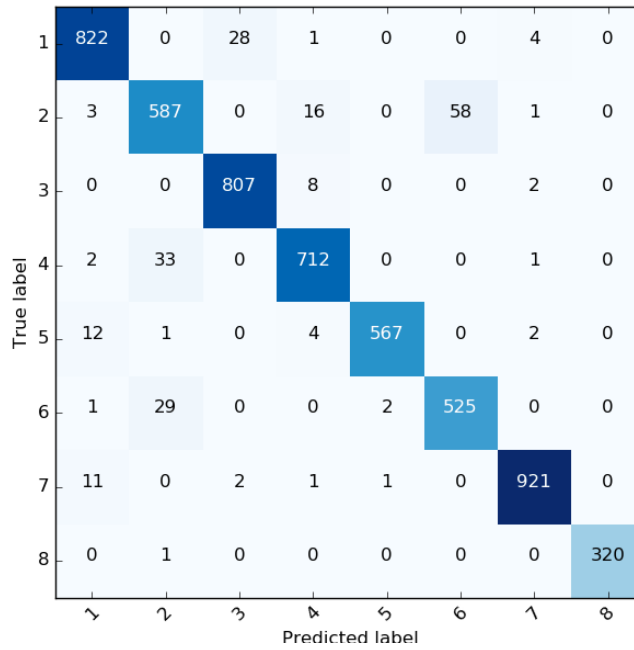
**Table 4.8:** Evaluation of different algorithms on test set  $T$

Algorithm	Precision	Recall	F-measure	ARI
Kmeans with unigrams	0.8075	0.7910	0.7930	0.7954
Kmeans with bigrams	0.8263	0.8217	0.8092	0.7622
Kmeans on enriched data	0.9594	0.9593	0.9592	0.9139
LDA	0.7017	0.7276	0.7060	0.7899
LDA on enriched data	0.7938	0.7723	0.7763	0.7080

Confusion matrices for the dataset  $T$  are shown in Figure 4.4 and 4.5 for the methods k-means with bigrams (4.4) and k-means on enriched data (4.5). The numbers 1-8 correspond to cluster descriptions in Table 4.2. In confusion matrices the darker the diagonal value the better the clustering result. However, in bigrams approach there are some non-diagonal element with dark colour which shows the number of documents included in the wrong cluster. For k-means on enriched data, there are no many false positives as indicated in 4.5.



**Figure 4.4:** Confusion matrix for k-means with bigrams on  $T$



**Figure 4.5:** Confusion matrix for k-means on enriched  $T$

Internal evaluation is performed on dataset  $P$  and  $H$ . Table 4.9 shows the Calinski-Harabasz values for different techniques.

**Table 4.9:** Evaluation with Calinski-Harabasz index on Dataset  $P$  and  $H$

Algorithm	$P$	$H$	$T$
Kmeans with unigrams	1490.7764	1400.1945	143.5178
Kmeans with bigrams	980.9094	1106.4575	110.264
Kmeans on enriched data	3445.5356	3569.7973	445.5300

As Table 4.9 indicates, for  $P$ ,  $H$ , and  $T$  k-means on enriched data performed the best and k-means on unigrams performed better than bigram approach.

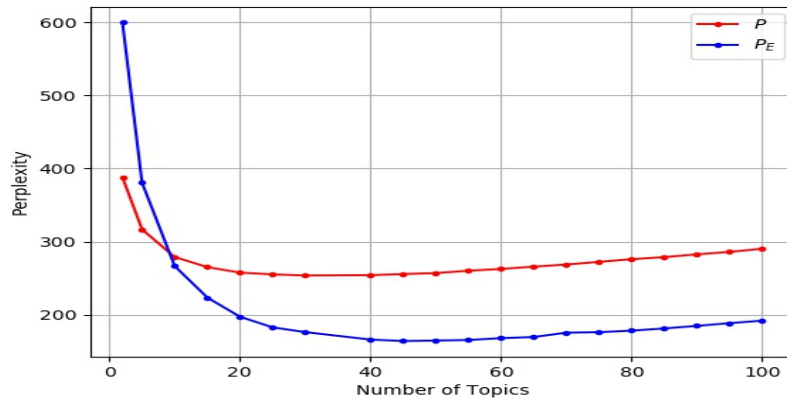
## 4.5.2 LDA Evaluation

The results from LDA algorithm are evaluated using perplexity and manual analysis of each topic. Manual analysis is done by examining each topic and the top words closely. The evaluation method will not be heavily dependent on perplexity. Perplexity is not selected as an ultimate evaluation method in this project for the following reasons. The primary factor is the lack of state-of-the-art evaluation method for a topic model. As mentioned in [5], evaluation methods for topic modeling is an active research area. Even though it is common to see perplexity used as an evaluation method for topic modeling, the perplexity is not a good indicator of how well the topic modeling algorithm fits the dataset [8]. Thus, we have supported the perplexity evaluation with manual analysis. Besides, by inspecting each resulting topics manually, we can tell if the purpose of the project is met.

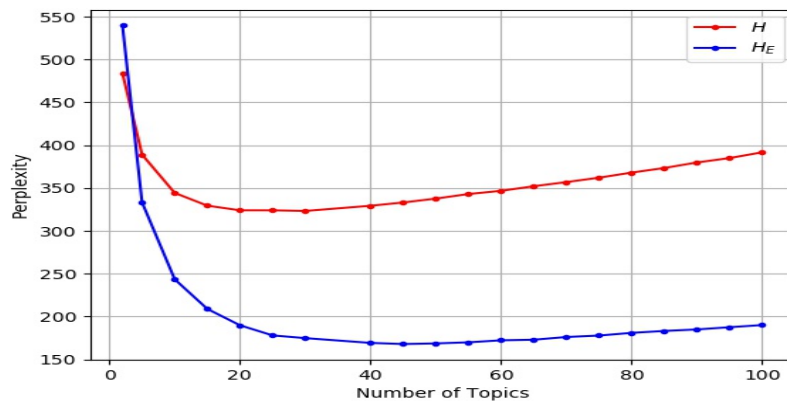
As Figure 4.7 and 4.6 indicates, the perplexity values for  $P_E$  and  $H_E$  are lower than

perplexity values for  $P$  and  $H$  respectively. Hence, based on the perplexity results the enriched datasets fit more to the LDA algorithm than the original dataset. The perplexity values are provided in Table C.1.

**Figure 4.6:** Perplexity results on  $H$  and  $H_E$



**Figure 4.7:** Perplexity results on  $P$  and  $P_E$



The topics for  $P$  and  $H$  contain a list of words where each word is associated with a different traffic accident. Thus, the result was not fine grained. However, after enriching datasets  $P$  and  $H$  the LDA algorithm produced good results. These topics can group similar words together and provide a clear description of the accidents. The following Table 4.10 contains few sample topics for  $P_E$ . All results are provided in Appendix A.

In Table 4.10, Topic 2 describes accidents related to traffic barriers where all the top words are related to each other. Another interesting result we observed by combining both LDA and word embeddings is that it can manage to group ill-typed words to



the same topic. This is indicated in Topic 14, which includes accidents happening due to unknown reasons and there are no much details to tell about the crash. The Swedish word "okänd" is wrongly spelt and the model was able to group misspelt words to the same topic.

**Table 4.10:** Sample topics for  $P_E$

Topics	Top ten words
2	vajerräcket vägräcke sidoräcket mittvajern vajerräcke räcke mit-träcke broräcket räcka mittbarriären skyddsäcket
8	trafikljus gul rödljuset stoppljus rödljus trafiksignal ljussignal signal ljus ljuda
14	oklar oförklarlig oförklarig synbar okönd obekant okönt oknd out-grundlig oländ okänd anledning
17	lb filbyte inbromsning lastbil körfältsbyte omkörning undanmanöver högersväng manöver omkörning
18	passera längas passerad svänga parallell ansluta permans martinssons passer elsa
20	kvinnu flicka pojke person dam fotgängare gångare gångtrafikanter övergångsställe obevakad
21	förare tappa kontroll föra mena försöka mede hona vägbana fuktig moddig isbelagd våt blöta hala spårig omkull äta isig isskorpa
22	förare öga stark solljus låg föra solsken morgonsol kvällssolen sol
23	hjort vildsvin rena rådjur katta hare älga grävling älgkalv älgko
26	cyklist trel cykla cykelbana cykel part buss bussa mopedist omkull

A pedestrian is described in the dataset as "woman", "man", "boy", "girl" or "person". Topic 20 of  $P$  tells about accidents involving pedestrians and the potential cause of the accidents. All the words describing a pedestrian are among the most probable words in Topic 20. Moreover, it also contains words such as "pedestrian crossing" ("övergångsställe") and "unattended" ("obevakad"). Therefore, we can infer that the accident involving pedestrians are associated with the person crossing a road unattended.

There is an interesting pattern in topics 21 and 22. These topics contain a clear reason for the accident. In Topic 21 the accident happened because the driver lost control. Topic 22 describes the poor vision problem where driver was not able to see because of sunlight.

The LDA on  $H_E$  gave interesting results which can be handy for researchers in the area of traffic safety. Sample topics for  $H_E$  is provided in Table 4.11. For instance, we can consider Topic 2, which describes road traffic accidents involving a bicycle. By observing the most probable words in this topic, one can tell that bicycle accident, and a bike lane with gravel have some connection. This hidden connection can then help researchers and road safety experts devise a solution.

Another advantage of using enriched dataset is to include all similar words describ-

ing the same accidents in the same topic; Topic 7 is a good example for this. Topic 7 includes all turning accidents such as left-turn, right-turn, U-turn. Similarly, Topic 13 which describes chain crashes. This crash type can be described using different words which are included in the topic. This advantage helps us solve the very fundamental problem we raised at the beginning of this project.

Also, LDA algorithm with word2vec provided to the discovery of hidden topics within the dataset. A good illustration for this is Topic 8 which encompasses documents which explain accidents that occurred while pedestrians are walking with a dog. It is necessary to differentiate those result with topic 19, which talks about accidents involving animals as well. In the later case, the animals are by themselves causing the accidents, and there is no pedestrian involved in the majority of the events.

All the relevant topics discovered by the LDA algorithm on original dataset  $P$  and  $H$  are also uncovered in the results of dataset  $P_E$  and  $H_E$ . For instance in  $P_E$ , all bicycle-related, motorbike-related accidents, traffic light accidents are grouped into Topic 26, Topic 17 and Topic 8 respectively.

Finally, it is meriting to state that not all the topics resulted from the LDA on the datasets are interpretable. While the majority of the Topics produced from dataset  $P_E$  and  $H_E$  describe the data well and are coherent, there are some diverse topics. For instance, in  $H_E$  Topic 4 and Topics 5 are a collection of different accidents. Similarly, Topic 18 of  $P_E$  is a mixture of various accidents descriptions. This problem gets worse in the case of dataset  $P$  and  $H$ .

**Table 4.11:** Sample topics for  $H_E$

Topics	Top ten words
2	cykla omkull cykel styre grusa framhjul flyga hjul grus cykelbana
4	viaduktväggen försökteåka framförvarande lastbilsida vilarinblandade snöskoteråkning bakifån stannaallt hand toga
5	jaga väga föra hem plötslig äta jobb fart hel hinna arbete
7	tappa nedförsbacke sväng högerkurva högersväng högerväng vänsterkurva kurva svängen nerförsbacke
8	koppel omkull hund häst hundvalp schäfer hunde hunder dragkärria
13	köbildning bilkö kö bakomvarande bak rödljus bakomliggande bakom okväll frordon bakifrån stillastående påkörda inblandad rusningstrafik kökrock seriekrock kedjekrock trafikolycka rad
18	passera längas passerad svänga parallell ansluta permans martinssons passer elsa korsa söder cyklist väga öster mede gata föra väster cykelbana
19	älga hjort djur rådjur vildsvin kanin föra räv vild grävling hare råddjur ruta framrutan vindruta framruta katta älgkalv bakruta krossa

### 4.5.3 Document Ranking Evaluation

In this section, we will evaluate the following text ranking techniques.

- **TFIDF based ranking:** In TFIDF based ranking the similarity between query and documents are computed by the dot product (cosine similarity) of TFIDF vectors of query and document.
- **TFIDF based ranking with query expansion:** The query and documents are enriched with keywords, and then we use TFIDF ranking as described before.
- **Ranking using Word Mover’s Distance (WMD):** In WMD the similarity is travel cost of words from query to a document, in this method documents are ranked based on this similarity.

In all ranking methods, the search result is sorted in ascending order of similarity values between documents and query. To evaluate the ranking algorithm we created gold standard query document pairs from the dataset  $T$ . We looked at the top 300 results of the search algorithm and calculated how many of them are in the document set corresponding to the query.

**Table 4.12:** Evaluation of different ranking algorithms on test set  $T$

Algorithm	Avg. precision	Avg. recall	Avg.F-measure
TFIDF	0.4382	0.4382	0.4382
TFIDF with query expansion	0.7972	0.7972	0.7972
Ranking via WMD	0.2867	0.2867	0.2867

In Table 4.12 the average precision and recall values are the same. This is because the size of retrieved documents is equal to the size of relevant documents. According to the test results, WMD based approach has lowest F-score value. This method cannot identify the important words as TFIDF. Therefore, it gives all words the same importance. However, this method performs better than TFIDF when the query is expressed with completely different words which the relevant documents does not contain. Table 4.13 contains the precision, recall and F-score values for few sample queries which does not have any occurrence in relevant documents. This better explains how the methods work on different kind of queries.

**Table 4.13:** Evaluation of different ranking algorithms with no occurrence of query words in relevant documents

Algorithm	Avg. precision	Avg. recall	Avg.F-measure
TFIDF	0.0	0.0	0.0
TFIDF with query expansion	0.9881	0.9881	0.9881
Ranking via WMD	0.2360	0.2360	0.2360

In TFIDF based ranking the similarity between documents become nonzero if and only if documents contain a common word. Since the query and relevant documents do not have any common word this ranking algorithm cannot capture any relevant documents and hence F-measure value becomes zero. The WMD algorithm may

produce better results regarding F-measure with document pairs. However, the experiments show that it does not work well on query document pairs probably because the query can be very short and even a single word. As Table 4.13 shows, the ranking via TFIDF with query expansion produced a result with high precision. The running time of document ranking algorithms is crucial. Because users cannot wait for query results too long. The running time of each ranking method for a single query is provided in Table 4.14.

**Table 4.14:** Running time of different ranking algorithms for a single query in MacBook Air with processor 2,2 GHz Intel Core i7 and Memory 8GB

Algorithm	Time (s)
TFIDF	0.1452
TFIDF with query expansion	0.5066
Ranking via WMD	12.3445

The TFIDF ranking performed better than the query expansion approach because it takes more times to enrich the query and documents than the document retrieval. Furthermore, the WMD for a single query on over 3000 documents took 12 seconds which is considerably expensive for query purposes. This method cannot be used without reducing search space on the whole dataset.

## 4.6 Discussion

The STRADA database is well structured and each accident case is labelled. However, the accidents which are assigned to a fixed label, e.g., “single accident” may actually link to many scenarios (e.g., "driver’s eyes-off-road", “too bright sun-light”). Such information can only be gathered from the text description.

For other cases, there may be no fixed label to describe a certain type of accidents. For example, there is no label called roadwork accidents in the database. Therefore, retrieving roadwork accidents was very hard according to [17] and the author used thirty different keywords for this purpose. The author has to think and find these thirty keywords manually.

Another example can be accidents happening due to an icy road. There is no such label in the database and this type of accidents can be under different labels such as single-vehicle accident, rear-end collision, etc. Therefore, the task of retrieving this kind of accidents becomes challenging; one has to search manually among the free-text description using relevant keywords (in this case, e.g., “icy road”, “ice on the roads”, “slippery road”). It is possible to make this process easier by employing techniques such as topic models, clustering and document ranking.

We have observed that clustering can find scenarios which are not explicitly defined as fixed labels in the database (e.g. "icy road", "multi-vehicle accidents", "weather condition") and therefore hard to retrieve with a single relational database search. After getting the groups, the analysis of specific crash scenario can be done by

studying the corresponding cluster. Clustering can potentially decrease the time and effort for doing this analysis. The crash reports, when clustered, can group accident description that may have been under different labels previously in the database. However, groups that we get from clustering may not represent the groupings we like.

Unlike clustering, the topic modeling algorithm is better for summarising the data and finding hidden topics. As discussed in Section 4.3.1, the topics identified by LDA summarise well the specific crash scenarios. It contains different terms which are related in context or meaning. Therefore, LDA topics highlight how similar crash types have been expressed in different terms. Even some of them contain the cause of accidents. The detailed analysis should be done by looking at most probable documents for these topics.

Document ranking method will provide search functionality for the dataset. As discussed in 4.5.3, by query expansion it was possible to get a reasonable number of accurate results. According to test results, this was the case even when query and relevant documents are expressed in different words. This method will make it easier to find specific accidents for which there is no fixed label describing them and it is difficult to think of all possible keywords that can be associated with these accidents. Ranking with query expansion overcomes this problem.

The keyword expansion method has the following limitations. Words which are extracted by the keyword extraction algorithm may not necessarily be keywords. Therefore, the method will give more weight to the unnecessary words and this will lead clustering algorithm to produce a bad result. Also, some words in the datasets can have a different meaning in different contexts. For instance, the word "woman" which can either be "driver" or "pedestrian" may be associated more to "pedestrian" as a result of word embeddings. Therefore, in the enrichment process, the word woman will be expanded to a combination of words which are all related to "pedestrian". This is the case for documents where "woman" is used as "driver" as well. In the latter case, the documents will be enriched by pedestrian related words which is not desirable.

The use of text enrichment to tackle the problem of sparsity in short text has been practised for some time. Many people have used external sources to make short text documents longer and therefore reducing the sparsity. Banerjee et al. [10] propose a method which uses Wikipedia as an external source to improve short text clustering. Hotho et al. [15] propose enrichment of short text documents using Wordnet (lexical database of English)[12]. A keyword expansion method instead of enriching every term is proposed by Wang et al. [7]. This method is similar to enrichment method which has been used in this project. The difference is word embeddings is used instead of synonym dictionary and keywords are extracted based on only document frequency values. The advantage of our method is no external source is needed if the datasets are large enough for word embeddings. Besides, our method is more efficient in reducing sparsity since it does not increase term space by adding new words.

# 5

## Conclusion

In this thesis, we explored how to cluster and summarise road traffic accident data which are collected by police and hospitals. First, we examined the pre-processing techniques such as pruning, stop-word removal and eliminating frequent words for these datasets. We have done some parameter tuning to identify the pre-processing parameters. It is observed that pruning helps to remove noise or unnecessary information from the data. Frequent term elimination removes non-expressive words. Moreover, we used a state-of-the-art technique to get the word embeddings. This is done by training a neural network on combined dataset  $P$  and  $H$ . We have seen that the trained word embedding model is capable of identifying similar words in the datasets. Besides, we explored the similarity measures to get the similarity between word and document pairs.

We used k-means algorithm with different pre-processing techniques. Firstly, k-means is used on unigrams and bigrams which produced similar results. However, for the dataset  $P$  and  $H$  it seems that unigram approach worked better than bigrams. One possible reason can be, the dimensionality of the term space is very high for bigram approach. Therefore, the TFIDF vectors end up to be sparse. Furthermore, we came up with a new pre-processing technique which is based on word embeddings. This technique is different in terms of enrichment via word embeddings and keyword extraction using only document frequency instead of TFIDF values of words. The advantages of using word embeddings over synonym dictionary were not increasing the dimensionality of term space and reducing the sparsity of document vectors. In addition, this pre-processing method helped to decrease the distance between documents with similar scenarios which has been expressed with different words. As our results indicate, after this processing the k-means performed well on the test set. With internal validation of clustering we have verified that after this pre-processing technique, the algorithm performed better than unigram and bigram approach.

We also used Latent Dirichlet Allocation to summarise the data. LDA algorithm is run on both enriched and non-enriched data. We have seen that after enriching the data, LDA topics became much more expressive which collect similar words together. By looking at these topics, it is possible to tell what kind of crash scenario the documents describe. However, this is not very explicit for the non-enriched dataset which includes some diverse words in the topics.

We examined three document ranking techniques which are simple TFIDF based

ranking, ranking via query expansion on enriched documents and Word Mover's Distance based method. Here query expansion approach outperformed the two other methods regarding accuracy. However, concerning running time TFIDF ranking retrieved the result faster.

Finally, we have tried clustering TFIDF weighted average of word vectors and distributed representation of document vectors [16]. By manual analysis, we have observed that clustering via these techniques does not produce good results probably because of the length of documents.

### 5.1 Future work

We have recognised that many words in the dataset are combined, or they are ill-typed. One can use tokenisation and spell correction algorithms to fix these issues. Since there is a possibility of relevant terms being ill-typed, it is expected that there should be some improvements in clusters, topics and document ranking results. Separating combined words will help to increase the similarity of documents which contain the same combined word.

Performing document ranking on the whole dataset may be very slow, and the number of documents is increasing each year. Therefore, to get fast query-search results, one can use search space reduction techniques. A simple approach can be, performing document ranking on a cluster that is very close to the query. Another approach can be retrieving Approximate Nearest Neighbours [9] of a query.

Short text analysis is an emerging research area. There are lots of new advancements such as Word2Vec word embeddings and Global Word Vectors (GloVe) [21]. One can also explore these techniques to get document vectors which can be used in clustering.

# Bibliography

- [1] Amparo Albalade and David Suendermann. “A combination approach to cluster validation using statistical quantiles”. In: *International Joint Conference on Computational Biological Systems (IJCBS)*. Shanghai, China, Aug. 2009.
- [2] David Arthur and Sergei Vassilvitskii. “K-means++: The Advantages of Careful Seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5. URL: <http://dl.acm.org/citation.cfm?id=1283383.1283494>.
- [3] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 1st edition. O’Reilly Media, Inc., 2009. ISBN: 9780596516499.
- [4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.
- [5] David M. Blei. “Probabilistic Topic Models”. In: *Commun. ACM* 55.4 (Apr. 2012), pp. 77–84. ISSN: 0001-0782. DOI: 10.1145/2133806.2133826. URL: <http://doi.acm.org/10.1145/2133806.2133826>.
- [6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 993–1022. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=944919.944937>.
- [7] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. “API design for machine learning software: experiences from the scikit-learn project”. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.
- [8] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. “Reading Tea Leaves: How Humans Interpret Topic Models”. In: *Neural Information Processing Systems*. Vancouver, BC, 2009.
- [9] Cong Fu and Deng Cai. “EFANNA : An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph”. In: *CoRR* abs/1609.07228 (2016). Accessed on: 25 May 2017. URL: <http://arxiv.org/abs/1609.07228>.
- [10] Andreas Hotho, Steffen Staab, and Gerd Stumme. “Wordnet improves Text Document Clustering”. In: *In Proc. of the SIGIR 2003 Semantic Web Workshop*. 2003, pp. 541–544.
- [11] Anna Huang. “Similarity Measures for Text Document Clustering”. In: *proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC)*. 2008.



- [12] Karen Spärck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of Documentation* 28 (1972), pp. 11–21.
- [13] Krzysztof Kryszczuk and Paul Hurley. “Estimation of the Number of Clusters Using Multiple Clustering Validity Indices”. In: *Multiple Classifier Systems: 9th International Workshop, MCS 2010, Cairo, Egypt, April 7-9, 2010. Proceedings*. Ed. by Neamat El Gayar, Josef Kittler, and Fabio Roli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 114–123. ISBN: 978-3-642-12127-2. DOI: 10.1007/978-3-642-12127-2\_12. URL: [http://dx.doi.org/10.1007/978-3-642-12127-2\\_12](http://dx.doi.org/10.1007/978-3-642-12127-2_12).
- [14] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. “From Word Embeddings to Document Distances”. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML’15. Lille, France: JMLR.org, 2015*, pp. 957–966. URL: <http://dl.acm.org/citation.cfm?id=3045118.3045221>.
- [15] Markus Forsberg Lars Borin and Lennart Lönngren. “SALDO 1.0 (Svenskt associationslexikon version 2)”. In: Språkbanken, Göteborgs universitet, 2008.
- [16] Quoc V. Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *CoRR* abs/1405.4053 (2014). Accessed on: 12 May 2017. URL: <http://arxiv.org/abs/1405.4053>.
- [17] Eva Lijegren. “Traffic accident in connection to road work in Sweden”. In: *Transport Research Arena* 1 (2014), pp. 1–10.
- [18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 3111–3119. URL: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.
- [19] Daša Munková, Michal Munk, and Martin Vozár. “Data Pre-processing Evaluation for Text Mining: Transaction/Sequence Model”. In: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, pp. 1198–1207. ISSN: 1877-0509. DOI: <http://dx.doi.org/10.1016/j.procs.2013.05.286>. URL: <http://www.sciencedirect.com/science/article/pii/S1877050913004298>.
- [20] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 9780262018029.
- [21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: <http://www.aclweb.org/anthology/D14-1162>.
- [22] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011. ISBN: 9781107015357.
- [23] W.M. Rand. “Objective criteria for the evaluation of clustering methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850.
- [24] Aniket Rangrej, Sayali Kulkarni, and Ashish V. Tendulkar. “Comparative Study of Clustering Techniques for Short Text Documents”. In: *Proceedings*

- of the 20th International Conference Companion on World Wide Web. WWW '11. ACM, 2011, pp. 111–112. ISBN: 978-1-4503-0637-9. DOI: 10.1145/1963192.1963249. URL: <http://doi.acm.org/10.1145/1963192.1963249>.
- [25] Sebastian Raschka. *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2015. ISBN: 1783555130.
- [26] 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.
- [27] Eréndira Rendón, Itzel M. Abundez, Citlalih Gutierrez, Sergio Díaz Zagal, Alejandra Arizmendi, Elvia M. Quiroz, and H. Elsa Arzate. “A Comparison of Internal and External Cluster Validation Indexes”. In: *Proceedings of the 2011 American Conference on Applied Mathematics and the 5th WSEAS International Conference on Computer Engineering and Applications*. AMERICAN-MATH'11/CEA'11. World Scientific, Engineering Academy, and Society (WSEAS), 2011, pp. 158–163. ISBN: 978-960-474-270-7. URL: <http://dl.acm.org/citation.cfm?id=1959666.1959695>.
- [28] Xin Rong. “word2vec Parameter Learning Explained”. In: *CoRR* abs/1411.2738 (2014). Accessed on: 05 April 2017. URL: <http://arxiv.org/abs/1411.2738>.
- [29] Julian Sedding and Dimitar Kazakov. “WordNet-based Text Document Clustering”. In: *Proceedings of the 3rd Workshop on RObust Methods in Analysis of Natural Language Data*. ROMAND '04. Geneva: Association for Computational Linguistics, 2004, pp. 104–113. URL: <http://dl.acm.org/citation.cfm?id=1621445.1621458>.
- [30] Prajool Shrestha, Christine Jacquin, and Béatrice Daille. “Clustering Short Text and Its Evaluation”. In: *Computational Linguistics and Intelligent Text Processing: 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part II*. Ed. by Alexander Gelbukh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 169–180. ISBN: 978-3-642-28601-8. DOI: 10.1007/978-3-642-28601-8\_15. URL: [http://dx.doi.org/10.1007/978-3-642-28601-8\\_15](http://dx.doi.org/10.1007/978-3-642-28601-8_15).
- [31] Jun Wang, Yiming Zhou, Lin Li, Biyun Hu, and Xia Hu. “Improving Short Text Clustering Performance with Keyword Expansion”. In: *ISNN*. 2009.



# A

## LDA topics

**Table A.1:** LDA topics for  $H$ . Top 10 most probable words are provided for each topic

Topics	Top 10 words
Topic 1	halka, isfläck, is, snöa, sanda, trottoar, isig, snö, pga, gångväg
Topic 2	cykel, styre, flyga, cykla, framhjul, landa, hoppa, hjul, fastna, tvärstopp
Topic 3	slå, ramla, huvud, ansikte, axel, asfalt, bakhuvud, falla, knä, mark
Topic 4	halka, toga, hand, emot, ramla, isfläck, ta, falla, arm, vä
Topic 5	omkull, cykla, falla, cykel, nedförsbacke, trottoarkant, trilla, ramla, cykelolycka, tillstånd
Topic 6	moped, körd, välta, mc, omkull, glida, motorcykel, kurva, åka, svänga
Topic 7	körd, lastbil, bil, traktor, sida, mitträcke, börja, km, släpa, motorväg
Topic 8	väg, körd, dike, väja, sladda, rådjur, träda, åka, volta, kurva
Topic 9	bil, stanna, körd, ljus, övergångsställe, korsning, röd, grön, släppa, se
Topic 10	sladda, dike, volta, rund, snurra, hamna, åka, väg, bil, tak
Topic 11	påkörd, bakifrån, bil, stillastående, köbildning, stanna, framförvarande, påkörda, seriekrock, rödljus
Topic 12	bil, svänga, pata, köra, plötslig, fila, svängd, se, vä, blinka
Topic 13	buss, falla, bussa, ramla, springa, gata, pata, kliva, stiga, trappa
Topic 14	oklar, olycka, hända, väg, anledning, okänd, veta, minnas, pata, troligen
Topic 15	bil, påkörd, ligga, hastighet, sida, korsa, övergångsställe, gata, bilist, fart
Topic 16	se, äta, hålla, titta, fart, smälla, köra, sena, försöka, sida
Topic 17	bil, passagerare, sitta, förare, km, patient, baka, körd, hastighet, sida
Topic 18	körd, krocka, väg, km, älga, stolpe, träda, sol, bil, blända
Topic 19	patient, vä, sid, hö, landa, knä, skada, slå, axel, ond
Topic 20	bil, korsning, körd, svänga, sida, se, svängd, köra, sid, korsa
Topic 21	snubbla, falla, trottoar, trottoarkant, gata, kant, fotgängare, ojämn, handlöst, snava

## A. LDA topics

Topic 22	tappa, hal, pga, kontroll, balans, väg, glida, väg bana, väglag, dålig
Topic 23	bil, bromsa, framförvarande, hinna, körd, kraftig, bromsare, inbromsning, stanna, plötslig
Topic 24	stå, bakifrån, påkörd, stilla, rondell, bil, påkörda, vänta, köra, rödljus
Topic 25	fot, trampa, hund, sned, falla, ramla, fastna, trottoarkant, vika, ben
Topic 26	km, körd, hastighet, cirka, bil, timme, väg, volta, enl, journal
Topic 27	bil, möta, sid, krocka, fela, kollidera, körbana, frontalkrock, väja, förare
Topic 28	cykla, cykel, cyklist, cykelbana, ramla, falla, kompis, trilla, välta, cykelvägen
Topic 29	åka, ramla, grusa, backa, lösa, backe, fart, stena, nedför, trilla
Topic 30	väga, hem, halka, jobb, parkering, arbete, skola, påväg, hus, hämta

**Table A.2:** LDA topics for  $H_E$ . Top 10 most probable words are provided for each topic

Topics	Top 10 words
Topic 1	uppfatta, upptäcka, uppmärksammad, observera, såg, notera, märka, se, uppmärksam, uppmärksammade
Topic 2	trilla, ramlare, ramla, vurpa, välta, falla, föll, stupa, ramalde, störta
Topic 3	cykla, omkull, cykel, styre, grusa, framhjul, flyga, hjul, grus, cykelbana
Topic 4	passera, passerad, blockera, enkelrikta, tillparkeringen, horndals, vägalternativ, garagplanen, komliggande, korsa
Topic 5	viaduktsväggen, försökteåka, framförvarande, lastbilsida, vilarinblandade, snöskoteråkning, mede, bakifån, stannaallt, och
Topic 6	jaga, väga, föra, hem, plötslig, äta, jobb, fart, hel, hinna
Topic 7	huvudled, utfart, stopplikt, korsn, vägskorsning, kors, fyrvägskorsning, vägskorsning, korsning, stoppskylt
Topic 8	tappa, nedförsbacke, sväng, högerkurva, högersväng, högerväng, vänsterkurva, kurva, svängen, nerförsbacke
Topic 9	koppel, omkull, hund, häst, hundvalp, schäfer, hunde, hunder, dragkärra, egon
Topic 10	övergångsställe, övergångstället, cyklist, person, järnvägsövergång, överfart, cykelöverfarten, cykelöverfart, övergångsställe, övergångsstället
Topic 11	km, ca, hastighet, volta, cirka, passagerare, timme, framförvarande, kört, stillastående
Topic 12	stoppskylt, vägskorsning, rödljus, avfart, korsning, påfart, rondell, bilkö, cirkulationsplats, prakeringen
Topic 13	cykel, cykla, jaga, fastna, fot, falla, hoppa, slå, sadel, pedal
Topic 14	köbildning, bilkö, kö, bakomvarande, bak, rödljus, bakomliggande, bakom, okväll, frondon

Topic 15	filbyte, undanmanöver, högersväng, densamma, vänstersväng, usväng, omkörning, omkörnig, räcka, vänstersvä
Topic 16	snubbla, falla, buss, trottoarkant, trampa, sned, gata, fot, kliva, bussa
Topic 17	spinna, vattenplaning, panik, punktering, motorstopp, sladd, sladda, retursladd, släpp, sladdoch
Topic 18	osandad, halkde, hlkade, halkolycka, glatta, halakde, slinta, ren, kullig, halka
Topic 19	blända, ögonen, solljus, skina, blendad, bländasdes, cykla, motljus, sola, halvljus
Topic 20	älga, hjort, djur, rådjur, vildsvin, kanin, föra, räv, vild, grävling
Topic 21	möta, fela, kollision, frontalkrockat, fronta, frontalkolliderat, frontalkolliderade, frontalkollision, frontalkrock, frontalkrockade
Topic 22	viadukt väggen, och, stannaallt, bakifån, mede, snöskoteråkning, vilarinblandade, lastbilsida, framförvarande, försökteåka
Topic 23	hala, blöta, halkig, glashal, vått, moddigt, hal, hallt, halrt, nalt
Topic 24	tvärnita, bromsa, nita, tvärbromsa, panikbromsa, bromsare, stanna, inbromsning, veja, väja
Topic 25	slå, patient, oklar, huvud, omkull, cykla, axel, vä, knä, hö
Topic 26	blankis, isgata, glansis, isfläckar, isbana, is, isfläck, isklump, iskant, isvall
Topic 27	svänga, svängd, blinka, gira, svänger, svände, svängde, vänstersväng, svägna, vänga
Topic 28	rödljuset, trafikljus, rödljus, gul, rörr, grönt, rött, röd, trafiksignal, stoppljus
Topic 29	framåt, falla, framstupa, handlost, frammåt, baklänges, framlänges, bakåt, slå, rygg
Topic 30	promenera, morgonpromenad, kvällspromenad, promenad, falla, ploga, joggingtur, skotta, sandning, salta

**Table A.3:** LDA topics for  $P$ . Top 10 most probable words are provided for each topic

Topics	Top 10 words
Topic 1	bil, bakifrån, påkörd, stanna, släppa, tur, påkörning, förbi, köbildning, sakta
Topic 2	förare, se, äta, köra, hastighet, börja, km, hålla, plötslig, stanna
Topic 3	väja, väg, förare, älga, kollision, undvika, rådjur, försöka, bil, springa
Topic 4	körd, okänd, anledning, väg, dike, sid, träda, singelolycka, sätta, därefter
Topic 5	fotgängare, övergångsställe, påkörd, gå, bussa, buss, övergångsställe, gata, tr, kvinna
Topic 6	plats, polis, skada, förare, avvika, person, lämna, patrull, smita, ambulans

## A. LDA topics

---

Topic 7	bil, korsning, ljus, kollidera, röd, grön, krocka, körd, svänga, trafikljus
Topic 8	bil, stå, stilla, bakifrån, stillastående, tur, körd, vänta, denna, kö
Topic 9	bil, möta, körbana, kollidera, sid, fela, körfält, motsatt, krocka, möte
Topic 10	bil, bromsa, hinna, stanna, bromsare, framförvarande, kraftig, stoppa, bakomvarande, uppfatta
Topic 11	cyklist, cykla, cykel, cykelbana, korsa, bilist, övergångsställe, påkörd, gata, gång
Topic 12	bil, korsning, svänga, köra, körande, kollidera, norrut, söderut, riktning, färdas
Topic 13	bil, körd, sida, svängd, uppmärksam, baka, varpå, bogsera, följd, ini
Topic 14	fordon, körd, personbil, inblandad, kollidera, framförvarande, seriekrock, förare, samband, avstånd
Topic 15	bil, väga, svänga, köra, sida, se, krocka, kollidera, korsa, denna
Topic 16	lb, lastbil, släpa, körfält, fila, traktor, släp, byta, ligga, välta
Topic 17	riktning, lv, färdas, norrut, söderut, avfart, höja, framföra, strax, km
Topic 18	vägbana, sikta, sikte, råda, mörk, kraftig, dagsljus, skymma, regna, torr
Topic 19	moped, mopedist, omkull, falla, passagerare, slå, mark, ramla, skada, ramlare
Topic 20	förare, tappa, körd, kontroll, väg, fordon, dike, somna, lyktstolpe, träda
Topic 21	bil, sladda, snurra, sid, därefter, rund, mitträcke, åka, körfält, vägräcke
Topic 22	bil, svänga, vänstersväng, omkörning, påbörja, llb, kollidera, samband, denna, samtidigt
Topic 23	bil, korsning, kollision, kollidera, väjningsplikt, uppstå, iaktta, huvudled, stopplikt, trafikant
Topic 24	bil, parkera, körd, stå, backa, parkering, sol, backe, se, förare
Topic 25	olycka, bil, körd, inträffa, förare, ske, reg, oklar, sakna, not
Topic 26	mc, förare, körd, omkull, motorcykel, kurva, välta, glida, bromsa, tunga
Topic 27	bil, körd, rondell, part, lämna, företräde, köra, sida, refug, cirkulationsplats
Topic 28	väg, meter, dike, fortsätta, sid, träda, körd, bromsspår, åka, därefter
Topic 29	dike, sladda, hamna, volta, väg, tak, sid, åka, vägbana, ligga

**Table A.4:** LDA topics for  $P_E$ . Top 10 most probable words are provided for each topic

Topics	Top 10 words
Topic 1	kollission, frontalkollision, sammanstötning, kollison, frontalkollis-sion, krock, kollissionen, kollisonen, kollusion, kollition
Topic 2	mede, företräde, huvudled, stopplikt, högerregel, väjningsplikten, väjningplikt, väjninsplikt, väjningsplik, högerreglen
Topic 3	vajerräcket, vägräcke, sidoräcket, mittvajern, vajerräcke, räcke, mit-träcke, broräcket, räcka, mittbarriären
Topic 4	högerkurva, nedförsbacke, uppförsbacke, svacka, vänsterböj, höger-böj, västerkurva, vänsterkurva, kurva, kruvan
Topic 5	möta, fela, norrgående, mede, körbana, körfält, mötande, mot-gående, höger, motstå
Topic 6	kungsgatan, stopplikt, fyrvägskorsning, korsn, kors, korsnigen, cirkulationsplats, vägkorsning, rondell, korsning
Topic 7	plats, förare, olycka, polis, skada, passagerare, mede, avvika, föra, uppgift
Topic 8	riktning, mede, lv, färdas, rv, väga, norrut, söderut, framföra, körande
Topic 9	trafikljus, gul, rödljuset, stoppljus, rödljus, trafiksignal, ljussignal, signal, ljus, ljuda
Topic 10	uppmärksammas, observera, uppfatta, upptäcka, märka, notera, uppmärssammade, se, uppmärksamma, veta
Topic 11	ca, förare, meter, mede, därefter, träda, km, volta, fortsätta, ratt
Topic 12	framförvarande, bakdel, köbildning, baka, bak, bakpartiet, bakom-varande, bakände, baktill, bakända
Topic 13	snömodd, snöoväder, dimma, modd, snöa, is, regna, snöfall, regnig, ishalka
Topic 14	cirkulationsplats, vägkorsning, väjningsplikten, korsning, rondell, tunnel, stoppskylt, samhälle, cirkulation, parkeringsområdet
Topic 15	oklar, oförklarlig, oförklarig, synbar, okönd, obekant, okönt, oknd, outgrundlig, oländ
Topic 16	hastighet, sjukdom, misstänka, drabba, sjukdomsfall, oregistrerad, mt, omkörningsolycka, halkolycka, skoterolycka
Topic 17	extrem, underlag, vägunderlaget, mycken, vägbana, halkig, glashal, blöt, moddigt, moddiga
Topic 18	lb, filbyte, inbromsning, lastbil, körfältsbyte, mede, omköring, un-danmanöver, högersväng, manöver
Topic 19	passera, längas, passerad, svänga, parallell, ansluta, permans, mar-tinssons, passer, elsa
Topic 20	kasta, vattenplaning, panik, retursladd, sladda, motorstopp, bred-sladd, bredställ, slad, saldd
Topic 21	kvinn, flicka, pojke, person, dam, fotgängare, gångare, gång-trafikant, övergångsställe, obevakad



## A. LDA topics

---

Topic 22	förare, tappa, kontroll, föra, mena, försöka, mede, hona, vägbana, fuktig
Topic 23	förare, öga, stark, solljus, låg, föra, solsken, morgonsol, kvällssolen, sol
Topic 24	hjort, vildsvin, rena, rådjur, katta, hare, älga, grävling, älgkalv, älgko
Topic 25	sikta, mede, gode, dagsljus, vägbana, mulen, siktförhållanden, finta, skymning, sikte
Topic 26	slå, volta, varva, rotera, tumla, kränga, studsare, grad, halv, baklänges
Topic 27	cyklist, trel, cykla, cykelbana, cykel, part, buss, busa, mopedist, omkull
Topic 28	kasa, kana, trilla, tippa, kase, tippare, flyga, ramla, fara, fora
Topic 29	vänstersväng, körfältsbyte, högersväng, omköring, rundpall, vänstersväng, vänsterväng, vänstersväng, usväng, vänstersvän
Topic 30	file, blinka, svänga, gira, svänge, svängd, sväng, vänga, sväga, svägna

# B

## Code Listings

**Listing B.1:** Document Ranking by TFIDF, query expansion and WMD

```
def tfidf_search(query, documents, ref_documents, num_results):
    """ Retrieve results according to similarity in TFIDF vectors of
        query and document pairs
        ref_documents - reference to original documents
    """
    Q = list()
    Q.append(query)
    vectorizer = TfidfVectorizer(max_df = 2000, min_df=30)
    X = vectorizer.fit_transform(documents)
    # Get tfidf vector of test document
    vect = vectorizer.transform(Q)
    # cosine similarity
    from sklearn.metrics.pairwise import cosine_similarity
    query_vect = vect[0]
    # dict contains pairwise similarities between query and
        documents
    # Key document: value similarity
    dict = {}
    similarity = cosine_similarity(vect, X)

    import numpy as np
    S = similarity[0]
    for i in range(len(documents)):
        dict[i] = S[i]

    # sort dictionary according to values
    import operator
    ordered = sorted(dict.items(), key=operator.itemgetter(1),
        reverse=True)
    # return relevant results
    query_result = list()
    for i in range(num_results):
        res, _ = ordered[i]
        query_result.append(ref_documents[res])
    return query_result

def query_expansion_search(query, documents, num_results,
    isEnriched, w2vmodel):
    """
        isEnriched - (boolean) if False enrich text
```

```

        W2vmodel - pre trained model, if it is None then the model
        is trained on w2v_train_set
    """
    # Expand documents
    if w2vmodel == None:
        #train wor2vec
        from definitions import train_word2vec
        vec, vocab, model = train_word2vec(w2v_train_set, 300, 6, 5)
    else:
        model = w2vmodel

    # keyword extraction:
    from definitions import extract_keywords
    keywords = extract_keywords(documents, 230, 1100)
    from definitions import enrich_text
    # enrich the query
    (expanded_query, related_terms) = enrich_text([query], [],
        keywords, 5, model)

    if isEnriched == False:
        (final, related_terms) = enrich_text(documents, [], keywords
            , 5, model)

    result = tfidf_search(expanded_query[0], final, documents,
        num_results)
    return result

def wmd_search(query, documents, num_results, model):
    """
        Rank according to Word Mover's Distance and retrieve top
        num_reuslts
        model - pre-trained word embedding model
    """
    dict = {}
    for i in range(len(documents)):
        dict[i] = model.wmdistance(query, documents[i])
    # sort dictionary according to values
    import operator
    ordered = sorted(dict.items(), key=operator.itemgetter(1))
    # return relevant results
    query_result = list()
    for i in range(num_results):
        res, _ = ordered[i]
        query_result.append(documents[res])

    return query_result

```

### Listing B.2: Eliminating Stopwords

```

def eliminateStopwordsDoc(document, stopList):
    """
        Eliminate stopwords in a single document
    """

```

```

A = []
for word in document.split():
    if word not in stopList:
        A.append(word)
return ' '.join(A)

def eliminateStopWordsCorpus(corpus, stopList):
    """
        Eliminate stopwords in whole document corpus
    """
    newCorpus = []
    for document in corpus:
        newCorpus.append(eliminateStopwordsDoc(document, stopList))
    return newCorpus

```

Listing B.3: TFIDF vectorization of documents

```

from sklearn.feature_extraction.text import TfidfVectorizer
def tf_idf_vectorizer(docs, maxdf, mindf, ngram_range_):
    """ Get tfidf vectors for documents 'docs', with params max_df
        and min_df
        Returns a tuple of tfidf vectors and vocabulary (tfidf,
            vocab)
        ngram_range_ is a tuple in some range e.g
    """
    vectorizer = TfidfVectorizer(max_df=maxdf, min_df=mindf,
        ngram_range=ngram_range_)
    tfidf_matrix = vectorizer.fit_transform(docs)

    # Get the vocabulary
    vocabulary_dict = vectorizer.vocabulary_
    vocabulary = []
    for (a, _) in vocabulary_dict.items():
        vocabulary.append(a.encode('utf8'))
    return (tfidf_matrix, vocabulary)

```

Listing B.4: Training Word2Vec word embeddings

```

from gensim.models import Word2Vec
def train_word2vec(docs, num_hidden_layers, window_, mincount):
    """ Train word2vec model to get word vectors,
        returns a tuple of vocabulary and word vectors list
    """
    doc_lst = []
    for doc in docs:
        doc_lst.append(doc.split())

    w2v_model = Word2Vec(doc_lst, size=num_hidden_layers, window=
        window_, min_count=mincount, workers=4)
    vocabulary_tuple = w2v_model.vocab.items()
    vocabulary = list()
    for (a, _) in vocabulary_tuple:
        vocabulary.append(a)

```

```
word_vectors = []
for word in vocabulary:
    word_vectors.append(w2v_model[word])

return (word_vectors, vocabulary, w2v_model)
```

**Listing B.5:** Keyword extraction and text enrichment

```
def extract_keywords(documents, low, high):
    """
        Keyword extractin based on document frequency values
        Keywords are words with frequency values between low and
        high
    """
    (a, keywords) = tf_idf_vectorizer(documents, high, low, (0,1))
    return keywords

def enrich_text(documents, w2v_train_set, keywords,
                n_related_terms, w2v_trained):
    """
        Expanding text with related words in order to explicitly
        separate it
        from other documents.
        if text contains a sentence with keyword then it will be
        expanded
        by adding related words to that keyword to the sentence
    """

    if w2v_trained == None:
        (vectors, vocab, model) = train_word2vec(w2v_train_set,
                                                300, 6, 1)
    else:
        model = w2v_trained

    # Getting related terms to the keywords
    related_terms = []
    related_terms_dict = {}

    for item in keywords:
        try:
            a = model.most_similar(positive=[item], negative=[])
        except KeyError:
            print item
            string = item
            count = 0
            for (i, _) in a:
                string = string + " " + i

            related_terms.append(string)
            related_terms_dict[item] = string

    documents_split = []
    for doc in documents:
```

```
        documents_split.append(doc.split())

    # text enrichment
    for doc in documents_split:
        for i in range(len(doc)):
            if doc[i] in expanded_keywords:
                doc[i] = related_terms_dict[doc[i]]
    final = []
    for i in documents_split:
        final.append(" ".join(i))

    return (final, related_terms)
```

# C

## Perplexity values for different topics

Table C.1: Perplexity for  $P$ ,  $H$ ,  $P_E$ , and  $H_E$

#Topics	$P$	$H$	$P_E$	$H_E$
2	388.0297	484.0261	599.5403	539.7443
5	316.7467	389.0353	380.5099	333.2821
10	279.0227	344.3034	266.6354	243.4594
15	265.1783	329.3884	223.5553	208.8286
20	257.6856	323.9852	197.3786	189.7355
25	255.2637	323.9608	182.8179	177.8984
30	253.7569	323.2362	176.3308	174.7421
40	254.1955	329.2791	166.1676	169.1734
45	255.7196	333.1078	164.1649	167.8493
50	257.0346	337.6669	164.8295	168.5603
55	260.2828	342.9963	165.6192	169.8557
60	262.6959	346.7295	168.0868	172.1852
65	265.8622	352.0857	169.6564	172.8815
70	268.7501	356.9354	175.5590	176.0444
75	272.3882	362.0802	176.2387	177.7420
80	275.9963	368.0324	178.3309	180.8703
85	278.8546	373.3583	181.3648	183.1144
90	282.6494	379.8532	184.8324	184.8689
95	285.9792	384.9963	188.6076	187.4895
100	290.3668	391.6887	192.0175	190.0082