8-18-2020 10:00 AM

Ranking comments: An Entropy-based Method with Word Embedding Clustering

Yuyang Zhang, *The University of Western Ontario*

Supervisor: Yu, Hao, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Statistics and Actuarial Sciences

Abstract

Automatically ranking comments by their relevance plays an important role in text mining and text summarization area. In this thesis, firstly, we introduce a new text digitalization method: the bag of word clusters model. Unlike the traditional bag of words model that treats each word as an independent item, we group semantic-related words as clusters using pre-trained word2vec word embeddings and represent each comment as a distribution of word clusters. This method can extract both semantic and statistical information from texts. Next, we propose an unsupervised ranking algorithm that identifies relevant comments by their distance to the "ideal" comment. The "ideal" comment is the maximum general entropy comment with respect to the global word cluster distribution. The intuition is that the "ideal" comment highlights aspects of a product that many other comments frequently mention. Therefore, it can be regarded as a standard to judge a comment's relevance to this product. At last, we analyze our algorithm's performance on a real Amazon product.

Keywords: word embedding, word2vec, word cluster, the general entropy, the maximum general entropy comment, K-L divergence.

Lay Summary

Gathering information based on other people's opinions is an essential part of the purchasing decision process. With the rapid growth of the Internet, these conversations in online markets provide a large amount of product information. So when doing online shopping, consumers rely on online product comments, posted by other consumers, for their purchase decisions.

In this thesis, we propose a new method to identify relevant comments under a product. Our method is sensitive to the content of a comment and can successfully filter out unrelated comments. By ranking these relevant comments higher, consumers can better evaluate the true underlying quality of a product.

Acknowledgements

In the process of writing this thesis, I received a great deal of support and assistance. First of all, I would like to extend my sincere gratitude to my supervisor, Dr. Hao Yu, for his patience, encouragement, and professional instructions during my thesis writing.

Secondly, I would like to express my sincere thanks to my thesis examiners, Dr. Xingfu Zou, Dr. Wenqing He, and Dr. Douglas Woolford, for taking the time out of their busy schedule to read my thesis. Their constructive and useful comments are very much appreciated.

I also owe thanks to my friends and my classmates who gave me their help and time to listen to me and help me out of difficulties.

Last but not least, my gratitude extends to my parents, who have been assisting, supporting, and caring for me all the time.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

Nowadays, online shopping has become popular all over the world. The total spending of Canadian online shoppers reached $57.4 billion in 2018, compared to $18.9 billion in 2012, with nearly 84% of Internet users buying goods or services online, which means more than 8 in 10 Canadians shopped online [2]. There are many benefits of online shopping. The most obvious benefit of online shopping is convenience, shoppers can simply access online stores from their computer whenever they have free time available. Another benefit is that online shopping provides a greater diversity of products. This means you can choose goods that suit your requirements and budget the most. However, there are also disadvantages of online shopping. One of the most obvious ones is the lack of interactivity. You can not touch and feel the product you want to buy. Besides, the lack of touch and feel creates concerns over the quality of the product. For example, many people don't like buying shoes online since people will not know if shoes fit unless they try it. With a large variety of goods and websites, people tend to do a lot of research before making a purchasing decision when doing online shopping. They will browse web pages about product details and, more importantly, check other buyer's comments on the product site.

Gathering information based on other people's opinions is an essential part of the purchasing decision process [8]. With the rapid growth of the Internet, these conversations in online markets provide a large amount of product information. So when doing online shopping, consumers rely on online product comments, posted by other consumers, for their purchase decisions.

However, a large number of comments for a single product may make it harder for people to evaluate the true underlying quality of a product. In this situation, consumers tend to focus on the average rating of a product, like the number of stars on Amazon.com. But in reality, some products can easily obtain high average ratings by cheating while some other products may get unfair low ratings. Therefore, it is very important to extract these relevant and high-quality comments from the product site, which can help consumers obtain accurate information about this product.

1.1 How to judge a comment's quality?

Before we start to construct a comment ranking algorithm, the fundamental question is how to judge a comment's quality. Most online business sites evaluate their comments' quality using criteria such as overall rating or helpfulness. Helpfulness is typically a score measured as the total votes given by consumers, which is an interesting way of defining a comment's relevance and quality. Many researches in comments ranking area also use this type of helpfulness score as their comments' evaluation score [32]. However, this method fails to identify these most recent comments with few votes. For example, we may always observe that only a few comments published a long time ago have a high helpfulness score in a product site, and most other comments have no votes. The reason for the phenomenon is that most people only read the first few pages of comments before making their purchase decisions. A new comment that has just appeared on the product site and has not received any votes until recently may remain at the bottom of the comment list. This comment may contain important information about this product, thus has the potential to rise to the top of the list.

There is also another type of comment's quality evaluation method, Chen and Tseng [7] construct an information quality (IQ) framework for Internet product reviews. The IQ framework is a multi-dimensional framework in which each dimension represents a single aspect or construct of information items and is described by a set of features. Chen and Tseng treat each comment as an information item and construct fifty-one features from each comment, including the content of comments and believability of this product and comment authors' reputation. Their framework is able to make a detailed comparison between each comment and also have a good performance when ranking comments. However, their IQ framework requires too much information where most online business websites may not be able to provide, like authors' reputation and product's believability, and some of the features need human-annotated. Their IQ framework may not be feasible in most applications.

So how to judge each comment's quality with only a collection of comments under a product? In our research, we judge each comments' quality by their **relevance** and **text complexity**. We want to explain this by using an example. Table 1.1 below shows five comments of one of my favorite book "The Little Prince" on Amazon. It's very easy for us to see that the first three comments have obviously better quality than the last two. Comment #4 has only two words "Fantastic book", actually using these two words to describe this book is not inappropriate. So in terms of **relevance**, we can see that this comment is relevant to this product. However, in terms of **text complexity**, this comment is too concise and contains little information. Combining these two features together, we can easily see why this comment is not as good as the first three comments. Similarly, comment #5 is apparently an advertisement. In terms of **text complexity**, it's better than comment #4 but for **relevance**, it has almost no relevance to this product.

#	**Comment**	**Helpfulness**
1	One of the most influential books every read. Never forget the light of the child. It is spoken through so many true persons if "time" gone by.	2
2	I bought this book to read to my students. The story is very cute but a little overwhelming with the underlying Concepts in the book. Although it is a story you can read to young children I would recommend waiting and having an older child read this book.	1
3	My favourite book arrived right on time. It has a folded corner but nothing that can't be ignored. This is the best book in the world. It's been written for little kids but has a lot of life lessons for us, the grown ups, who have forgotten about that kid that we once were. I highly recommend everyone to read it, at least once.	1
4	Fantastic book!	0
5	Please use this link http://bit.ly/125 to buy "wiki" book.	2

Table 1.1: Example of Product Comments

Relevance is a measure of how a comment is releted to this product and **text complexity** is a measure of a comment's information richness. In our studies, we use these two features to judge a comment's quality.

1.2 Ranking Comments using Entropy

Ranking comments can be a very important task, and there is no doubt that there are many studies in this field. Some researches treat this ranking task as a supervised learning task, like [13] [3]. Most of them used the consumers' votes, like helpfulness scores, as their training target. Then they adopted or designed several statistical or machine-learning models based on training data. As mentioned before, the reliability of this training data is hard to be assured; some high-quality comments may have relatively low votes. Moreover, these supervised ranking models can not be used on multiple products simultaneously since they have to be retrained for different products.

In this situation, we would like to develop a comment ranking algorithm that is **unsupervised**, which means we do not require any human-annotated training set. Besides, as we mentioned before, in most cases, online business sites may not be able to provide some information, like the reviewer's reputation. We prefer to develop an algorithm that ranks comments based on their contents. To construct this ranking algorithm, we need to solve these two problems below..

1. How to define a metric that can evaluate both comments' relevance and text complexity?

2. How to effectively retrive information from comments' content?

Let's take a look at the first question first. When it comes to text complexity or information richness, we will naturally think of **Shannon's entropy** [26]. In statistics, **entropy** is a quantity that can measure any random variable's average rate of information inherent in the variable's possible outcomes. For a discrete random variable X with all possible outcomes $\{x_1, x_2, ...x_m\}$ and probability mass function $P_X(x)$, the entropy is defined as,

$$H(X) = -\sum_{i=1}^{m} P_X(x_i) \log P_X(x_i) = \sum_{i=1}^{m} P_X(x_i) I_X(x_i) = E[I_X], \tag{1.1}$$

where $I_X(x_i)$ is self-information associated with outcome x_i. We can treat $I_X(x_i) = -\log P_X(x_i)$

as a random variable, thus entropy is actually expectation of self-information I_X. Self-information can be regarded as the rate of information associated with one particular outcome of a random variable, and then entropy is the average rate of information of a random variable. To use entropy as a ranking metric, we need to consider each comment as a distribution of words, defined as $\mathbf{P} = [P_0, ..., P_n]$, where n is the number of unique words and P_i is the frequency of ith words in this comment. There are different ways to represent a comment as a distribution, and we will describe later. Entropy is an effective measure of comment's text complexity, Hsu et al [13] also used entropy as a comment complexity measure in their supervised comments ranking application. For example, if a comment only has one type of word in it like "good good good...good", then this comment has a distribution with $P(good) = 1$, and the entropy of this comment is zero.

Take a look at the entropy formula again. We can see that this metric does not measure comment's relevance to the product. So how to redefine entropy and take comment's relevance into account? Zhang [31] defined a new entropy value called **the general entropy**. In his thesis, he developed an unsupervised ranking method on Amazon's dataset and used the general entropy to measure the answer's information quality. The general entropy is defined as follows.

$$E(\mathbf{P}) = -\sum_{i=0}^{n} Q_i \cdot P_i \cdot \log P_i, \tag{1.2}$$

where $\mathbf{P} = [P_0, ..., P_n]$ is the words distribution of an individual comment and $\mathbf{Q} = [Q_0, ..., Q_n]$ is the distribution of the words of all comments combined under the same product, which is called global distribution. We can see that general entropy assigns weight on each self-information of word where the weight is the corresponding word probability in the global comment set. Since we want to measure the information richness and the relevance of a comment, we can give higher weight to these words that other comments also mentioned and lower weight to words that other comments hardly mentioned. More detail about this definition are described in Chapter 3.

The general entropy seems a good ranking metric that can measure both relevance and text complexity. However, in our experiment described in Chapter 4, comment with high complexity (for example, very long comment with many different kinds of words) and almost no relevance to this product can get very high general entropy. These comments may have high ranks since most comments' entropy scores are close to each other. So instead of calculating scores for every comment, we first find an "ideal" comment and then judge comment by how far or how different it is from our "ideal" comment. Naturally, we can define comment with maximum general entropy as our "ideal" comment, and we call this "ideal" comment the **Maxmium General Entropy Comment**. Since the maximum general entropy comment has the maximum general entropy, it keeps a good balance of relevance and information richness. We define the **Maxmium General Entropy Comment** as follows,

$$\mathbf{B} = \underset{\mathbf{P}}{\operatorname{argmax}}\, E(\mathbf{P}). \tag{1.3}$$

Note that the maximum entropy comment is a comment with the maximum general entropy within all possible comments. This comment may not exist in the existing comment set. Now, the question is how to measure each comment' "distance" to this "ideal" comment. Since we treat each comment as a distribution, we can use **Kullback–Leibler (K-L) divergence** defined as follows,

$$D_{KL}(\mathbf{P}|\mathbf{B}) = \sum_{i=0}^{n} P_i \cdot \log(\frac{P_i}{B_i}). \tag{1.4}$$

Notice that this is a divergence, not a "distance". Actually, K-L divergence is used to measure how one probability distribution is different from others. In our application, we need to compare how each comment is different from the maximum general entropy comment. If a comment is similar to the maximum general entropy comment, it can get low divergence. Otherwise, if a comment is very different from the maximum general entropy comment, it may get high divergence. Compare to the method purely using the general entropy, this method achieved better performance in our experiment since it is more sensitive to comments' relevance to the product.

1.3 Text Representation

Now, let's consider the second problem described before: **How to retrieve information from comments' content effectively?**. As we discussed in the previous section, in order to use entropy as our comments' ranking metric, we need to treat each comment as a distribution of words $\mathbf{P} = [P_1, ..., P_n]$. $\mathbf{P}$ is a numerical vector where each dimension indicates the frequency of a word that appeared in the comment, and n indicates the number of unique words in the whole collection of comments. This is actually called the Bag of Words (BOW) model. A bag-of-words is a representation of text that describes the occurrence of words within a document. Despite the simplicity of this representation method, it has two significant disadvantages:

(1) the number of unique words in comments data is about 10000 while each comment has only 10-200 words, using the BOW model will lead to high-dimensional and sparse vectors.

(2) BOW representation doesn't consider the semantic relation between words, it assumes all the words are independent. This assumption may have some problems, for example, "used bicycle" and "old bike" will be considered entirely different phrases because they have no words in common.

In Zhang's research [31], he designed a new text representation method to solve these problems. In his method, firstly, he takes the maximum general entropy comment as a reference to find "keywords" in the global comment set. Then he digitalizes each comment only considering these "keywords" and treats all other words as noise. For example, imagine we have a comment:

```
 ‘‘This is very easy to clean, very adjustable, and cute! Beautiful!
This is the second one I have purchased."
```

And after data cleaning and removing stopwords, we have

```
‘‘easy-clean adjustable cute beautiful product second purchased".
```

Since the vocabulary size is normally far larger than the number of keywords selected, it's possible that only "adjustable" and "product" are selected as keywords based on the maximum general entropy comment. It has seven words in total, then this comment can be digitalized as $[\frac{1}{7}, \frac{1}{7}, \frac{5}{7}]$, the first two dimensions of this vector indicates two keywords "adjustable" and "product" and the last dimension is the frequency of noise words. Finally, he will calculate each comment's general entropy based on the frequency of these keywords and noise and rank comments based on their entropy value. This method can solve the BOW model's sparsity problem, but it only considers a few keywords and treats other words as noise, which may lose a lot of information. In Zhang's thesis, he also tried to solve the semantic relation problem of the BOW model. In his thesis, if two words represent the same meaning, he used one of them to replace another word. For example, he used "baby" to replace words like "diminutive", "wee", "babyish", "tiny" and "youthful". These synonyms are chosen based on a online dictionary called "thesaurus". As we can see, this method can only partly solve the problem, one word may have multiple meanings in different situations. For example, using "baby" to replace every "tiny" in the comments is appropriate and may give the word "baby" an unrealistic high frequency.

In our thesis, we construct an entirely different method that can solve both the BOW model's sparsity and semantic problem. We called our model: the bag of word clusters model. Unlike the traditional bag of words model that treats each word as an independent item, we group semantic-related words as clusters using pre-trained word2vec word embeddings [19] [20]. For example, consider the "used bike" and "old bicycle" example, we have four unique words: "old", "bike", "used" and "bicycle". By using traditional BOW model, we can represent "old bike" as vector $[\frac{1}{2}, \frac{1}{2}, 0, 0]$ and represent "used bicycle" as $[0, 0, \frac{1}{2}, \frac{1}{2}]$. As we can see, these two vectors are orthogonal, two vectors have no element in common, but in reality "used bicycle" and "old bike" are semantic related. Using our methods, we will first group similar words like "bicycle" and "bike" into the same cluster and treat them as the same item. For example, we have two groups: cluster #1: "used", "old"; cluster #2: "bike", "bicycle". Then we can represent "old bike" as $[\frac{1}{2}, \frac{1}{2}]$ where each dimension indicates one cluster, then the bag of word clusters representation of "used bicycle" is also $[\frac{1}{2}, \frac{1}{2}]$. Using this example, we can see that our method can solve the BOW model's sparsity problem, the number of clusters is significantly smaller than the number of unique words. Also, we can retrieve semantic information from text, similar phrases "old bike" and "used bicycle" are represented as the same vector in our model. Besides, unlike Zhang's method, which only considers keywords and treats all other words as noise, our method keeps most words and treats related words as the same item. We believe our method can extract more information from text, thus has a better ranking performance. More detail about the bag of word clusters model is introduced in Chapter 2.

1.4 Thesis Outline

In this thesis, we propose a new unsupervised method to identify relevant comments under a product. There are several advantages of our ranking methods: a) it is entirely unsupervised, which requires no prior domain knowledge and no training data. Therefore, this method can be applied to most of the products' comments ranking applications. b) This method is sensitive to the content of a comment and can successfully filter out unrelated comments. c) This method has low computational cost since it only requires statistical information from the text.

To help achieve better performance, in Chapter 2, we propose a new text representation method: the bag of word clusters model. Unlike the traditional bag of words model that treats each word as an independent item, we group semantic-related words as clusters using pre-trained word2vec word embeddings and represent each comment as a distribution of word clusters. This method can extract both semantic and statistical information from the text.

In Chapter 3, we give a detailed description of our ranking algorithm. Firstly, we introduced the general entropy, a ranking metric based on the Shannon entropy with respect to the global word distribution of a product's comments. The general entropy is a simple metric that focuses on both text complexity and relevance. Then, we develop an advanced algorithm that we first define the maximum general entropy comment as an "ideal" comment and rank comments based on their Kullback-Liebler (K-L) divergences to this "ideal" comment. This method is more sensitive to the relevance of the comments to the product and has a better ranking performance. Last but not least, we introduced our evaluation metrics: normalized Discounted Cumulative Gain (nDCG).

Chapter 4 introduces our experiment with a real Amazon product using the Amazon product dataset [12] [17]. We applied both pure general entropy and K-L divergence method on this product and K-L divergence method outperforms pure general entropy method in our experiment. By comparing these two methods, we can have a clear view on each method's characteristics and how they distinguish unrelated comments from actual comments. Moreover, we analyzed the relationship between the two ranking metrics of the general entropy method and K-L divergence method.

Chapter 2

Bag of Words Model with Word Embedding Clusters

In most text mining or text analytics applications, the first and fundamental problem is how we are going to represent text as input to our model or algorithm. More specifically, how to represent the text documents to make them mathematically computable. Various text representation methods were proposed during the last few years, the most commonly used text representation model in the area of text mining is called the Vector Space Model (VSM) [16], which aims to represent a text document as numerical vectors. One main advantage of VSM is that it is straightforward to compute the similarity between each vector (document), for example, by using cosine similarity.

One of the commonly used VSM is the Bag of Words model (BOW). A bag-of-words is a representation of text that describes the occurrence of words within a document. And to build a BOW model, people need to provide two things: the vocabulary of known words and a measure of the presence of known words. Given the document collection $D = \{d_i,\ i = 1, 2, 3...n\}$ and m unique words in these documents. Mathematically, each document d_i can be represented by an $m \times 1$ vector $v_i \in R^{m\times1}$. For instance, consider there are three documents in this collection D:

d_1: I like learning text mining.

d_2: What is text mining?

d_3: Apple tastes good.

Now we can make a list of all words in our model's vocabulary. The unique words here (ignoring case and punctuation) are:$\{I, like, learning, text, mining, what, is, apple, taste, good\}$. As you can see, we have 10 unique words and 12 words in total within this collection D.

The next step is to score each word in the documnet. There are many methods of scoring which we will discuss later. Let's consider the simple Boolean first. If a word appears in a document, its corresponding weight is 1; otherwise, it is 0. Since our vocabulary has 10 words, we can use a fixed-length vector representation, with each position in the vector to score a word. Then the vector representations of these three documents are like this,

$$v_1 = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]$$

$$v_2 = [0, 0, 0, 1, 1, 1, 1, 0, 0, 0]$$

$$v_3 = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1],$$

notice that the order of word index is the same as the unique word list above.

The intuition of this model is that the information within a document is from its content, which are words in this case. Documents are similar if they have similar words in it. Since each of the three vectors has a fixed length, we can use cosine similarity to measure their similarity. Consider two vectors $\vec{A}$ and $\vec{B}$ with a fixed length N, cosine similarity is defined as follows,

$$Cosine\ Similarity = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}||||\vec{B}||} = \frac{\sum_{i=1}^{i=N} A_i B_i}{\sqrt{\sum_{i=1}^{i=N} A_i^2}\sqrt{\sum_{i=1}^{i=N} B_i^2}}, \tag{2.1}$$

where A_i and B_i are components of vectors $\vec{A}$ and $\vec{B}$, respectively.

The cosine value ranges between $[-1, 1]$, 1 for vectors pointing at the same direction, 0 for orthogonal, and -1 for vectors pointing in the opposite direction. For documents, the term values are usually non-negative, so the cosine similarity ranges between $[0, 1]$, the higher the value is, the more similar two documents are.

Now we can calculate the similarity between documents d_1, d_2 and d_3 using this formula,

$$\cos(d_1, d_2) = 0.4472;\ \cos(d_1, d_3) = 0;\ \cos(d_2, d_3) = 0.$$

We believe that this result is consistent with our observation, d_1 and d_2 are similar to each other because they are both talking about text mining, d_3 has no relation with d_1 and d_2 thus their cosine similarity is zero.

The BOW model is very straight forward and easy to implement. For the word weight in the BOW model, besides the simple Boolean model, we can also use counts of words, frequency or term frequency–inverse document frequency (tf-idf) as word weight, more information about this is referred to [25].

Despite the simplicity of this representation method, we will face two significant disadvantages if we want to adapt this method on our comments data: (1) the vocabulary size in comments data is about 10000 while each comment has only 10-200 words, using the BOW model will lead to high-dimensional and sparse vectors. (2) BOW representation doesn't consider the semantic relation between words, it assumes all the words are independent. This assumption may have some problems, for example, "used bicycle" and "old bike" will be considered as entirely different phrases because they have no words in common.

In this chapter, we will develop a new text representation method based on the BOW model to overcome these disadvantages. Firstly, we will introduce word embeddings, a learned representation of words where words with the same meaning will have similar representations [16]. Then we will introduce word2vec [19] [20], word2vec is a very effective algorithm to train word embeddings based on the local documents. With these word embeddings, we can easily group similar words as a cluster using the clustering method. Finally, instead of representing document (comment) as "bag of words", we represent them as "bag of word clusters". In this way, we can retrieve semantic information from the text, for example, "bike" and "bicycle" will be grouped together because they have the same meaning. Moreover, the BOW model's sparsity problem will be handled since the number of clusters is significantly smaller than the vocabulary size.

2.1 Word2vec

Just like the BOW model, representing words as vectors is also a very essential topic in text mining and especially natural language processing area. With the increasing applications of machine learning and neural network in the natural language area, we need an efficient word vectorization method that can represent word as vectors and also bring enough semantic information within the vector. Several methods are proposed during the last few years, and one of the most popular methods is word2vec.

Word2vec was created and published in 2013 by a team of researchers led by Tomas Mikolov at Google [19] [20]. Word2vec is a group of related models used to produce word vectors (also called word embeddings). Usually, word2vec can be referred to two model architectures and two related training techniques:

- 2 model architectures: continuous bag-of-words (CBOW) and skip-gram(SG). CBOW aims to predict a center word from the surrounding context in terms of word vectors. Skip-gram does the opposite and predicts the probability of context words from a center word.

- 2 training techniques: negative sampling and hierarchical softmax. Negative sampling defines an objective by sampling negative examples, while hierarchical softmax defines an objective using an efficient tree structure to compute probabilities of appearance for all the vocabulary.

In our application, the skip-gram model with negative sampling will be used and introduced in this chapter, for a detailed explanation of other models in word2vec, one can refer to [24]. Moreover, since skip-gram is a neural network model, if you are not familiar with the neural network model, you can refer to [10].

2.1.1 Skip-gram

The basic idea of SG model is to predict context words based on a center word. More precisely, as in a sentence, our model is to take each current word as an input and predict words within a specific range before and after the current word. For example, we have the sentence “I am closing the window”, take the center word ”closing” as an input, the model will be able to predict the surrounding words: “I”, “am”, “the”, “window”. Generally, this model is a classifier on a binary prediction task: “is word w_i likely to appear near word w_j in a sentence?” However, we do not care about the prediction task, and we will take the learned classifier weights W as our word embeddings.

As you can see in Figure 2.1, the structure of skip-gram model is actually a simple 2-layer neural network with only one hidden layer. In our setting, the vocabulary size is V, and the hidden layer size is N. The units on adjacent layers are fully connected. The input is a one-hot encoded vector, which means for a given word w_i in vocabulary, only the *ith* term in vector $[x_1, x_2, x_3, ...x_V]$ will be 1, and all the other terms are 0.

As you can see, there are C panels in the output layer, indicating the predicted multinomial distributions of C surrounding words of the input word. For example, $y_{c,j}$ is the predicted probability of word w_j's presence at position c. Before training the model, you can arbitrarily change the length of context window C and the size of hidden layer N.

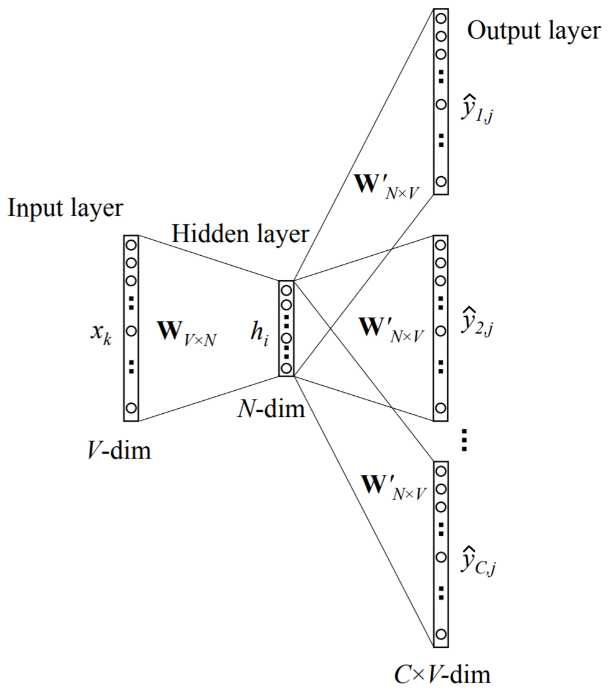


Figure 2.1: Skip-gram model [24]

As you can see from the Figure 2.1, the weight matrix between the input layer and hidden layer is represented by a $V \times N$ matrix $\mathbf{W}_{V\times N}$. Each row of $\mathbf{W}_{V\times N}$ is the N-dimension vector representation of the corresponding word of the input layer. Formally, we denote row i of $\mathbf{W}_{V\times N}$ as $\mathbf{v}_{w_i}{}^T$. Given an input word w_I with index i^*, the input layer would be its corresponding vector

$\mathbf{x}$ with $x_{i^*} = 1$ and $x_j = 0$ for $j \neq i^*$, we have

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(i^*,)} := \mathbf{v}_{w_I}. \tag{2.2}$$

So the hidden layer $\mathbf{h} = \{h_i\}$ is simply copying a row of the weight matrix $\mathbf{W}_{V\times N}$, which means the link function between the input layer and the hidden layer is linear. From the hidden layer to the output layer, we have another weight matrix $\mathbf{W'}_{N\times V}$, and all panels of the output layer share the same weight matrix. For each position in the context of the input word, we can compute a score vector $\mathbf{u_c}$,

$$\mathbf{u_c} = \mathbf{W'}^T\mathbf{h}, \tag{2.3}$$

where c is from 1 to C. Then we denote jth column of $\mathbf{W'}_{N\times V}$ as $\mathbf{v'}_{w_j}$, and each term of the vector $\mathbf{u_c}$ would be,

$$u_{c,j} = \mathbf{v'}_{w_j}{}^T\mathbf{h} = \mathbf{v'}_{w_j}{}^T\mathbf{v}_{w_i}. \tag{2.4}$$

This is the score of word w_j at position c, but not the final output. Remember the output is the predicted probability of each word in the vocabulary at each position in the context, which is a multinomial distribution. Here we use the softmax function to convert each score to a probability, given the input word w_I, the predicted probability of the word w_j appear at position c is,

$$\hat{P}(w_{O,c} = w_j|w_I) = \hat{y}_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V}\exp(u_{c,j'})}, \tag{2.5}$$

where $w_{O,c}$ is the actual context word at position c and $\hat{y}_{c,j}$ is the jth term of the output panel c. Substituting (2.4) into (2.5), we have

$$\hat{P}(w_{O,c} = w_j|w_I) = \hat{y}_{c,j} = \frac{\exp(\mathbf{v'}_{w_j}{}^T\mathbf{v}_{w_I})}{\sum_{j'=1}^{V}\exp(\mathbf{v'}_{w_{j'}}{}^T\mathbf{v}_{w_I})}. \tag{2.6}$$

From Mikolov, et al [20], $\mathbf{v}_w$ and $\mathbf{v'}_w$ are called the "input" and "output" vector representation of word w.

The training objective of this model (for one training sample) is to maximize the conditional probability of observing actual output words $w_{O,1}, w_{O,2}, ..., w_{O,C}$ given the input center word w_I, so the loss function E is the negative log of this conditional probability (since we always want to minimize the loss function),

$$\begin{aligned} E &= -\log \hat{p}(w_{O,1}, w_{O,2}, ..., w_{O,C}|w_I) \\ &= -\log \prod_{c=1}^{C} \hat{p}(w_{O,c}|w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{c,j'})} \\ &= -\sum_{c=1}^{C} u_{c,j_c^*} + \sum_{c=1}^{C} log \sum_{j'=1}^{V} \exp(u_{c,j'}), \end{aligned} \tag{2.7}$$

where j_c^* is the index of the actual c-th output context word in the vocabulary, notice that this model makes a very strong assumption that all context words in different positions are independent of each other. With the loss function E defined, we can now derive the update equation for the weight matrix in the model. Here we used a training technique called backpropagation, which in general is to update weight backward. In our case, it is to update hidden→output matrix $\mathbf{W}'$ first, then update input→hidden matrix $\mathbf{W}$, for more detail about backpropagation, please refer to [10].

First, let us derive the gradient of the loss function with respect to output score $u_{c,j}$,

$$\frac{\partial E}{\partial u_{c,j}} = \hat{y}_{c,j} - y_{c,j} := e_{c,j}, \tag{2.8}$$

where $y_{c,j}$ will be 1 if j is the index of the actual output word and 0 otherwise, note that the derivative is simply the prediction error $e_{c,j}$.

Now let us denote each component of $\mathbf{W}'$ as ω'_{ij}, by using the chain rule, we can obtain the gradient of E with regard to ω'_{ij},

$$\frac{\partial E}{\partial \omega'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial \omega'_{ij}} = \sum_{c=1}^{C} e_{c,j} \cdot h_i. \tag{2.9}$$

Then with learning rate η we can have the update equation for hidden layer→output layer weight matrix $\mathbf{W}'$,

$$\omega_{ij}'^{(new)} = \omega_{ij}'^{(old)} - \eta \cdot \sum_{c=1}^{C} e_{c,j} \cdot h_i \tag{2.10}$$

or

$$\mathbf{v}_{w_j}'^{(new)} = \mathbf{v}_{w_j}'^{(old)} - \eta \cdot \sum_{c=1}^{C} e_{c,j} \cdot \mathbf{h}. \tag{2.11}$$

(2.11) is the update equation for the output vector. Now we can move to the input→hidden matrix **W**, first we take derivative of E on hidden layer h_i,

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V}\sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial h_i} = \sum_{j=1}^{V}\sum_{c=1}^{C} e_{c,j} \cdot \omega_{ij}' \tag{2.12}$$

and take x_k as the kth unit of the input layer, using the chain rule again, the derivative of h_i with respect to each element of **W** is

$$\frac{\partial E}{\partial \omega_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial \omega_{ki}} = \sum_{j=1}^{V}\sum_{c=1}^{C} e_{c,j} \cdot \omega_{ij}' \cdot x_k. \tag{2.13}$$

Remember the input **x** is a one-hot encoded vector, only one component is 1, and all the others are 0. So at each iteration, only one row of the weight matrix **W** will be updated, which is the ”input vector” $\mathbf{v}_{w_I}$, and the update equation is

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta \cdot \mathbf{EH}, \tag{2.14}$$

where **EH** is a N-dimensional vector with each element EH_i is defined as

$$EH_i = \sum_{j=1}^{V}\sum_{c=1}^{C} e_{c,j} \cdot \omega_{ij}'. \tag{2.15}$$

Now we have already obtained the update equation of **W** and $\mathbf{W}'$, when training this model, at each iteration, the output matrix $\mathbf{W}'$ will be update first using equation (2.11) and then we use the updated matrix $\mathbf{W}'^{(new)}$ to update matrix **W** by using equation (2.14). Normally, we use the input vector $\mathbf{v}_{w_I}$ as our learned vector representation of words.

2.1.2 Negative sampling

In the previous section, we discussed the original form of the skip-gram model without any optimization techniques. As you can see, there is a problem with the update equation (2.11): its computational complexity is too large. During training process, for each training iteration, it will scan all the words w_j in the vocabulary and compute its predicted probability and error. Since there are C context words, the computational complexity will be $O(C \times V)$ while the vocabulary size V can go easily beyond 10^4. Doing such computation at each iteration is very expensive. And to get a high-quality vector representation of words, the model needs to be trained on a large amount of text data.

To fix this problem, Mikolov et al. proposed a method in [20] called negative sampling. Instead of updating every output vectors at each iteration, we only update a sample of them. Of course, we will keep these positive samples w_{c,j_c^*} in our sample set and randomly select a few words in the vocabulary as negative samples.

Before training, people can determine a distribution themselves for sampling negative samples, in Mikolov et al.'s paper [20], they used a unigram distribution:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}, \tag{2.16}$$

where α is commonly set as 0.75. We can choose k noise words for every postion c according to this distribution, and these groups of noise words are denoted as $W_{c,neg} = \{w_{c,j} | j = 1, 2, 3, ..., k\}$. For the loss function E, instead of using a multinomial distribution like 2.7, Mikolov et al defined a simplified loss function,

$$E = -\sum_{c=1}^{C} log\sigma(u_{c,j_c^*}) - \sum_{c=1}^{C} \sum_{j' \in W_{c,neg}} \log \sigma(-u_{c,j'}), \tag{2.17}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function. To obtain the update equation, let's first take derivative of E on output score $u_{c,j}$,

$$\frac{\partial E}{\partial u_{c,j}} = \begin{cases} \sigma(u_{c,j}) - 1, & if\ w_{c,j} = w_{c,j_c^*} \\ \sigma(u_{c,j}), & if\ w_{c,j} \in W_{c,neg} \\ 0, & otherwise \end{cases} \tag{2.18}$$

$$= \begin{cases} \sigma(u_{c,j}) - y_{c,j}, & if\ w_{c,j} \in \{w_{c,j_c^*}\} \cup W_{c,neg} \\ 0, & otherwise \end{cases}. \tag{2.19}$$

Remember that w_{c,j_c^*} is the actual *cth* output context word and $y_{c,j}$ equals to 1 only if $j = j_c^*$ and equals to 0 otherwise. Now we can obtain the update equation for the output vector,

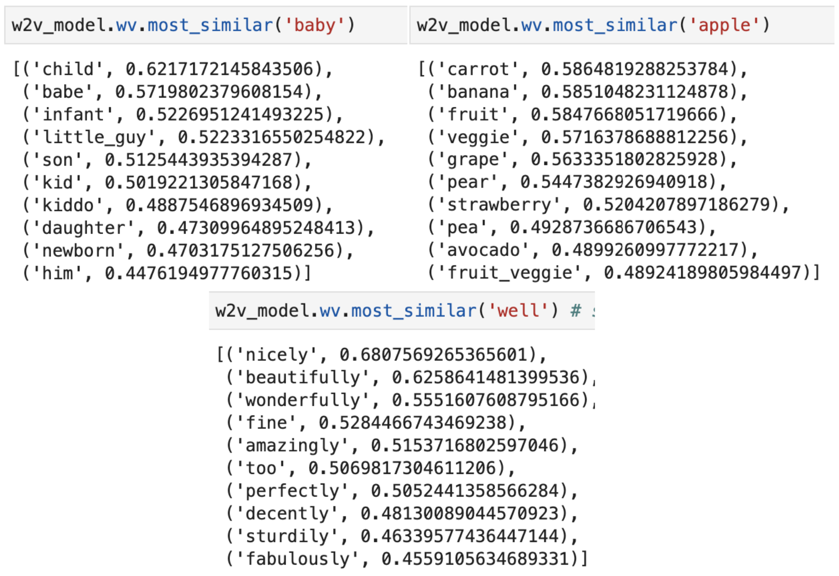
$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial \mathbf{v}'_{w_j}} = \begin{cases} \sum_{c=1}^{C}(\sigma(u_{c,j}) - y_{c,j}) \cdot \mathbf{h} & if\ w_{c,j} \in \{w_{c,j_c^*}\} \cup W_{c,neg} \\ 0 & otherwise \end{cases} \tag{2.20}$$

$$\mathbf{v}'^{(new)}_{w_j} = \begin{cases} \mathbf{v}'^{(old)}_{w_j} - \eta \cdot \sum_{c=1}^{C}(\sigma(u_{c,j}) - y_{c,j}) \cdot \mathbf{h} & if\ w_{c,j} \in \{w_{c,j_c^*}\} \cup W_{c,neg} \\ \mathbf{v}'^{(old)}_{w_j} & otherwise \end{cases}, \tag{2.21}$$

as you can see, we only need to update vectors of $w_{c,j} \in \{w_{c,j_c^*}\} \cup W_{c,neg}$, while other output vectors will remain the same. So the computational complexity per iteration will be reduced from $O(C \times V)$ to $O(C \times k)$ where k usually ranges from 5 to 20.

2.2 Word Embedding clusters with K-means

Training using word2vec model produces N-dimensional vectors for each word in our vocabulary. These word embeddings have many good properties that we can use. Since each word embeddings have the same size N, it is easy to measure the distance between a pair of word embeddings. Another property of these pre-tained word embeddings is that semantically related words usually have a close distance. Here we use the cosine similarity defined at (2.1), and the more similar two words are the higher cosine similarity of their word embeddings. Fig 2.2 are three examples of words "baby" "apple" and "well" with their top 10 most similar words in the whole vocabulary with around 10^4 words. More detail about how these word embeddings trained is described in Chapter 4.



```
w2v_model.wv.most_similar('baby')

[('child', 0.6217172145843506),
 ('babe', 0.5719802379608154),
 ('infant', 0.5226951241493225),
 ('little_guy', 0.5223316550254822),
 ('son', 0.5125443935394287),
 ('kid', 0.5019221305847168),
 ('kiddo', 0.48875468969343509),
 ('daughter', 0.47309964895248413),
 ('newborn', 0.4703175127506256),
 ('him', 0.4476194977760315)]
```

```
w2v_model.wv.most_similar('apple')

[('carrot', 0.5864819288253784),
 ('banana', 0.5851048231124878),
 ('fruit', 0.5847668051719666),
 ('veggie', 0.5716378688812256),
 ('grape', 0.5633351802825928),
 ('pear', 0.5447382926940918),
 ('strawberry', 0.5204207897186279),
 ('pea', 0.4928736686706543),
 ('avocado', 0.4899260997772217),
 ('fruit_veggie', 0.48924189805984497)]
```

```
w2v_model.wv.most_similar('well') #

[('nicely', 0.6807569265365601),
 ('beautifully', 0.6258641481399536),
 ('wonderfully', 0.5551607608795166),
 ('fine', 0.5284466743469238),
 ('amazingly', 0.5153716802597046),
 ('too', 0.5069817304611206),
 ('perfectly', 0.5052441358566284),
 ('decently', 0.48130089044570923),
 ('sturdily', 0.46339577436447144),
 ('fabulously', 0.4559105634689331)]
```

Figure 2.2: Word2vec Example

As you can see, the most similar words calculated by cosine similarity are very similar to the target words, for example, “child”, “babe”, “infant”, “little_guy” are indeed very similar with target word “baby”. The top 10 most similar words of “apple” are all fruit or vegetables. And what’s more surprising is that these most similar words are not just semantically related with target words but also syntactically. In the most similar words of “well”, most of the words are adverbs.

As we mentioned before, instead of treating each word as an independent term, we prefer to group similar words as a cluster and treat the words in the same cluster as the same term. Here we need a clustering method to be able to put similar words in the same group as well as keep not similar words in different groups.

The clustering of word embedding is actually applied to various studies. Gonzalez et al. [21] developed a system to extract mentions of adverse drug reactions (ADRs) from the highly informal text in social media. In their model, by grouping word embeddings using k-means, they can assign the same cluster number to similar words as a feature that adds a higher-level abstraction to the feature space. Wang et al. [28] constructed a convolutional neural network

(CNN) for short text classification and used word embedding clusters as a feature extension.

2.2.1 K-means Algorithm

In our method, we adopt the K-means algorithm [11] to perform word embeddings clustering. K-means is a straightforward and efficient algorithm for general clustering. We will introduce how k-means are applied in our method, let us review this algorithm first.

In this clustering problem, we are given n word embeddings as our training set $\{w^{(1)}, w^{(2)}, ..., w^{(n)}\}$ and $w^{(i)} \in \mathbb{R}^N$. The number of clusters is a pre-set parameter K, and the K-means algorithm are as follows:

1. Initialize cluster centroids $\mu_1, \mu_2, ..., \mu_K \in \mathbb{R}^N$ randomly;

2. Repeat until convergence: {

for every $i \in \{1, ..., n\}$, set

$$c^{(i)} := \arg\min_j ||w^{(i)} - \mu_j||^2 \tag{2.22}$$

for every $j \in \{1, ..., K\}$, set

$$\mu_j = \frac{\sum_{i=1}^n 1\{c^{(i)} = j\}w^{(i)}}{\sum_{i=1}^n 1\{c^{(i)} = j\}} \tag{2.23}$$

}

For every repetition, there are two steps, first is to assign each training sample $w^{(i)}$ to its nearest cluster μ_j and update its assigned cluster index c^i. Then, update the cluster μ_j to the mean of the points assigned to it. The K-means algorithm can also be regarded as a coordinate descent on the distortion function J,

$$J(c, \mu) = \sum_{i=1}^{n} ||w^{(i)} - \mu_{c^{(i)}}||^2. \tag{2.24}$$

As you can see, the distortion function is a non-convex function, so the K-means algorithm can easily get stuck in local minima. One common solution to this problem is to run K-means many times with different random initialization of μ and out of all different clusters founded, use the one with the lowest distortion J as our final solution.

Normally, we use cosine similarity to measure the distance between word embeddings as we mentioned before, but K-means use only Euclidean distance as the distance measure. Although other clustering methods can use cosine as a distance measure like K-medoids [15], we still use K-means due to its much higher computational efficiency. And we can justify that for normalized vectors, cosine similarity and Euclidean distance are linearly connected. For two normalized vectors $A = \{A_i\}$, $B = \{B_i\}$ ($\sum A_i^2 = \sum B_i^2 = 1$), the Euclidean distance between A and B is

$$\begin{aligned} ||A - B||^2 &= \sum (A_i - B_i)^2 \\ &= \sum (A_i^2 + B_i^2 - 2A_iB_i) \\ &= \sum A_i^2 + \sum B_i^2 - 2\sum A_iB_i \\ &= 1 + 1 - 2\cos(A, B) \\ &= 2(1 - \cos(A, B)). \end{aligned} \tag{2.25}$$

Note that for normalized vectors $\cos(A, B) = \frac{\sum A_iB_i}{\sqrt{\sum A_i^2}\sqrt{\sum B_i^2}} = \sum A_iB_i$. The higher two word embeddings' cosine similarity is, the closer their Euclidean distance is, which is consistent with our objective. Thus in our application, we will perform k-means on the normalized pre-trained word embeddings.

2.2.2 Bag of word clusters representation

As we discussed at the beginning of this chapter, instead of representing text documents as "bag of words", we would like to represent them as "bag of word clusters". Now we have our pre-trained word embeddings, and then we perform K-means algorithm, which assigns each word a unique cluster index. Then constructing bag of word clusters representation of text document can be summarized as the following steps:

1. Preprocess and tokenize the text (see Chapter 4 for more detail of data preprocess), then each text will be represented as a list of words;

2. Given pre-trained K word cluster, replace each word in the list as its cluster index, if there are unknown words, replace them with $K + 1$, so this vector will be transformed to numerical lists with the number 1 to $K + 1$;

3. Calculate each cluster's frequency in the text list, construct a vector with length k+1

where each term will be the calculated frequency of clusters with the corresponding index number. And this new vector is our “bag of clusters” representation of text.

As you can see, we will be able to transform each text into a $K + 1$-dimensional vector. For example, we have a short text:

```
‘‘I love eating apples, they are delicious"
```

and we have four pretrained word clusters: $C_1 = \{“I”, “they”, “you”\}$, $C_2 = \{“apple”, “pear”, “banana”\}$, $C_3 = \{“is”, “are”, “was”\}$, $C_4 = \{“delicious”, “good”, “tasty”\}$. First, we tokenize this text as a vector $v = [“I”, “love”, “eating”, “apples”, “they”, “are”, “delicious”]$, then replace these words with corresponding clusters $v = [1, 5, 5, 2, 1, 3, 4]$, remember to replace unknown words with ”k+1” which is 5 here. Next we calculate each cluster’s relative frequency: $f_1 = \frac{2}{7}$, $f_2 = \frac{1}{7}$, $f_3 = \frac{1}{7}$, $f_4 = \frac{1}{7}$, $f_5 = \frac{2}{7}$, and represent this text with new vector $v' = [f_1, f_2, f_3, f_4, f_5] = [\frac{2}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{2}{7}]$.

Text digitalization or representing text as numerical vectors is an essential part of every text mining application. Our “bag of word clusters” model can extract not only statistical information but also part of semantic information from text.

Chapter 3

Ranking comments with general entropy

In this chapter, we will introduce our comment ranking method. Firstly, we will introduce **General Entropy** [31] and how to rank comments directly based on the entropy value. Then we will introduce the **Maximum Entropy Comment**, by treating the maximum entropy comment as an ”ideal” comment, we can measure each comments distance to the ”ideal” comment by using K-L divergence and rank comments based on this distance. Moreover, there are two features of our comment ranking algorithm:

1) Our method is unsupervised, which means there is no human-labeled training set we can learn from, all we have is a group of comments without any order. In other words, our method does not depend on an annotated training set;

2) Judging the quality of a comment is subjective, and we can't just create a judgement standard from nothing. So the objective of our method is not to distinguish these top-ranked comments but to make sure those unrelated or “fake” comments have as lower ranks as possible. In general, one of the objectives of our method is to filter out “bad” comments.

Ranking comments can be a very important task. As the internet develops fast, people tend to get information from these comments on websites, for example, when doing online shopping, people always like to check the comments of a product and these comments have a great influence on their decisions. So there is no doubt that there are many studies in this field. Chiao-Fang Hsu et al [13] proposed a machine learning based approach for ranking comments on the social Web. They extract several different features from each comment: comment visibility, user reputation and content-based features, in the content-based features, they also used entropy as a comment complexity measure. Since the comment data they used have already been ranked by community ratings (for example, the number of "likes"), they treat this ranking problem as a regression problem where the objective y is community ratings. Burges et al. [6] designed a cost function for the ranking problem and constructed a neural network model, Ranknet. This model is simple to train and has excellent performance with a large amount of data. Tulio et al. [3] developed an application to filter out spam comments on youtube called TubeSpam. They vectorized each comment using the BOW model with term weight as term frequency and used these vector as input to their model. Their study compared several different machine learning models like decision tree, SVM, knn, etc. In their application, the Bernoulli Naive Bayes model is chosen since it offered a good balance between robustness and computational effort.

We learned a lot from these studies about how they extract information from these comments. However, most of them used a supervised learning technique with a large amount of pre-classified data. There are also some researches about unsupervised text ranking, but not as many as the supervised ones. Rada and Paul proposed TextRank [18], a graph-based model for text processing. This model's objective is to extract keywords and key sentences from a long document, which is also called text summarization. For key sentence extraction, each sentence will be the node of this graph model and fully connectedto each other, and the transition probability between each node is the calculated similarity between sentences. After setting up the graph model, they let computer simulate traversing the whole graph and calculate the score of each sentence. At last, sentences with high scores will be extracted as key sentences. This algorithm is motivated by Google's PageRank [5] and perform well on long documents like research papers. Vinicius et al. [29] proposed MRR (Most Relevant Reviews), an unsupervised algorithm that identifies relevant reviews based on the concept of graph centrality. The intuition behind this approach is that central reviews highlight aspects of a product that many other reviews frequently mention. MRR is a graphic model where vertices represent reviews connected by edges and each edge is the similarity between a pair of reviews. Then reviews are ranked with the centrality scores calculated by the PageRank algorithm.

The graphic model like TextRank is state of the art in unsupervised text ranking area. However, the graphic model focuses more on the relevance or similarity between each pair of comments. In our methods, we focus more on comments information quality as well as its relevance to the whole group of comments under a specific product.

General Entropy was first introduced by Zhang [31], he developed a ranking method on Amazon's question-answering dataset and used general entropy as a measure of answer's information quality. However, he has an entirely different text digitalization method with us, he only considered each word's frequency in the document and chose some of the most frequent words as keywords while all other words as noise. His method can also solve the high-dimension problem of text digitalization but did not extract semantic information from text. For example, under a food product, most comments described this food as “delicious”, only a few of them used the word “tasty”. If we treat high-frequency word ”delicious” as keyword and low-frequency word “tasty” as noise word, then comments using ”delicious” will be considered having higher information quality than comments using “tasty”, which is not reasonable. While in our digitalization method, we will group similar words like “delicious” and “tasty” and treat them as the same item. There are also other differences in terms of the ranking algorithm, and we will discuss them later in this chapter.

3.1 General Entropy

After the pre-trained word embedding clustering, given n clusters of words and m comments under a product, we regard the collection of all m comments together as the **Global Comments Set**. Then we can calculate the number of each word cluster appears in the global comments set, which can be represented as $\{Num_0^G, Num_1^G, Num_2^G, ..., Num_n^G\}$, notice that Num_0^G is the number of unknown words that appear in the collection. Now we can define the **global probability** of word cluster i in the global comments set as,

$$Q_i = \frac{Num_i^G}{Num_0^G + Num_1^G + Num_2^G + ... + Num_n^G}. \tag{3.1}$$

And for all global probabilities, we have

$$\sum_{i=0}^{n} Q_i = 1. \tag{3.2}$$

Remember that in Chapter 2, to ensure these word embeddings' quality, our word2vec model has to be trained on a large corpus with around 10^4-10^5 unique words. Then we can group these words into n word clusters. One feature of our bag of word clusters model is that these pre-trained word clusters can be used for many products at the same time. So for one product, it's possible that no word within its global comments set falls into the word cluster i^*. In other words, it's possible that global probability $Q_{i^*} = 0$ for this product.

In our comments ranking method, we tend to treat each text or comment as a distribution of word clusters. If two comments have similar distributions, they probably expressed similar meanings. And as an unsupervised method, without any training set, the global comments set's distribution can be an essential reference for determining each individual comment's relevance to others. We believe that under a product, most comments will focus on some specific aspects of this product, which tend to have similar distributions of words, if a comment have a completely different word distribution with the others, it might be a "fake comment."

The same as global probability, for an individual comment with index j, we have the number of each word cluster in the comment as $\{Num_0^j, Num_1^j, Num_2^j, ..., Num_n^j\}$, the probability of word cluster i in the jth comment can be defined as,

$$P_i^j = \frac{Num_i^j}{Num_0^j + Num_1^j + Num_2^j + ... + Num_n^j}, \tag{3.3}$$

where

$$\sum_{i=0}^{n} P_i^j = 1, \tag{3.4}$$

note that $if Q_i = 0$ then $P_i^j = 0$.

As we mentioned in Chapter 2, each comment including the global comments set can be represented by a $n + 1$ dimensional vector, with our new definition, for global comment set, the vector is $[Q_0, Q_1, Q_2, ..., Q_n]$ and for individual comment j is $[P_0^j, P_1^j, P_2^j, ..., P_n^j]$. By treating each comment as a distribution, we can assess each comment's information quality by calculating entropy based on these probabilities.

In statistics, **entropy** is a quantity that can measure any random variable's average rate of information inherent in the variable's possible outcomes, and the concept of entropy was first introduced by Claude Shannon [26]. For a discrete random variable X with all possible outcomes $\{x_1, x_2, ...x_m\}$ and probability mass function $P_X(x)$, the entropy is defined as,

$$H(X) = -\sum_{i=1}^{m} P_X(x_i) \log P_X(x_i) = \sum_{i=1}^{m} P_X(x_i) I_X(x_i) = E[I_X], \tag{3.5}$$

where $I_X(x_i)$ is self-information associated with outcome x_i. We can treat $I_X(x_i) = -\log P_X(x_i)$ as a random variable, thus entropy is actually expectation of self-information I_X. Self-information can be regarded as the rate of information associated with one particular outcome of a random variable, then entropy is the average rate of information of a random variable. As you can see, self-information $I_X(x_i)$ is simply a negative log of the probability of this event, and this is monotonically decreasing in probability $P_X(x_i)$. To help understand self-information, take a simple example, imagine a person randomly select a book from a library, and he observes that the word "the" is in this book. This observation hardly gives him any information since the word "the" has a very high probability appearing in any document. However, if he observes the word "Shakespeare" in that book, he may learn that the book might relate to literature or poetry. The word "Shakespeare" has a lower probability appearing in a book, and this observation definitely contains more information than the last one.

In terms of comments, we treat each comment as a multinomial distribution of word clusters with probability $\mathbf{P}^j = [P_0^j, P_1^j, P_2^j, ..., P_n^j]$, that is if we randomly sample a word from this comment, this word should have this probability distribution. For the worst scenario, if a comment only has one type of word in it like "good good good...good", then this comment has a distribution with $P(good) = 1$, and the entropy of this comment is zero. For the best scenario, without any constraint, the uniform distribution is the maximum entropy probability distribution for a random variable. The reason is that the entropy score is the "expected information gain" and the hardest distribution to predict is the uniform distribution when using a binomial score. [30]. For example, if a comment has an equal probability of every word cluster in it, it would have the maximum entropy. However, if we use entropy defined at (3.3) as our ranking score, a comment with uniform distribution would rank highest under any product, which cannot be used in our application. That is why we have to consider each comment relevance to the others, so we define the **General Entropy** as follows,

Definition 1 (General Entropy) *Given global probability* $\mathbf{Q} = \{Q_0, Q_1, ..., Q_n\}$ *and a comment with probability* $\mathbf{P}^j = \{P_0^j, P_1^j, ..., P_n^j\}$*, the general entropy of this comment is*

$$E(\mathbf{P}^j) = - \sum_{i=0, P_i^j \neq 0}^{n} Q_i \cdot P_i^j \cdot \log P_i^j.$$

Notice that P_i^j can be 0 since a comment may not include all word clusters. The general entropy can measure the average information rate for an individual comment j with respect to the global probability. From the entropy definition at 1, we can see that general entropy assigns weight on each self-information of word cluster where the weight is the corresponding global probability. As we mentioned before, the global probability can be regarded as a high-level abstraction of the topic in the comments under this product. Since we want to measure the information richness and the relevance of a comment, we can give higher weight to these words that other comments also mentioned and lower weight to words that other comments hardly mentioned.

At last, we can summarize our general entropy ranking algorithm as follows:

Algorithm 1 Ranking comments based on the general entropy

```
Input:
    The set of $n$ word clusters;
    The set of $m$ comments under a product;
Output:
    Ranking results of all comments;
1: Covert all comments into their bag of word clusters representations;
2: Calculate the global probability $\mathbf{Q} = [Q_0, Q_1, Q_2, ..., Q_n]$;
3: Calculate each comment's probability: $\mathbf{P}^j = [P_0^j, P_1^j, P_2^j, ..., P_n^j],\ j = 1, 2...m$;
4: Calculate each comment's general entropy $E(\mathbf{P}^j)$;
5: Rank comments based on their general entropy, comment with higher general entropy is
   ranked higher;
6: return Ranking results;
```

3.2 Kullback–Leibler (K-L) divergence to the Maximum General Entropy Comment

In the previous section, we introduced the general entropy, which simultaneously measures information richness and relevance of a comment. However, in our experiment, comment with high complexity (for example, very long comment with many different kinds of words) and almost no relevance to this product can get pretty high general entropy. These comments may have high ranks since most comments' entropy scores are close to each other. So instead of calculating scores for every comment, we first find an "ideal" comment and then judge comment by how far or how different it is from our "ideal" comment. Naturally, we can define comment with maximum general entropy as our "ideal" comment, and we call this "ideal" comment the **Maxmium General Entropy Comment**. Since the maximum general entropy comment has the maximum general entropy, it keeps a good balance of relevance and information richness. We define the **Maxmium General Entropy Comment** as follows,

Definition 2 (Maxmium General Entropy Comment) *Given global probability* $\mathbf{Q} = \{Q_0, Q_1, ..., Q_n\}$*, the maximum general entropy comment* $\mathbf{B} := \{B_0, B_1, ..., B_n\}$ *are defined as*

$$\mathbf{B} = \underset{\mathbf{P}}{\operatorname{argmax}}\, E(\mathbf{P}),$$

where $\mathbf{P} := \{P_0, P_1, ..., P_n\}$ *and* $\sum_{i=0}^{n} P_i = 1$.

Note that the maximum entropy comment is a comment with the maximum general entropy within all possible comments. This comment may not exist in the existing comment set. However, it can be regarded as a standard to judge each comment's relevance to the product. The following theorem shows that the maximum entropy comment exists and is unique, given a collection of comments.

Theorem 3.2.1 *Given gloabl probability* $\mathbf{Q} = \{Q_0, Q_1, ..., Q_n\}$ *and an index set* C *that* $i \in C$ *if* $Q_i \neq 0$ *and* $i \notin C$ *otherwise. Then there exist an unique maximum general entropy answer* $\mathbf{B} = \{B_0, B_1, ..., B_n\}$ *so that* $\mathbf{B} = \operatorname{argmax}_{\mathbf{P}} E(\mathbf{P})$ *and* $B_i = \begin{cases} e^{-1-\frac{\lambda}{Q_i}} & i \in C \\ 0 & i \notin C \end{cases}$*, where* λ *is a unique* *value and* $\sum_{i=0}^{n} B_i = 1$.

Proof For $i \notin C$, $Q_i = 0$, which means word clusters with index i never appear in the collection of all comments. Then for any comments in the collections, $P_i^j = 0$ for $i \notin C$ and $j = 1, 2, ...m$. Since we would like to compare each comment in the collections with the maximum entropy comment, we determine all $B_i = 0$ for $i \notin C$.

For $i \in C$, in order to maximize $E(\mathbf{P})$, we define the objective function f as,

$$f(P_0, P_1, ..., P_n, \lambda) = -\sum_{i \in C} Q_i \cdot P_i \cdot \log P_i + \lambda(1 - P_0 - P_1 - ... - P_n),$$

where λ is the Lagrange multiplier.

Take the derivative of f with respect to P_i

$$\begin{aligned}\frac{\partial f}{\partial P_i} &= -Q_i - Q_i \cdot \log P_i - \lambda \\ &= -(1 + \log P_i)Q_i - \lambda,\end{aligned}$$

then set the derivative to 0,

$$-(1 + \log P_i)Q_i - \lambda = 0.$$

We have the solution to the equation above,

$$\hat{P}_i = e^{-1-\frac{\lambda}{Q_i}} \; for \; i \in C. \tag{3.6}$$

Assume we have more than one element in set C, to solve λ, we have two conditions $1 > \hat{P}_i > 0$ and $\sum_{i=0}^{n} \hat{P}_i = 1$. Based on the first condition,

for all $i \in C$,

$$1 > \hat{P}_i > 0.$$

$\hat{P}_i$ is exponential thus bigger than 0, then for any $i \in C$,

$$\hat{P}_i = e^{-1-\frac{\lambda}{Q_i}} < 1.$$

Take logarithm on both sides,

$$1 + \frac{\lambda}{Q_i} > 0,$$

then,

$$\lambda > -Q_i.$$

Since this inequality holds for all Q_i, to conclude, we have

$$\lambda > -min\{Q_i | i \in C\}. \tag{3.7}$$

Based on the second condition, we set the objective function

$$\begin{aligned} g(\lambda) &= \sum_{i=0}^{n} \hat{P}_i - 1 \\ &= \sum_{i \in C} e^{-1-\frac{\lambda}{Q_i}} - 1. \end{aligned}$$

Taking the derivative of g with respect to λ,

$$\frac{\partial g}{\partial \lambda} = -\sum_{i \in C} \frac{1}{Q_i} \cdot e^{-1-\frac{\lambda}{Q_i}}.$$

The derivative of g is negative for $\lambda > -min\{Q_i | i \in C\}$ which means g is monotone decreasing function, then we have

$$\lim_{\lambda \to -min\{Q_i | i \in C\}} g(\lambda) > 0 \; and \; \lim_{\lambda \to \infty} g(\lambda) = -1.$$

Thus the solution of λ is unique.

In conclusion, we have the maximum general entropy comment $B_i = \begin{cases} e^{-1-\frac{\lambda}{Q_i}} & i \in C \\ 0 & i \notin C \end{cases}$, where λ is a unique value .

After the definition of the "ideal" comment, now we need a method to measure each comment's distance to the maximum general entropy comment. As we mentioned before, we treat each comment as a multinomial distribution with probability $\{P_0^j, P_1^j, ..., P_n^j\}$, that is if we randomly sample a word from this comment, this word can belong word cluster i with probability

P_i^j. We treat the maximum general entropy answer the same way, the maximum general entropy answer is a multinomial distribution with $\{B_0^j, B_1^j, ..., B_n^j\}$, to measure how one probability distribution is different from others, we use **Kullback–Leibler (K-L) divergence** define as follows,

Definition 3 (Kullback–Leibler (K-L) divergence) *Given jth comment probability* $\mathbf{P}^j = \{P_0^j, P_1^j, ..., P_n^j\}$ *and the maximum general entropy comment* $\mathbf{B} = \{B_0, B_1, ..., B_n\}$*, the Kullback–Leibler divergence from the maximum general entropy comment* $\mathbf{B}$ *to jth comment* $\mathbf{P}^j$ *is defined to be*

$$D_{KL}(\mathbf{P}^j|\mathbf{B}) = \sum_{i=0,P_i^j\neq 0}^{n} P_i^j \cdot \log(\frac{P_i^j}{B_i}).$$

In statistics, we call $\mathbf{B}$ the prior probability distribution and $\mathbf{P}^j$ the posterior probability distribution, and the K-L divergence from $\mathbf{B}$ to $\mathbf{P}^j$ is,

$$\begin{aligned} D_{KL}(\mathbf{P}^j|\mathbf{B}) &= \sum_{i=0,P_i^j\neq 0}^{n} P_i^j \cdot \log(\frac{P_i^j}{B_i}) \\ &= -\sum_{i=1,P_i^j\neq 0}^{n} P_i^j \cdot \log(B_i) - (-\sum_{i=1,P_i^j\neq 0}^{n} P_i^j \cdot \log(P_i^j)) \\ &= -\sum_{i=1,P_i^j\neq 0}^{n} P_i^j \cdot \log(B_i) - H(\mathbf{P^j}). \end{aligned} \tag{3.8}$$

According to the entropy defination at (3.5), $D_{KL}(\mathbf{P}^j|\mathbf{B})$ is actually the information gain if we use distribution $\mathbf{B}$ to approximate $\mathbf{P}^j$. When two distributions are close to each other, this value can be relatively small, and large if two distributions are very different. The first item $-\sum_{i=1,P_i^j\neq 0}^{n} P_i^j \cdot \log(B_i)$ in (3.8) are called **cross-entropy**, which is a very popular loss function of classification problem in machine learning area.

Finally, we can summarize our ranking process as follows:

Algorithm 2 Ranking comments based on K-L divergence to the maximum general entropy comment

```
Input:
    The set of $n$ word clusters;
    The set of $m$ comments under a product;
Output:
    Ranking results of all comments;
1: Covert all comments into their bag of word clusters representations;
2: Calculate the global probability: $\mathbf{Q} = [Q_0, Q_1, Q_2, ..., Q_n]$;
3: Calculate each comment's probability: $\mathbf{P}^j = [P^j_0, P^j_1, P^j_2, ..., P^j_n],\ j = 1, 2...m$;
4: Based on the global probability $\mathbf{Q}$, find the maximum general entropy comment: $\mathbf{B} = \{B_0, B_1, ..., B_n\}$;
5: Calculate each comment's K-L divergence to the maximum general entropy comment $\mathbf{B}$;
6: Rank comments based on their K-L divergence, comment with lower divergence is ranked higher;
7: return Ranking results.
```

3.3 Evaluate Ranking Quality with nDCG

In the previous section, we introduced our new comment ranking algorithm, naturally, the next step is to assess this algorithm by evaluating the ranking quality. Here we introduce **normalized Discounted Cumulative Gain (nDCG)** [14], it is often used to measure the effectiveness of web search engine algorithms, but it can also applied to text ranking application. Many research mentioned before [13] [29] adapt this method to assess their ranking algorithm. Firstly, let us define **Discounted Cumlative Gain (DCG)**.

Definition 4 (Discounted Cumlative Gain (DCG)) *Given a ranked list with m comments, and rel_i is graded relevance of the result at position i, Discounted Cumulative Gain is defined as*

$$DCG_m = \sum_{i=1}^{m} \frac{rel_i}{\log_2(i+1)}.$$

According to this definition, if a comment with high graded relevance appears lower in the ranking result, it will be penalized as the graded relevance value is reduced logarithmically proportional to the position of the ranking result. To achieve high DCG value, the algorithm should rank a high relevance comment higher than low relevance one. Notice that in our application, graded relevance rel_i is a manually annotated comment quality score that will not be used as an input in our algorithm, more detail about our experiment are described in Chapter 4.

While DCG is already a valid measure of ranking quality, it does not have a proper upper and lower bound to let people better compare the performance of different ranking results, then **normalized Discounted Cumulative Gain (nDCG)** is defined as follows,

Definition 5 (normalized Discounted Cumulative Gain (nDCG)) *Given a ranked list with m comments and its DCG value, the normalized Discounted Cumulative Gain is computed as,*

$$nDCG_m = \frac{DCG_m}{IDCG_m},$$

where $IDCG_m$ is the Ideal Discounted Cumulative Gain.

$IDCG_m$ is straightforward to compute where the ideal ranking result is to rank these comments directly based on their graded relevance. nDCG ranges from 0 to 1 while 0 will not be able to achieve, and the closer our nDCG is to 1, the better quality our result has.

Chapter 4

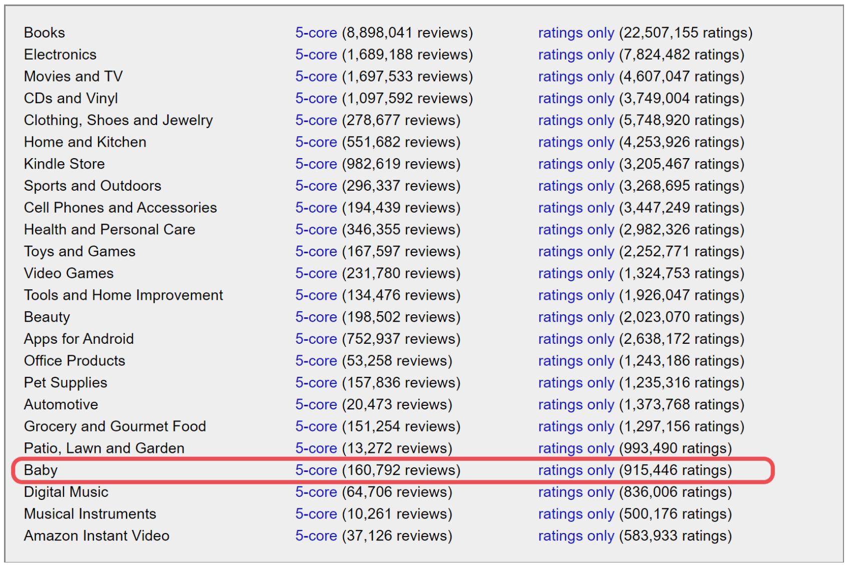
Experiment with Amazon Review Data

In this chapter, we will introduce our experiment built on the Amazon product dataset [12] [17], which contains users' reviews on Amazon website spanning 1996-2014. We will choose one of the Amazon products and rank its comments using both pure general entropy and K-L divergence to "ideal comment". By comparing these two methods on a real dataset, we can understand each method's characteristics and how they distinguish "fake" comments from actual comments. Moreover, we will also analyze the relationship between general entropy and K-L divergence.

4.1 Amazon Product dataset

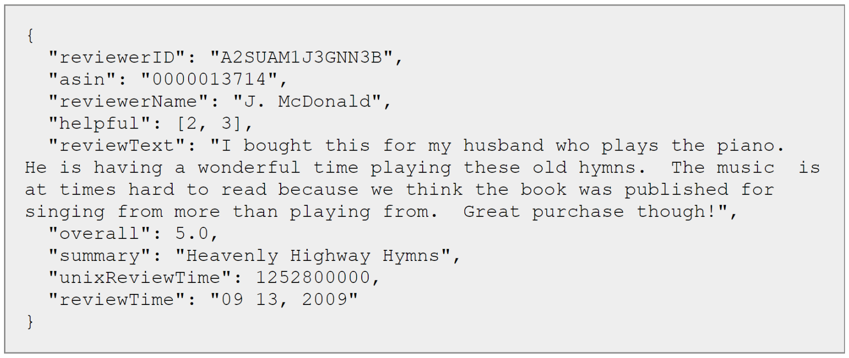
Amazon product data contains 142.8 million reviews from millions of products, and these reviews were grouped into different categories as in Fig 4.1. In our experiment, we chose the category "baby", which includes 160,782 comments of 7701 products. Notice that the dataset is titled "5-core", which means each of the users and products has at least 5 comments. In that case, we can assume that most of the comments in this dataset are reasonable.

Fig 4.2 shows a sample within this dataset, it contains multiple information. In our application, since our method is unsupervised, we only use two parts of this dataset: "asin": ID of the product; and "reviewText": text of this comment.



Books	5-core (8,898,041 reviews)	ratings only (22,507,155 ratings)
Electronics	5-core (1,689,188 reviews)	ratings only (7,824,482 ratings)
Movies and TV	5-core (1,697,533 reviews)	ratings only (4,607,047 ratings)
CDs and Vinyl	5-core (1,097,592 reviews)	ratings only (3,749,004 ratings)
Clothing, Shoes and Jewelry	5-core (278,677 reviews)	ratings only (5,748,920 ratings)
Home and Kitchen	5-core (551,682 reviews)	ratings only (4,253,926 ratings)
Kindle Store	5-core (982,619 reviews)	ratings only (3,205,467 ratings)
Sports and Outdoors	5-core (296,337 reviews)	ratings only (3,268,695 ratings)
Cell Phones and Accessories	5-core (194,439 reviews)	ratings only (3,447,249 ratings)
Health and Personal Care	5-core (346,355 reviews)	ratings only (2,982,326 ratings)
Toys and Games	5-core (167,597 reviews)	ratings only (2,252,771 ratings)
Video Games	5-core (231,780 reviews)	ratings only (1,324,753 ratings)
Tools and Home Improvement	5-core (134,476 reviews)	ratings only (1,926,047 ratings)
Beauty	5-core (198,502 reviews)	ratings only (2,023,070 ratings)
Apps for Android	5-core (752,937 reviews)	ratings only (2,638,172 ratings)
Office Products	5-core (53,258 reviews)	ratings only (1,243,186 ratings)
Pet Supplies	5-core (157,836 reviews)	ratings only (1,235,316 ratings)
Automotive	5-core (20,473 reviews)	ratings only (1,373,768 ratings)
Grocery and Gourmet Food	5-core (151,254 reviews)	ratings only (1,297,156 ratings)
Patio, Lawn and Garden	5-core (13,272 reviews)	ratings only (993,490 ratings)
Baby	5-core (160,792 reviews)	ratings only (915,446 ratings)
Digital Music	5-core (64,706 reviews)	ratings only (836,006 ratings)
Musical Instruments	5-core (10,261 reviews)	ratings only (500,176 ratings)
Amazon Instant Video	5-core (37,126 reviews)	ratings only (583,933 ratings)

Figure 4.1: Amazon Product Dataset



```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

Figure 4.2: Amazon Data Sample

As described in Chapter 2, in order to keep our word embedding's quality, we need a large amount of data to train our word2vec model. In that case, we will combine all comment text in "baby" category as our corpus, which can be used as an input to our word2vec model.

4.2 Data Cleaning

Data cleaning is an essential part of every text mining application. We need to carefully remove all noise or unnecessary words in the text and keep as much information as possible. We performed our data cleaning process using a Python package called Gensim [23]. Gensim is one of the most effective and robust packages to realize unsupervised semantic modeling. It also provides several useful toolkits for text cleaning and processing. We use this package to build our word2vec model.

Now we can summarize our data cleaning process as following steps:

Step 1 Remove non-alphabetic: This step is very straightforward, which is to remove all non-alphabetic characters from the text. Since in our application, we only care about the word distribution, while other elements like punctuation marks or special symbols and numbers have no effect. Moreover, we removed tokens that are too short or too long since after removing non-alphabetic characters, there are always a bunch of garbled characters left. For example, there might be a hyperlink within the raw comment.

Step 2 Remove stopwords: Sometimes, some extremely frequent words have little value in our comments ranking application. These words like “a”, “and”, “the” are called stopwords and it’s always necessary to remove them in text mining tasks. Removing stopwords can help us focus more on those informative words and reduce the dimensionality needed to digitalize each comment. Also, there are many stopword lists avaliable where most of them are pretty similar, and the gensim’s method adapts the stopword list from Stone et al [27].

Step 3 Lemmatization: Lemmatization is the process of converting a word to its base form. Words in English can have different forms, for example, “saw” and “see”, “apple” and “apples”, while computers will tend to treat these words as totally different items. A lemmatizer is able to identify these words and covert them to their meaningful base form. However, precise lemmatization needs every word’s POS tag in the sentence, which is not feasible in our experiment. So we can only lemmatize nouns in the corpus. And the lemmatizer we used is the Wordnet lemmatizer [9] from the Python library called nltk [4].

Step 4 Merging bigram: Bigram is referred as a sequence of two adjacent words in a text. There are many words frequently appear together like “new” “york”, “car” “seat” and “ice” “cream”, and in fact, we would like to treat these bigrams as one item. Therefore, we detect and merge bigrams that appears more than 30 times in our corpus, for example, “new” “york” will be combined as “new_york”.

Table 4.1 is an example of the cleaning process. In Step 3, we can see that plural noun “products” was converted to its base form “product”. And in Step 4, since “easy” and “clean” appear simultaneously in our corpus more than 30 times, they will be merged as one word "easy_clean". All the above is the entire data cleaning process, and we will use the cleaned comments as input to our word embedding training model and ranking algorithm.

Raw Text	"This is very easy to clean, very adjustable, and cute! Love OXO products! This is the second one I have purchased."
Step 1	"This is very easy to clean very adjustable and cute Love OXO products This is the second one I have purchased "
Step 2	"easy clean adjustable cute love oxo products second purchased "
Step 3	"easy clean adjustable cute love oxo product second purchased "
Step 4	"easy_clean adjustable cute love oxo product second purchased "

Table 4.1: Example of Data Cleaning Process

4.3 Word2vec and K-means

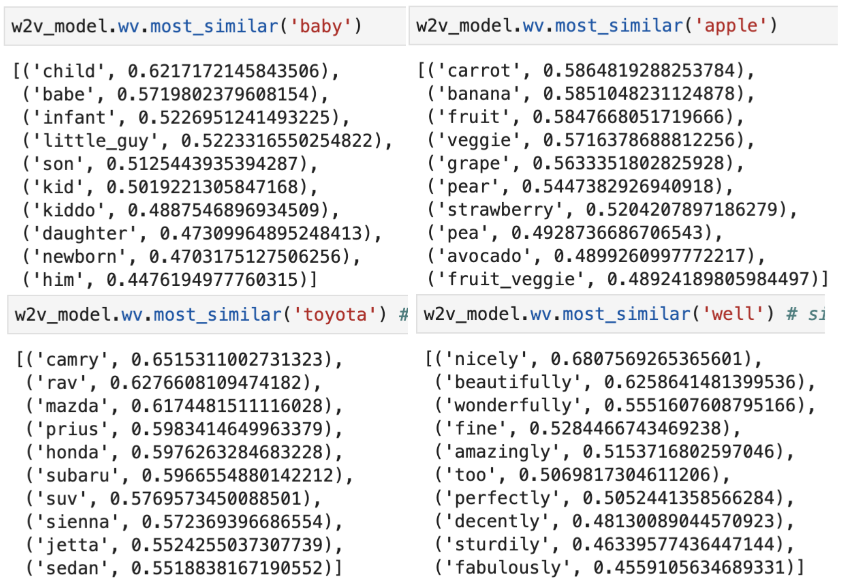
After cleaning the dataset, we can feed the corpus to our word2vec model. As discussed in Chapter 2, we adapt the skip-gram model with negative sampling using the python library Gensim [23]. Table 4.2 shows the parameters of our model. Parameter **Window** is the maximum distance between the current and predicted word within a sentence. For the skip-gram model, our window size is 2, which means we have 1 input word and 4 output words. **Size** is the dimensionality of the word vectors, usually ranges from 100-200, we set as 150 in our model. **Sample** is the threshold for configuring which higher-frequency words are randomly downsampled, since we already removed stop words, we set as 0. **Alpha** and **Min_Alpha** are the initial learning rate and the minimum learning rate respectively. The initial learning rate will linearly drop to the minimum learning rate as training progresses, and this process can help our results converge faster. **Min_Count** is 30, which means that all words with a total frequency lower than this will be ignored in our training process. The last parameter is **Negative**, which indicates how many "noise words" should be drawn when using negative sampling.

We then trained the word2vec model on our corpus, which has 6,032,591 words and 9,497 unique words. The whole training process ran 30 epochs on the data set using the stochastic gradient. The training result is that each unique word in the corpus has a corresponding 150-dimension vector. Figure 4.3 shows the top 10 most similar words of input words: 'baby', 'apple', 'toyota' and 'well'. Here the similarity is cosine similarity between our trained word

Min_Count	30
Window	2
Size	150
Sample	0
Alpha	0.03
Min_Alpha	0.0007
Min_Count	30
Negative	20

Table 4.2: Parameters of Word2vec Model

embeddings. We can observe that these words are semantically similar to each other, which means the word2vec model performs well on this data set.



```
w2v_model.wv.most_similar('baby')

[('child', 0.6217172145843506),
 ('babe', 0.5719802379608154),
 ('infant', 0.5226951241493225),
 ('little_guy', 0.5223316550254822),
 ('son', 0.5125443935394287),
 ('kid', 0.5019221305847168),
 ('kiddo', 0.4887546896934509),
 ('daughter', 0.47309964895248413),
 ('newborn', 0.4703175127506256),
 ('him', 0.4476194977760315)]
```

```
w2v_model.wv.most_similar('apple')

[('carrot', 0.5864819288253784),
 ('banana', 0.5851048231124878),
 ('fruit', 0.5847668051719666),
 ('veggie', 0.5716378688812256),
 ('grape', 0.5633351802825928),
 ('pear', 0.5447382926940918),
 ('strawberry', 0.5204207897186279),
 ('pea', 0.4928736686706543),
 ('avocado', 0.4899260997772217),
 ('fruit_veggie', 0.48924189805984497)]
```

```
w2v_model.wv.most_similar('toyota') #

[('camry', 0.6515311002731323),
 ('rav', 0.6276608109474182),
 ('mazda', 0.6174481511116028),
 ('prius', 0.5983414649963379),
 ('honda', 0.5976263284683228),
 ('subaru', 0.5966554880142212),
 ('suv', 0.5769573450088501),
 ('sienna', 0.572369396686554),
 ('jetta', 0.5524255037307739),
 ('sedan', 0.5518838167190552)]
```

```
w2v_model.wv.most_similar('well') # si

[('nicely', 0.6807569265365601),
 ('beautifully', 0.6258641481399536),
 ('wonderfully', 0.5551607608795166),
 ('fine', 0.5284466743469238),
 ('amazingly', 0.5153716802597046),
 ('too', 0.5069817304611206),
 ('perfectly', 0.5052441358566284),
 ('decently', 0.48130089044570923),
 ('sturdily', 0.46339577436447144),
 ('fabulously', 0.4559105634689331)]
```

Figure 4.3: Word2vec results

The next step is to group similar words using K-means clustering. In our application, we used a very famous Python package scikit-learn [22], scikit-learn contains a group of efficient tools for predictive data analysis and machine learning. There are two parameters when implementing k-means clustering. The first one is the number of clusters. There are many methods determining the number of clusters. Most of these methods are purely based on the quantitative relationship between vectors, but may not lead to a good ranking performance. We have tried the number of clusters from 100 to 500, where 300 clusters lead to the best ranking performance. Another parameter is the number of initializations, which is the number of times the k-means algorithm will run with different centroid seeds. As discussed in Chapter 2, k-means can easily fall in local minima, so it is useful to restart it several times with different initial centroids. We set our number of initializations as 100, then the final result will be the best output of 100 consecutive runs in terms of the loss function.

Cluster#	Semantic category	Examples of clustered words
133	Baby	baby, son, daughter, child, kid, she, little_guy, kiddo, babe,...
11	Automobile	car, trunk, vehicle, drive, suv, truck, sedan, van, ford,...
248	Food	banana, apple, veggie, pea, chicken, meat, pasta, avocado,...
116	Adverb	well, fine, perfectly, nicely, properly, beautifully, poorly,...
104	Media	picture, movie, video, image, show, pic, visual, television,...

Table 4.3: Examples of the word embedding clusters

Table 4.3 shows part of the results of our word clusters, notice that the "Semantic category" titles are manually assigned and not used in the ranking algorithm. We can observe that in cluster#133, word "she" is in the same cluster as "baby". That's an interesting feature of the word2vec model, remember our data set is the collection of all products' comments under the "baby" category. In these comments, "she" always indicates a "baby", where these two words have similar context words thus have similar word embeddings. That's also why word "poorly" and "perfectly" are in the same cluster, despite the two words are antonyms, they have similar context words in the corpus. This property does not affect our application since we only care about each comments' relevance, criticism and praise of a product are both information-rich comments.

Assigning similar words with the same cluster number can help us distinguish comments more accurately. For example, if most of the words in a comment are in word cluster#11, we can tell that this comment might be under a car-related product. Moreover, if a comment has an entirely different distribution of word clusters with other comments under the same product, this comment may not be relevant.

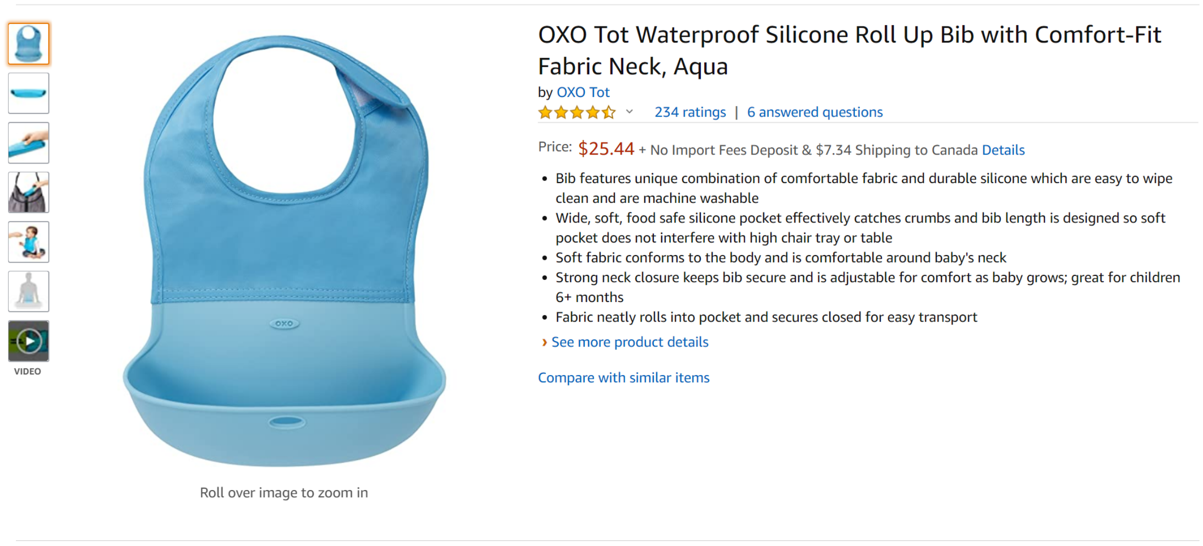
Now we get our word embedding clusters ready, we are able to transform each comment to its "bag of word clusters" representation. Table 4.4 shows how we transform a comment into its bag of word clusters representation. We first covert every word in the comment to its cluster# and then transform to a 300-dimension vector where each element is its frequency of the corresponding cluster.

Raw Text
“This is very easy to clean, very adjustable, and cute! Love OXO products! This is the second one I have purchased.”
Cleaned Text
“easy_clean adjustable cute love oxo product second purchased ”
Covert Words to Cluster#
[170,24,62,221,212,66,199,274]
Bag of Word Clusters Representation
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Table 4.4: Examples of bag of word clusters representation

4.4 Ranking Comments under a Real Amazon Product

Finally, after all the text digitalization process, we can apply our ranking algorithm on a real product. We used a product called "OXO Tot Waterproof Silicone Roll Up Bib with Comfort-Fit Fabric Neck"(ASIN: B00D3TPGAO) [1]. This product also belongs to the "baby" category, which means we will have no difficulties transforming comments to their Bag of word clusters representation. The detail of this product is shown in Figure 4.4.



OXO Tot Waterproof Silicone Roll Up Bib with Comfort-Fit Fabric Neck, Aqua

by OXO Tot

234 ratings | 6 answered questions

Price: $25.44 + No Import Fees Deposit & $7.34 Shipping to Canada Details

- Bib features unique combination of comfortable fabric and durable silicone which are easy to wipe clean and are machine washable
- Wide, soft, food safe silicone pocket effectively catches crumbs and bib length is designed so soft pocket does not interfere with high chair tray or table
- Soft fabric conforms to the body and is comfortable around baby's neck
- Strong neck closure keeps bib secure and is adjustable for comfort as baby grows; great for children 6+ months
- Fabric neatly rolls into pocket and secures closed for easy transport

› See more product details

Compare with similar items

Figure 4.4: Product Detail [1]

As we mentioned before, this data set follows the 5-core principle, where all users and items have at least 5 reviews. This product has 95 comments in total. We checked all comments and made sure that they are all related to the products.

4.4.1 Global Word distribution

Let us first take a look at the word cluster distribution of all comments, which we also called global probability in Chapter 3. Figure 4.5 is the histogram of global word distribution, where the horizontal axis represents the cluster index, and the vertical axis represents frequency.

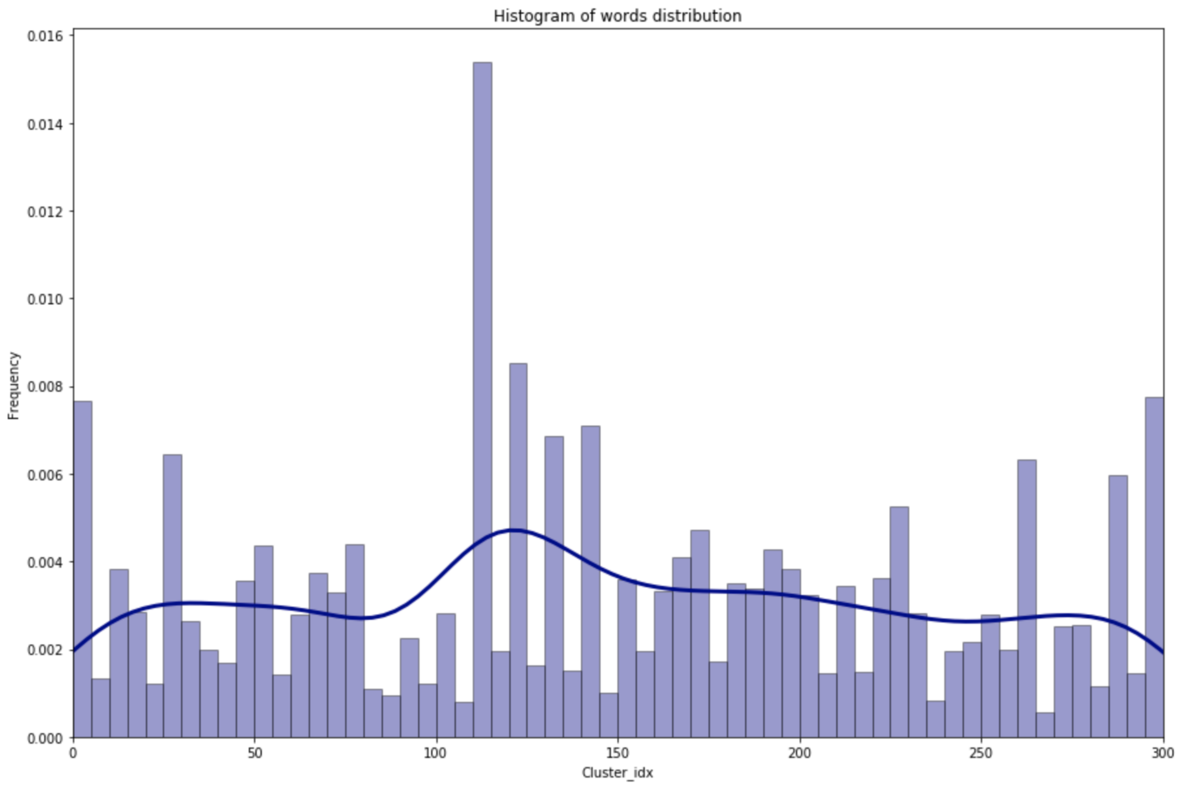


Figure 4.5: Global Word Distribution

The word clusters distribution is not very uniform, some clusters have significantly higher frequencies than the others. Table 4.5 and Table 4.6 show the top 3 most frequent and least frequent word clusters in the whole collection of comments. Obviously, one of the most frequent words under this product should be “bib”, and its corresponding word clusters has the highest frequency. It is not just because this cluster includes the word “bib”, it also includes many related words like “spoon”, “fork”, “catch_food’, which may also appear a lot in the comments. The second most frequent cluster includes “baby” related words; these words appear a lot in the comments as this is a baby product. The third cluster is “easy” related words; many users describe this product using these adjectives. If these clusters have high frequency appearing in a comment, this comment is likely related to our product.

Cluster#	Frequency	Examples of clustered words
112	0.0701	bib, spoon, bowl, plate, dish, fork, catch_food...
133	0.0299	baby, son, daughter, child, kid, little_guy, kiddo...
1	0.0216	easy, easier, simple, useful, handy, make_easier...

Table 4.5: Top 3 Most Frequent Word Clusters

Cluster#	Frequency	Examples of clustered words
20	0	phone, iphone, tablet, computer, laptop, smartphone...
259	0	brush, hair, bristle, toothbrush, nail_cliper, scalp...
263	0.0047	sleep, sleeping, fall_asleep, cozy, snuggle, cuddle...

Table 4.6: Top 3 Least Frequent Word Clusters

Let us take a look at the top 3 least frequent word clusters. Apparently, words like "phone", "laptop", "hair" and "sleep" have little or no relation to our products. If these clusters appear a lot in a comment, this comment may not be as applicable to our product.

After analyzing the global word cluster distribution, we can see that global probabilities contain a lot of information regarding our product and are capable of judging the relevance of a comment to this product, which partially proves our methods' feasibility.

4.4.2 Ranking Performance

In this section, we will apply our two ranking methods on our dataset: general entropy and K-L divergence to the maximum general entropy comment. In the dataset, we have one product with 95 comments, and these comments are considered as **relevant comment**. Since we will not judge these 95 comments' quality, we arbitrarily select 5 sets of comments that are not related to this product. Each set contains 10 comments. We called these comments **fake comment**, and they are all real comments under other Amazon products. It is worth mentioning that during the experiment, we are not aware of which comment is fake, which is not. We will calculate global distribution and the maximum general entropy comment based on all comments, including fake comments.

To assess the ranking performance, we used the evaluation metric nDCG defined at Definition 5. According to the definition, we need to assign each comment a relevance score rel_i. In our experiment, we assigned the 95 original comments with relevance score **10** and fake comments with relevance score **-10**. To achieve higher nDCG value, original comments should rank higher than fake comments. Moreover, in the best scenario, fake comments happen to have the lowest ranks, we can calculate our ideal DCG (iDCG) based on this case.

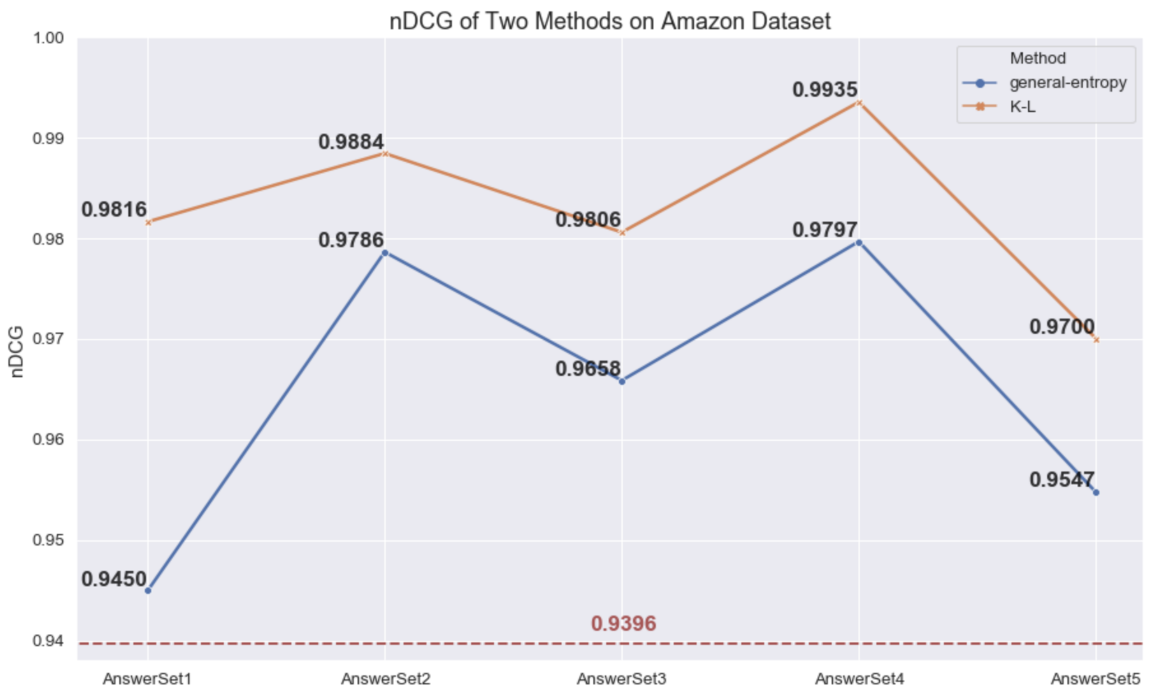


Figure 4.6: nDCG of Two Methods on Amazon Dataset

Figure 4.6 shows the nDCG values of two methods on 5 different fake answer sets. Remember that nDCG ranges from 0 to 1, where a higher score indicates better performance. The red dashed line at the bottom of this figure is the baseline of our application. To construct the baseline, we generate 100000 random sequences as the ranking results and calculate the mean of their nDCG values, which is 0.9396. Both of the two methods have better performance than the baseline, which shows our methods' effectiveness. The K-L method generally outperforms the general entropy method, and they have the same trend. We can observe that when the K-L achieves higher nDCG scores, the general entropy always has a higher score as well. The reason for this phenomenon is that they both rank comments based on their relevance to the global distribution. K-L method is more sensitive to each comment word distribution, and the variation of comments' K-L score is bigger than that of the general entropy method, so it has better discrimination power. We will see more detail about this later.

Table 4.7 below is the ranking detail of the fake comment set #1. The detail of other 4 fake comment sets is in appendix A. Note that **Entropy** and **E-Rank** are comment's general entropy value and the rank based on the entropy, while **K-L** and **K-Rank** are comment's K-L divergence to the maximum general entropy comment and its rank based on K-L value.

#	Comment Text	Entropy	E-Rank	K-L	K-Rank
1	‘I remember the controversy about this movie when it came out in ’83 and it was rightly so. It showed a real ugly side of Miami and the drug wars and this movie didn’t hold back. The chainsaw scene still sends shivers whenever I see it and the movie just continues on delivering a punch right up until the finale. Had a DVD copy of it but the price was too good to resist and now I’m glad to have a great looking Blu-ray copy and get to enjoy this classic again and again.’	0.006987	84	56.6583	102
2	‘My son loves this formula it really helped wean him from breastfeeding. It’s also a good supplement since he’s still not eating a ton and can be a picky with what he does eat. We use the single packets when traveling so we don’t have to bring a big can’	0.005426	96	34.4120	57
3	‘Have to give only 4 stars on this version of the paperwhite. The reader works perfectly. Sits well in your hands. I have no issues with the lighting quality. I absolutely love the flush screen. But, the bezel attracts a lot of fingerprints and my number 1 reason for only giving 4 stars is that the canadian version does not have bluetooth or Audible. I think this is absolutely ridiculous and Amazon has no reason for not including this. It’s in the international version if bought from Amazon.com but not if you but from Amazon.ca.’	0.008193	68	55.6625	101

4	‘This is definitely the worst car seat I’ve ever encountered. I’ve dealt with about 15 carseats in my time. (1) it’s false advertising: I scoured the listing to ensure it doesn’t have the ARB and nothing stated it was included. I even searched model # separately etc, and nothing. It arrives and course it has the ARB, which is super dangerous (my daughter has entangled her feet in it a few times making me pull over ON THE HIGHWAY to help release her little feet). (2), it is extremely difficult almost impossible to install and once done, it is STILL UNSTEADY AND WOBBLY. Horrible. (3) the buckles aren’t very effective as my daughter has been able to remove it more than once and she just turned one and has never unbuckled anything else in her life. (4) once buckled, the belts often cannot be tightened because something awful in this horrendous company’s design prohibit the belt from being pulled through the tiny crack they’ve allowed for it. It’s just bad bad bad. literally the worst. I’m embarrassed and sad to admit I haven’t taken my baby out as often as a result of this since surgery complications left me with low tolerance for abdominal pain. I’m now recovered but feel as if im about to have a hernia just pulling the belt to tighten it. I HATE THIS THING.’	0.009914	50	49.6285	97
5	‘This was a treat to myself and boy does it deliver my place was a haven for dust. it was immense not being held back by a cord and the extra applications are superb. A super clean flat now delivered in Apple style packaging. MUST BUY!’	0.006959	85	44.8305	93

6	‘This product is terrible, not only did the analog stick start jittering after 2 months of usage, it started getting discolored after about 1 months of usage. The build quality is terrible and does not feel like an official PS4 controller and if it is by Sony, then why are they making such bad controllers for so expensive nowadays?. I have no idea why Amazon is even allowing such a terrible product to be sold on their site. I do not recommend buying this, the only time I would recommend this controller is if they offered a warranty. You’re better off buying it from a store where you can actually get a warranty and return it when it decides to break on you, because it will.’	0.012239	17	41.1088	84
7	‘I ordered this after looking for a while for a hand soap with no perfume or fragrance. Itś not listed on the website but it’s listed on the bottle. I hope this wasn’t intentional, it’s really frustrating trying to find a natural hand soap. Claiming it’s natural but putting fragrant chemicals in it is really misleading.’	0.010473	40	49.8882	98
8	‘Shoe says its size 9 wide but its not wide. Been buying this model for years and this is the first time I’ve had an issue. The only difference between these shoes and my others seem to be that these were made in India and the others were made in Vietnam. I’m out of the return window so I’m stuck, my fault, however Rockport quality assurance seems to have slipped at least in my case.’	0.006418	90	42.6048	90

9	‘Love them. So much so that I bought another set (in pink) right after sleeping with this set when they arrived! So happy I finally have these, I’ve heard so many good things/benefits of having satin pillowcases! It also came with washing instructions which I appreciated.’	0.015290	2	20.8472	10
10	‘It stored all my European holiday pictures, but now I cań t transfer them into my computer or look at them for long... the computer recognizes it, but it promptly disappears and I get an ”improperly ejected” message even though its still plugged in!...so FRUSTRATING!!!’	0.004958	99	59.0544	105

Table 4.7: Ranking Detail of Fake Comment Set 1

Obviously, these comments have no relation to our product by our judgment, but more important here is how our algorithms distinguish these comments. Remember that including the fake answers,in each experiment, we have 105 comments in total. We can see that most fake comments have relatively low ranks, which indicate that our methods perform well on this answer set. Take a look at the fake comment #1, which can be regarded as a tough one since it is also a comment from “baby” category. Using the general entropy method, this comment ranks 50th in the list, which is around the median. However, using K-L divergence method, we have a very low rank: 97th. One reason for this situation is that this comment does have some words in common with most other comments. Another reason is that this is a relatively long comment with many different kinds of words, and comments with high complexity like this tend to have high general entropy. However, K-L divergence method focuses on the difference between word distributions. This comment obviously has quite a different word distribution with these original comments, so this comment has a high K-L divergence. The same situation also happens in fake comment #4 and 6. Fake comment #9 is also worth noticing; both methods give this comment pretty high ranks. As we can see, this comment is about a pillowcase, but it does not have any specific words or descriptions regarding the pillowcase. Only based on the words this comment has, this comment can also be used to describe a bib, which is similar to most other relevant comments.

In conclusion, the K-L method is more sensitive to the word distribution while the general entropy put more weight on the text complexity. To illustrate this, we make up a comment that only repeats three words: ‘baby’, ‘bib’ and ‘easy’; remember that these three words are in the top three most common word clusters shown in Table 4.5. Table 4.8 below shows the ranks of this comment using two methods, as we can see, the results are completely different using two methods. This comment only contains three unique words, which implies very low text complexity, thus a low entropy score.

Comment Text	Entropy	E-Rank	K-L	K-Rank
‘baby bib easy baby bib easy baby bib easy baby bib easy baby bib easy baby bib easy’	0.002482	104	17.853996	4

Table 4.8: An Example of Comment with Low Complexity

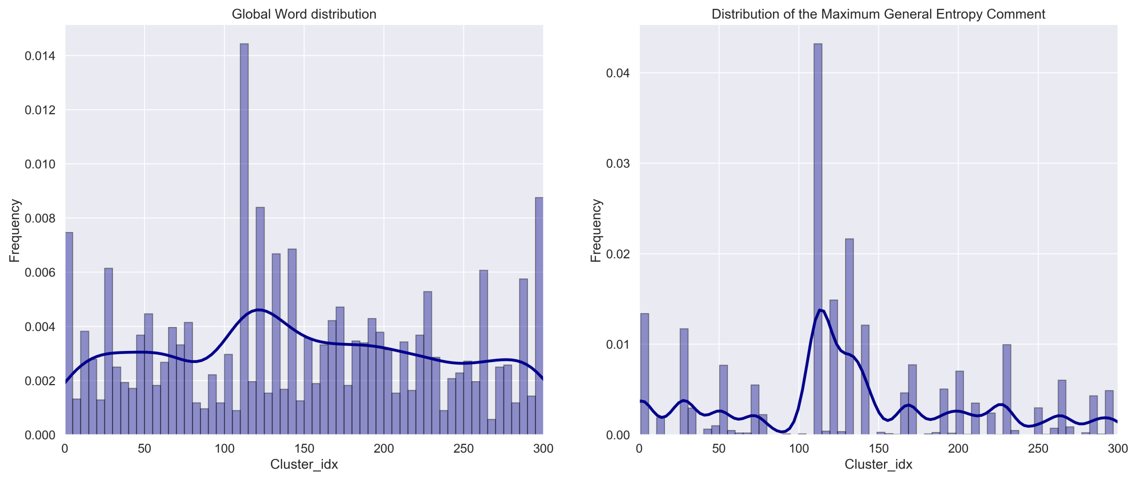


Figure 4.7: Global Distribution and the Maximum General Entropy Comment

Figure 4.7 above shows the comparison between the global distribution and the maximum general entropy comment. We can see that the maximum general entropy comment put more attention on the word clusters that frequently appear in the dataset and less attention on those less frequent clusters. We can call these frequently appeared word clusters as “essential” word clusters. This plot tells us why the K-L method is more sensitive to these non-relevant fake comments than pure general entropy: the K-L method focus on the distribution of few "essential" word clusters. If a comment has none of these "essential" word clusters included, it must result in high K-L divergence and low rank. However, if a comment can precisely contain these "essential" word clusters, it may result in low K-L divergence and high rank even though it has low text quality like the one in Table 4.8.

4.4.3 Relationship Between K-L Divergence and the General Entropy

In the previous section, we analyzed our two ranking methods' performance. Now let us take a look at the relationship between these two ranking metrics. Figure 4.8 below illustrate these two metrics' relationship, note that this plot includes comments from fake comments set #1. In general, the K-L divergence has a negative relationship with the general entropy. Since lower K-L divergence indicates closer distance to the maximum general entropy comment and higher general entropy indicate better text quality with respect to the global distribution, this result corresponds to our statement that two methods are ranking comments based on comments' relevance to this product.

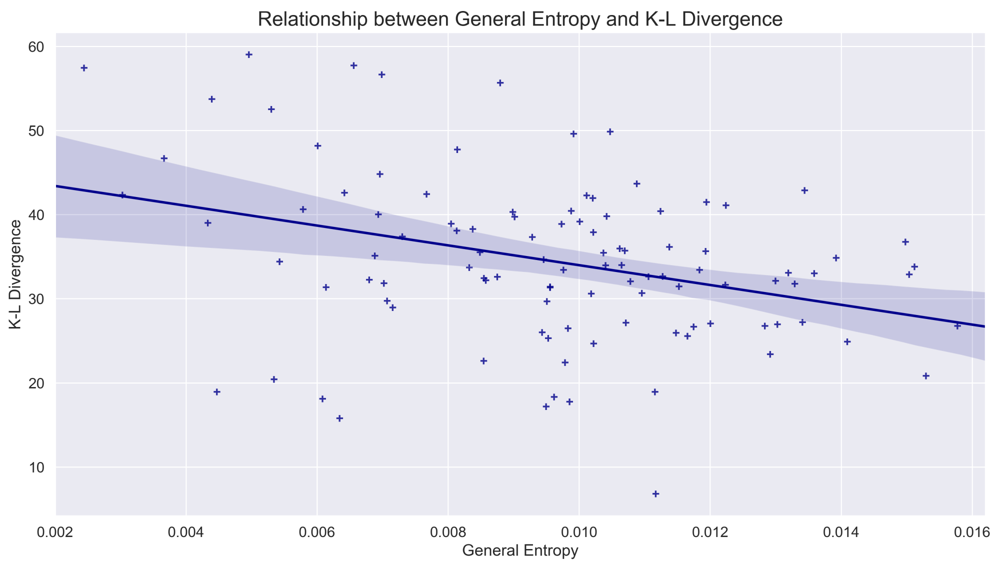


Figure 4.8: Relationship between General Entropy and K-L Divergence

To compare two metrics' distributions, we generate a group of random comments. Each one of these comments is a list of cluster indexes randomly generated from a discrete uniform distribution of [1,300]. Each comment's length also follows a uniform distribution with a mean of 53, which is the same as our dataset. Figure 4.9 shows distributions of our dataset's general entropy and K-L divergence compared with the randomly generated comments.

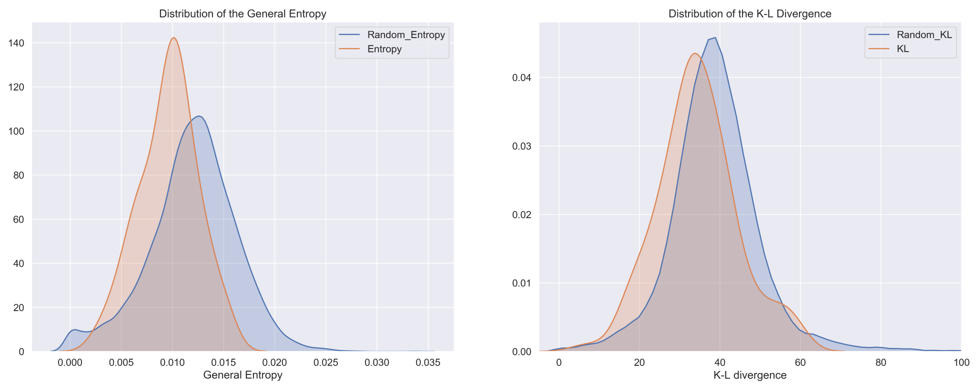


Figure 4.9: Distribution of the general entropy and K-L Divergence

In terms of the variation, our dataset's general entropy distribution is more concentrated than random comments. In contrast, K-L divergence does not have much difference with random comments. As we discussed before, the K-L method is more sensitive to comments' word distribution than the general entropy method, resulting in a higher variation of the scores and better ranking performance.

In terms of the average, we can see that our dataset's general entropy distribution is left-shifted compared to random comments, which means that comments in our dataset generally have lower general entropy than random comments. This is because the general entropy method focuses more on the text complexity of a comment. Remember that these random comments are generated from a uniform distribution, where all word clusters can be selected with equal probabilities. So these random comments tend to have higher text complexity than comments in our dataset. Since the K-L method focuses more on the comments' relevance to the product, its distribution is also left-shifted compared to random comments. Random comments tend to have lower relevance to this product, thus having higher average K-L divergence to the maximum general entropy comment.

Chapter 5

Conclusion

In this thesis, we first proposed a new text representation model: the bag of word clusters model. Unlike the traditional bag of words (BOW) model that treats each word as an independent item, we group semantic-related words as clusters using pre-trained word2vec word embeddings and represent each comment as a distribution of word clusters. This model successfully solves the high dimensions and sparsity problem of the BOW model since the number of clusters is far less than the vocabulary size. A sufficient amount of computational power can be saved by using our text representation method. Another advantage of the bag of word clusters model is that it considers the semantic relationship between words, where words with the same semantic meaning will be treated as the same item in our model. In that way, our model can perform better when making a comparison between two texts.

Next, we proposed our comment ranking algorithm: the K-L divergence to the maximum general entropy comment. We consider the maximum general entropy comment as an "ideal" comment concerning the global word cluster distribution and judge each comment by its distance to this "ideal" comment. The intuition is that the "ideal" comment highlights aspects of a product that many other comments frequently mention. In our experiment with a real Amazon product, this method outperforms the method using the general entropy purely and successfully identifies most of the fake comments. Besides the excellent ranking performance, there are two advantages of our ranking methods: a) it is entirely unsupervised, which requires no prior domain knowledge and no training data. Therefore, this method can be applied to most of the products' comments ranking applications since only the comments texts are needed; b) this method has low computational cost since it only requires statistical information from the text. Training word2vec word embeddings is generally more time consuming, but in fact, many high-quality pre-trained word embeddings can be downloaded from the internet.

There are still many areas regarding our works we can focus on for future research:

1) **Finding Other Clustering Method**: During the research, we are aware that clustering results using k-means are not very stable because our word embeddings have relatively high dimensions. In the future, we may be able to find a more suitable method for clustering word embeddings.

2) **Other Applications of the Bag of Word Clusters Model**: As a text representation method, the bag of word clusters model performs well in our comments ranking application. However, this method can also be used in other text mining applications, such as text classification and sentiment analysis.

3) **Combining the General Entropy Method and the K-L Method**: In Chapter 4, we compare the performances of the general entropy method and the K-L method. We find that the entropy method focuses on the text complexity, while the K-L method focuses more on the distribution difference. If we can combine these two methods, we may be able to get better ranking performance.

4) **Develope a Python Package**: Our algorithm is purely implemented on Python, which can be developed as an open-source Python package.

Bibliography

[1] Oxo tot waterproof silicone roll up bib with comfort-fit fabric neck. https://www.amazon.com/dp/B00D3TPGAO, 2014. Accessed: 2020-03-02.

[2] Online shopping in canada, 2018. https://www150.statcan.gc.ca/n1/pub/89-28-0001/2018001/article/00016-eng.htm, 2019. Accessed: 2020-08-01.

[3] Alberto, T. C., Lochter, J. V., and Almeida, T. A. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (2015), pp. 138–143.

[4] Bird, S., Klein, E., and Loper, E. *Natural Language Processing with Python*, 1st ed. O'Reilly Media, Inc., 2009.

[5] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks 30* (1998), 107–117.

[6] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to rank using gradient descent. 89–96.

[7] Chen, C. C., and Tseng, Y. Quality evaluation of product reviews using an information quality framework. *Decision Support Systems 50*, 4 (2011), 755 – 768.

[8] Chevalier, J. A., and Mayzlin, D. The effect of word of mouth on sales: Online book reviews. *Journal of Marketing Research 43*, 3 (2006), 345–354.

[9] Fellbaum, C. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[10] Goodfellow, I., and Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[11] Hartigan, J. A., and Wong, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[12] He, R., and McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web* (2016), pp. 507–517.

[13] Hsu, C., Khabiri, E., and Caverlee, J. Ranking comments on the social web. *2009 International Conference on Computational Science and Engineering 4* (2009), 90–97.

[14] Järvelin, K., and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst. 20*, 4 (Oct. 2002), 422–446.

[15] Kaufmann, L. Clustering by means of medoids. *Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987* (1987), 405–416.

[16] Manning, C. D., Schütze, H., and Raghavan, P. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[17] McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (2015), pp. 43–52.

[18] Mihalcea, R., and Tarau, P. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (2004), pp. 404–411.

[19] Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. Efficient estimation of word representations in vector space. *CoRR abs/1301.3781* (2013).

[20] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 3111–3119.

[21] NIKFARJAM, A., SARKER, A., OCONNOR, K., GINN, R., AND GONZALEZ, G. Pharmacovigilance from social media: Mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association : JAMIA 22* (03 2015).

[22] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[23] ŘEHŮŘEK, R., AND SOJKA, P. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (Valletta, Malta, May 2010), ELRA, pp. 45–50. http://is.muni.cz/publication/884893/en.

[24] RONG, X. word2vec parameter learning explained. *CoRR abs/1411.2738* (2014).

[25] SALTON, G., AND BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Information Processing Management 24*, 5 (1988), 513 – 523.

[26] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal 27*, 3 (1948), 379–423.

[27] STONE, B., DENNIS, S., AND KWANTES, P. J. Comparing methods for single paragraph similarity analysis. *Topics in Cognitive Science 3*, 1 (2011), 92–122.

[28] WANG, P., XU, B., XU, J., TIAN, G., LIU, C.-L., AND HAO, H. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing 174* (2016), 806 – 814.

[29] Woloszyn, V., dos Santos, H. D. P., Wives, L. K., and Becker, K. Mrr: An unsupervised algorithm to rank reviews by relevance. In *Proceedings of the International Conference on Web Intelligence* (New York, NY, USA, 2017), WI '17, Association for Computing Machinery, p. 877–883.

[30] Woolford, D. G., Cao, J., Dean, C. B., and Martell, D. L. Characterizing temporal changes in forest fire ignitions: looking for climate change signals in a region of the canadian boreal forest. *Environmetrics 21*, 7-8 (2010), 789–800.

[31] Zhang, G. How to rank answers in text mining. *Electronic Thesis and Dissertation Repository. 6250* (2019). PHD Thesis, Western University.

[32] Zhang, Z., and Varadarajan, B. Utility scoring of product reviews. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2006), CIKM '06, Association for Computing Machinery, p. 51–57.

Appendix A

Lists of Fake Comments

asin	Comment Text	Entropy	E-Rank	K-L	K-Rank
B07PDHT5XP	’Currently waiting for two weeks to receive a replacement speaker. Like so many (60-90 in the reviews) people on Amazon who purchased the Bose Soundlink, mine failed to charge or turn on after a few short months. Don’t know why, but I do know that in spite of this extremely common software, Bose won’t send a replacement - you have to send yours in, wait for it to arrive at Bose - they say 10-12 days -and be verified as non functional. Then they will send out a replacement. Don’t know how long that will take. The Bose rep verified that to me on the phone that it would not turn on. He then asked me if I had updated the firmware. This was difficult to do, and the speaker would not work. Oh well. With any luck, the replacement will last as long as the inexpensive ones I’ve purchased. Time will tell!’	0.007413	80	46.0707	97

B07BBKVK8G	'Decent sound for something the size of a hockey puck. Pretty handy i must say. Keep it in the kitchen, good for timers music, conversions, weather and asking general or even area specific. questions etc. I was surprised how well it picks up voice. Wife still basically can't fight the instinct to yell at it like an overseas call. My two year old somehow gets her to play whitney houston consistency even though she is speaking gibberish. I'm starting to think its an actual language that only alexa understands. Some free advice: never ask it to play hide and seek, it will never shut up, be warned. Would reccomend to people who aren't afraid of robots eventually taking over the world. I think it would be a bit overwhelming for my mom's generation.(no offence to you oldr folks). Thumbs up.'	0.006517	89	48.6258	100
B01GY7IKEA	'Fantastic glasses that truly prevent eye strain. As someone with a history of concussions and traumatic brain injury, I am sensitive to screen time headaches. However, since wearing these I have noticed an astronomical deffierence and best of all... no vision change or headaches! My loved ones have already begun to buy pairs for themselves!'	0.007115	83	45.2737	94

B018K1EHFO	’I bought this for my 10 month old son (currently 25 pounds at 12 months) when we went to Washington D.C. and would highly recommend for anyone traveling with a little one. He would get tired of the stroller and would want to be held, there were also times where I wanted him up close with me and not in the stroller. He was very happy and comfortable in this. I have only used the frontal position, facing me so far, I have yet to try any of the other 3 - but its a great product and he loves it. Its also very comfortable with the wide straps and the strap that goes across the lumbar. My husband who is 6’3” 250 pounds and I who am 5’3” 140 pounds, can both wear this - so it also has a wide range as far as who can use.’,	0.011175	32	22.0476	19
B00KWKD64U	’My husband finds these shorts super comfortable. They’re so thin and airy without being see through. He is 6’7” and the tall size fall just at the centre of his knees. The band is very comfortable and the material has great stretch. They wash well without shrinking in the dryer. They look great on him.’	0.010365	45	41.9543	90

B002L3J3H0	'Super fun! I bought this to take a White Elephant gift exchange. Once gifts were opened, we tore open this gift to start playing. So FUN! There are many blank tiles and at first we were bummed but then we made changes to the game. Every time we started a new game, we made new rules for the blank tiles. The ability to make the game as interesting and fun as possible was the best. I had so much fun playing this game, I bought one to keep at my house!'	0.008695	69	32.2230	60
B01LWVX2RG	'Received controller opened box and found a used controller in poor condition, Multiple scratches, curse words carved into the plastic, paint worn away in areas, control knobs sticking due to filth. Seems to be a very used old controller'	0.002048	105	64.8508	105
B01I58TWAW	'The products themselves are lovely (although, Umbra: not thrilled with the logo emblazened on the side of the product), however! one of them arrived with a chip. :(They were packaged very well and obviously new/never opened, and the chip was nowhere to be found within the packaging. Too much of a hassle to return, so I'll keep it.'	0.006462	90	58.4808	104

B06ZXX3LJB	'If you are a wipes snob you will love these! We were using Babies R Us brand before which are like hard wet paper towel... But beware after you try these you will never switch! They are so moist and gentle on your babies skin and you really only need 1 wipe to get the job done. They are like the Viva paper towel of baby wipes!'	0.012192	18	42.749037	92
B00O8PNW6M	'I was introduced to the author in the Norwegian documentary on firewood (National Firewood Night) on netflix - what a fascinating cultural phenomenon! Firewood is a national passion and science, and useful for anyone with a fireplace or wood stove. I read most of it around a campfire during an extended camping trip in New Hampshire, and highlighted and bookmarked so much - It's like a textbook in terms of the validity of the information. I highly recommend this read even if purely for pleasure.'	0.005926	95	46.8511	98

Table A.1: Detail of Fake Comment Set 2

asin	Comment Text	Entropy	E-Rank	K-L	K-Rank
B001XSFW42	This is a very poorly designed pizza cutter. The grey surface which tries to mimic a circular saw deck just scrapes the cheese from any pizza regardless of the thickness. There is a ridiculous blade guard, again designed to look like a circular saw blade guard which just gums up, and partially falls out of place while you're using the cutter.Waste of money. Selling ASAP. There are far better pizza cutters than this!Boooo!'	0.011898	22	38.8387	75
B00VND51XE	'In all fairness it seems like the scent was changed 5 years ago. I used Dreft when my son was a baby, and found the fragrance heavenly. I then wanted to reintroduce Dreft into our family, and was extremely disappointed to find out that the scent is now different. Like others said it smells way too strong and perfumy. Why ruin such an amazing product?!'	0.004810	98	48.0275	97

B01IVTVK3W	’I decided to buy this on prime day since the price was reduced and I’m so glad I did. I was worried because I have a large 2 month old son and if he couldn’t fit, I would have no use for it. Not only is it big enough for him, but my 2 year old also loves it. Obviously she’s too big to lay down but she fits comfortably sitting, even with my son laying down next to her. It’s also relatively lightweight and folds very easily. I plan on sticking it in the car seat bag when we travel via airplane, I think it will fit easily when folded but we’ll see. We spend a lot of time outdoors so this way I wont have to hold my son outside, it gets very hot in the summertime so this way he can stay cooler, especially with the canopy.’	0.011028	34	24.8252	11

B00Z9QVE4Q	'The power bank needs more than 5 hours to charge from 50% status (which is the status when I received the item). I directly wrote to Anker when after 4 hours or so it is still not fully charged, suspecting that the item is defective. ANKER responded within hours with infosuggestion to use the original USB cable that came together with the item. It seems that my usual long USB cable is not suitable for this power bank even though it is working fine wirth my other power bank. Please note that to get a quicker charge this Anker power bank need that short USB cable, do not use your own cable, it will take longer to charge. Good lessons learned. It is working fine and charge well for the last 10 days.'	0.006185	88	40.9257	82
B01D1HEP0E	This jasmine candle is very fragrant , it's a soy candle and burns down leaving no residue on the inside of the opal container which has a lovely lid when not in use .'	0.005992	91	45.6706	94
B00UTO8YKU	'Switched from a Bodum brand travel press to this. There is no comparison between the two. The filter on this one works far better and the coffee is much cleaner tasting. It is well made, stylish and not a speck of grit gets into the coffee. An excellent purchase.'	0.004587	99	59.2816	102

B012A4IXES	’Have been using these wipes since my son was born. This is what the hospital used and I’ve been using it since. They are soft, fragrance free and strong. They don’t rip very easily unlike the other competitors. I was given a sample of huggie wipes and these are far superior. My son doesn’t have sensitive skin but I wouldn’t purchase any other type. I even use them to wipe his hands when soap and water aren’t in reach. Great value and fast shipping, thanks Amazon.’	0.014022	7	42.5158	86
B00OK3TXJC	’Never owned a Pneumatic drill before. While building planters & deck, worked great for 1/8” holes back to back... but didn’t seem to have the sustainable power to do the 1” hole in a 2x8. Pulled out my corded Dewalt and went through in about 20 seconds.’	0.005774	94	60.6564	103
B01LNSAYJ4	’This truly large lens is very sharp, even at 600mm with a full frame sensor. It’s a bit tricky to learn to use the lens because of the lens aperture, which requires a lot of light, or a high ISO and a low shutter speed. Highly recommended to use with a tripod for lower ISO shooting of stationary or slow-moving subjects. Not for indoor or night sports. Highly recommended.’	0.007040	80	34.2202	48

1419717987	’I contemplated buying this book a year ago to see what all the ’Non-Fiction book of the year’ fuss was about but thought it’s got to be way too boring, like seriously, wood chopping, drying and stacking. Got it for Christmas and I was very wrong, can’t put it down, takes you to a place of relaxation, now where’s my firelighters and matches.’	0.005244	97	51.5619	99

Table A.2: Detail of Fake Comment Set 3

asin	Comment Text	Entropy	E-Rank	K-L	K-Rank
B07P68CZ5D	'A good quick and easy gift idea for ANYONE! It took less than 3 minutes for me to purchase and have the gift card emailed to me which is great whenever you're in a pinch to find a last minute gift. Love that there are so many options...amazon will create a printable version for you or you can choose a date and time to have it emailed directly to the recipient and of course if you have time you can order a phsyical gift card.'	0.010153	49	45.2188	85
07STGGQ18	'First CPU I've ever seen in 25 years of building my own PC's that was dead on arrival.Tested on 3 different MB's (including X570 Board), 2 known working sets of Ram, 2 known working PSUś, 2 known working Graphics Cards.Refused to boot in all cases. Sent it this back and grabbed a 2600 instead and it worked fine.'	0.009022	59	58.0232	101
B07FK9KVPM	'I would have given these awesome shoes 5 stars other than they are way TOO SMALL. I wear a 39 but returned that size. This pair is 40/41 and barely fits my 8.5 foot. But they're gorgeous with a nice rubber tread and seem well constructed. Given that Ottawa is plowing through snowstorm after snowstorm I have yet to try them on the beach BUT trust me..I will!'	0.010875	35	37.8595	53

B0876XJ6CX	’Much like yourself, I enjoy having eyes. I also prefer when they aren’t burning from the heat of the sun radiating straight into my skull, or from the light reflecting from the snow. This is why I bought these polarized aviators. I’m a man of class and style, which is why I try and look like I travel the world (I don’t leave my apartment) by wearing aviators. These help me see straight into the depths of the river when I’m fishing, and they have helped me reel in more behemoths than Jeremy Wade (y’know, the old white dude from River Monsters). They are also quite large, which help hide the bags under my eyes while I’m at work. No, Susan, I’m not hungover - I was up all night crying because my wife left me and took the kids. All in all, I would recommend these sunglasses if you have eyes and don’t want glaucoma when you’re 34.’,	0.008559	73	53.7139	98
B00DI1H614	’This thing is made to be impossible to repair. I’m a electronic technician and after trying to repair a defective display (almost gone), Iv́e notice that most of the screws are rusted. Also, they made it to be impossible to repair without breaking the enclosure. Oster, never again.’	0.004458	101	55.3778	100

B086YHGDNC	'Exactly as shown in photos, arrived early! It's weird that I have a darker red band that is between the black and purple. The bands weren't too stiff upon arrival but still need some "breaking in". I use the tiny red for everyday stretching.'	0.010596	40	42.5785	78
B07XR5XMJX	'One of the best toothbrushes I've had! It's a lot less clunky than other electric toothbrushes but it still has great battery life. I haven't had to charge it since I first got it. I really appreciate the different modes, which allow me to you less power on parts of my mouth that are more sensitive. Definitely a great buy for the price.'	0.007699	82	47.1460	87
B01M6C5SMP	'I have newborn twin girls so I cannot even begin to express how much easier these pyjamas have made my life! First off the zip is a godsend... no really! When you are struggling through the newborn phase, pulling a zipper open and closed to get your little one ready is really convenient and above all quick! No more struggling with poppers in the dark with a fussy baby. Times that by two and well.. there you have it lolSecond the material is really warm so I no longer have to layer my girls' clothes as much to keep them nice and cosy during these winter months.'	0.011366	27	42.2115	76

B00N1X6JV2	'Saw these pens in a video and being the stationery addict i am, i had to buy! They are honestly the nicest pens ive used...ink dries quick and doesnt smear with the zebra mildliners, which was the main concern as my muji gel pens smear every where when highlighted. The barrel is slightly uncomfortable to hold, as it is one width all the way, so i would recommend putting the ink in a zebra sarasa 0.5mm barrel, as ive seen many people do. definitely makes a difference'	0.010508	41	49.2298	91
B07MLWYXJM	'This knife comes with a beautiful gift box, is very nice making the knife a great gift if you choose. Knife is super sharp right out of the box .It also come with a knife sharpener so nice.The handle and blade look great, and it feels great in the hand. It has light weight that mean your hand is tireless when you prepare the dishes for your family. We've used it on veggies and meats so far and have no complaints. This knife will been great tool in my kitchen. I love it so much.'	0.008556	74	47.334071	88

Table A.3: Detail of Fake Comment Set 4

asin	Comment Text	Entropy	E-Rank	K-L	K-Rank
B073SNVD13	’this machine has been worth its weight in gold. it has been that good!! I believe I’m going on 3 years and I use it a minimum of at least once a week. Tip: press air out at the beginning of sealing feature it seems to help prime the vacuum if even possible, plus lift package up slightly as its vacuuming as it is being ’suck’ towards the machine.’	0.014773	5	27.3552	12
B01N1UX8RW	’It’s really good and pretty accurate, however please note that this does not take it your overall body fat percentage, only for your lower half. To take in overall body fat, you’ll need a scale with hand sensors for the upper body.However still a good gadget to keep track of progress.’	0.009084	67	45.4925	89
B00XISYB42	’I bought this particular product solely based off product reviews, only to find most of the ones on amazon suspiciously posted by one time posters or people only pumping up EVL products. I would stay away from EVL solely due to the shady marketing practices and stick to better known brands such as ON’	0.009494	62	35.4899	49
B082WRL64V	’Ordered these for my hubby and they fit him perfectly. He wears slippers inside and out on our porch and patio. So I put them by the door and He said they are super comfortable . They also have a harder bottom so that works nicely for him when he steps outside . Easy slip on and off .’,	0.007007	86	30.3250	25

B00NLZUM36	'Bought this thing ages ago and had a ton of issues but never bothered to review it. But then I bought another product from these people and had even MORE issues that they ignored me about until I left them a bad review and they promised me a replacement that they never sent. So i decided to go back and review this one too. Its super cheap, the LED's in the keyboard NEVER worked, the mouse scroll wheel stopped working almost immediately and the left click is sporadic at best and clearly as mentioned above the customer service is AWFUL.'	0.007411	82	56.3201	101
B07SW1LSRG	'Absolutely terrible. Items do not transfer when you reach for them, doors do not open or close, zombies kill you from a distance by some type of teleportation while the screen goes completely black???? What? How is this a full release? Wow. Just don't spend your hard earned money on this. I am willing to suffer some bugs, but this is honestly completely unplayable.'	0.010067	49	37.9314	60

B006PDJZ48	'This is a great product. I misplaced the original one I purchased and I recently purchased two new ones. I keep one at my office desk and one at my favourite TV viewing chair at home. My only issue with the chest expander is that I would like my wife to be able to use it but at her current strength she would need to have a set of 3 20 pound resistance bands or perhaps even 3 10 pound ones. There does not appear to be any way to purchase such lower strength resistance bands for the chest expander.'	0.008228	79	70.1274	105
B085QC2Q32	'The overall quality is okay but it is advertised as being stainless steel/ hypoallergenic but I do not think this is the case. I could only stand the pain for a few hours before taking it off completely. Also, the earrings are waaay to big for the cartilage - even the smallest size.'	0.006817	90	44.6643	86
B075JS5RHD	'Great photo and video quality. The video output was disappointing though, HLG 10-bit is not supported yet so you have no "effective" way of getting 10-bit "out of the box". Then you also need to do some post processing to get the right color grading. In other words don't expect to shoot your kids at the park and then watch them in beautiful vivid color on your NVidia Shield, it's just not designed for that use-case.'	0.011397	26	35.1542	47

B07JWJTD1Y	’This product is very nice. This is the second Zilla product I have bought and I have been pleased with them both. The pump works well and is very quiet. The small plants are a nice touch too. The even include a nice thick plastic feeding dish. I am not using it, but it is nice to have. It hold around 48 oz. of water and helps keep the humidity up in my sons crested gecko enclosure. It matches the zilla rock hide I have. I highly recommend this.’	0.010994	35	38.9952	66

Table A.4: Detail of Fake Comment Set 5

Appendix B

Python Code

B.1 Data Cleaning

```
import pickle
import pandas as pd
from nltk.stem import WordNetLemmatizer
import gensim.parsing.preprocessing as preprocess
from gensim.utils import simple_preprocess
from gensim.models.phrases import Phrases, Phraser
from time import time  # To time our operations

def parse(path):
        g = open(path, 'rb')
        for line in g:
                yield eval(line)

def getDF(path):
        i = 0
        df = {}
        for d in parse(path):
```

```python
                df[i] = d
                i += 1
        return pd.DataFrame.from_dict(df, orient='index')


def lemmatize(sentence, lemmatizer):
        """
        @param: sentence(string);
        @param: lemmatizer(function)
        return: list of str
        """
        sentence = simple_preprocess(sentence, min_len=3)
        sentence = [lemmatizer.lemmatize(w) for w in sentence]
        return sentence


def clean_data(text, lemmatizer):
        """
        @param: text(String),
        @param: lemmatizer(function)
        return: corpus(list)
        """
        res = []
        for sentence in text:
                sentence = preprocess.remove_stopwords(sentence.lower())
                sentence = preprocess.strip_non_alphanum
                   (sentence)
                # remove punctuation
                sentence = preprocess.strip_numeric(
                   sentence)
                # remove number
                sentence = lemmatize(sentence,
                   lemmatizer)
                # lemmatize data
                if len(sentence) > 2:
                        res.append(sentence)

        return res


if __name__ == "__main__":
        T = time()
        text = ""
        true, false = 1, 0
        df = getDF('reviews_Baby_5.json')
        for t in df['reviewText']:
                if not type(t) == str:
                        continue
                text = text + t
        # every line represent one sentence
        text = text.split(".")  # split data by sentence
        lemmatizer = WordNetLemmatizer()
        text = clean_data(text, lemmatizer)
        # automatically detect common phrases (bigrams)
           from a list of sentences
        phrases = Phrases(text, min_count=30, progress_per=10000)
        bigram = Phraser(phrases)
        text = bigram[text]
        # save bigram model
        bigram.save(".../my_bigram_model.pkl")
        # write text into our file
        with open(".../train.txt", "wb") as f:
                pickle.dump(text, f)
                f.close()
        print('Time\␣to\␣clean\␣the\␣data:\␣{}␣mins'.format(round((time() - T) / 60, 2)))
```

B.2 word2vec model

```
import pickle
from time import time  # To time our operations
import multiprocessing
from gensim.models import Word2Vec
import logging  # Setting up the loggings to monitor gensim
logging.basicConfig(format="%(levelname)s␣-␣%(asctime)s:␣%(message)s", datefmt='%H:%M:%S', level=logging.INFO)

# Read data
with open("train.txt", "rb") as f:   # Unpickling
        train = pickle.load(f)

```

```
cores = multiprocessing.cpu_count()  # Count the number of cores in a computer

w2v_model = Word2Vec(min_count=30,
        window=2,
        sg=1,
        size=150,
        sample=0,
        alpha=0.03,
        min_alpha=0.0007,
        negative=20,
        workers=cores-1)

# Build Vocabulary
t = time()
w2v_model.build_vocab(train, progress_per=10000)
print('Time␣to␣build␣vocab:␣{}␣mins'.format(round((time() - t) / 60, 2)))
# Train
t = time()
w2v_model.train(train, total_examples=w2v_model.corpus_count, epochs=30, report_delay=1)
print('Time␣to␣train␣the␣model:␣{}␣mins'.format(round((time() - t) / 60, 2)))

# Save model
w2v_model.save("word2vec.model")
```

B.3 word embedding clustering

```
import pickle
from gensim.models import Word2Vec
from time import time
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# load model
w2v_model = Word2Vec.load("word2vec.model")

# normalize vectors in the model
w2v_model.init_sims(replace=True)

# Build training data
X = w2v_model.wv.vectors

# initialize k-means model
kmeans = KMeans(n_clusters=50, n_init=100, random_state
    =777, n_jobs=-1)


# fit k-means model
t = time()
kmeans.fit(X)
labels = kmeans.labels_
res = silhouette_score(X, labels)
print('Time␣to␣train␣the␣model:␣{}␣mins'.format(round((
    time() - t) / 60, 2)))

```

```python
# build map: word->cluster label
wordlist = w2v_model.wv.index2word
word2cluster = {wordlist[i]: kmeans.labels_[i] for i in
    range(len(wordlist))}
# build map: cluster->word cluster[list]
cluster2word = {}
for i in range(kmeans.n_clusters):
        group = [key for key in word2cluster if
            word2cluster[key]==i]
        cluster2word[i] = group

with open("word2cluster", "wb") as f:
        pickle.dump(word2cluster, f)
        f.close()
```

B.4 Experiment with an Amazon Product

The experiment in this thesis is implemented using a Jupyter Notebook. You can download this notebook using this link.

Curriculum Vitae

Name:	Yuyang Zhang
Post-Secondary Education and Degrees:	Shandong University Jinan, China 2014 - 2018 B.Sc.
	University of Western Ontario London, ON 2018 - Present M.Sc.
Honours and Awards:	Western Graduate Research Scholarship 2018-2020
Related Work Experience:	Teaching Assistant University of Western Ontario 2018 - 2020