# LEARNING TOPICS USING SEMANTIC LOCALITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The topic modeling discovers the latent topic probability of given the text documents. To generate the more meaningful topic that better represents the given document, we proposed a universal method which can be used in the data preprocessing stage. The method consists of three steps. First, it generates the word/word-pair from every single document. Second, it applies a two way parallel TF-IDF algorithm to word/word-pair for semantic filtering. Third, it uses the k-means algorithm to merge the word pairs that have the similar semantic meaning.

Experiments are carried out on the Open Movie Database (OMDb), Reuters Dataset and 20NewsGroup Dataset and use the mean Average Precision score as the evaluation metric. Comparing our results with other state-of-the-art topic models, such as Latent Dirichlet allocation and traditional Restricted Boltzmann Machines. Our proposed data preprocessing can improve the generated topic accuracy by up to 12.99%. How the number of clusters and the number of word pairs should be adjusted for different type of text document is also discussed.

## 1 INTRODUCTION

After millennium, most collective information are digitized to form an immense database distributed across the Internet. Among all, text-based knowledge is dominant because of its vast availability and numerous forms of existence. For example, news, articles, or even Twitter posts are various kinds of text documents. For the human, it is difficult to locate one's searching target in the sea of countless texts without a well-defined computational model to organize the information. On the other hand, in this big data era, the e-commerce industry takes huge advantages of machine learning techniques to discover customers' preference. For example, notifying a customer of the release of "Star Wars: The Last Jedi" if he/she has ever purchased the tickets for "Star Trek Beyond"; recommending a reader "A Brief History of Time" from Stephen Hawking in case there is a "Relativity: The Special and General Theory" from Albert Einstein in the shopping cart on Amazon. The content based recommendation is achieved by analyzing the theme of the items extracted from its text description.

Topic modeling is a collection of algorithms that aim to discover and annotate large archives of documents with thematic information Blei (2012). Usually, general topic modeling algorithms do not require any prior annotations or labeling of the document while the abstraction is the output of the algorithms. Topic modeling enables us to convert a collection of large documents into a set of topic vectors. Each entry in this concise representation is a probability of the latent topic distribution. By comparing the topic distributions, we can easily calculate the similarity between two different documents. Steyvers & Griffiths (2007)

Some topic modeling algorithms are highly frequently used in text-mining Mei et al. (2008), preference recommendation Wang & Blei (2011) and computer vision Wang & Grimson (2008). Blei (2012) Many of the traditional topic models focus on latent semantic analysis with unsupervised learning. Latent Semantic Indexing (LSI) Landauer (2006) applies Singular-Value Decomposition (SVD) Golub & Reinsch (1970) to transform the term-document matrix to a lower dimension where semantically similar terms are merged. It can be used to report the semantic distance between two documents, however, it does not explicitly provide the topic information. The Probabilistic Latent Semantic Analysis (PLSA)Hofmann (1999) model uses maximum likelihood estimation to extract latent topics and topic word distribution, while the Latent Dirichlet Allocation (LDA) Blei et al. (2003) model performs iterative sampling and characterization to search for the same information.

The availability of many manually categorized online documents, such as Internet Movie Database (IMDb) movie review Inc. (1990), Wikipedia articles, makes the training and testing of topics models possible. All of the existing workds are based on the bag-of-words model, where a document is considered as a collection of words. The semantic information of words and interaction among objects are assumed to be unknown during the model construction. Such simple representation can be improved by recent research advances in natural language processing and word embedding. In this paper, we will explore the existing knowledge and build a topic model using explicit semantic analysis.

The work studies the best data processing and feature extraction algorithms for topic modeling and information retrieval. We investigate how the available semantic knowledge, which can be obtained from language analysis or from existing dictionary such as WordNet, can assist in the topic modeling.

Our main contributions are:

- We redesign a new topic model which combines two types of text features to be the model input.
- We apply the numerical statistic algorithm to determine the key elements for each document dynamically.
- We apply a vector quantization method to merge and filter text unit based on the semantic meaning.
- We significantly improve the accuracy of the prediction using our proposed model.

The rest of the paper is structured as follows: In Section 2, we review the existing methods, from which we got the inspirations. This is followed in Section 3 by details about our topic models. Section 4 describes our experimental steps and evaluate the results. Finally, Section 5 concludes this work.

## 2 Related Work

Many topic models have been proposed in the past decades. This includes LDA, Latent Semantic Analysis(LSA), word2vec, and Restricted Boltzmann Machine (RBM), etc. In this section, we will compare the pros and cons of these topic models.

LDA was one of the most widely used topic models. LDA introduces sparse Dirichlet prior distributions over document-topic and topic-word distributions, encoding the intuition that documents cover a small number of topics and that topics often use a small number of words. Blei et al. (2003) LSA was another topic modeling technique which is frequently used in information retrieval. LSA learned latent topics by performing a matrix decomposition (SVD) on the term-document matrix. Dumais (2004) In practice, training the LSA model is faster than training the LDA model, but the LDA model is more accurate than the LSA model.

Traditional topic models did not consider the semantic meaning of each word and cannot represent the relationship between different words. Word2vec can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. Mikolov et al. (2013a) During the training, the model generated word-context pairs by applying a sliding window to scan through a text corpus. Then the word2vec model trained word embeddings using word-context pairs by using the continuous bag of words (CBOW) model and the skip-gram model. Mikolov et al. (2013b) The generated word vectors can be summed together to form a semantically meaningful combination of both words.

Moreover, there were a lot of extensions of the word2vec model, such as paragraph2vec and LDA2vec. Paragraph2vec was an unsupervised framework that learned continuous distributed vector representations for pieces of texts. The training of the paragraph2vec model is based on the similar idea as the word2vec. One of the advantages of the generated paragraph vector was that they take into consideration the word order. Le & Mikolov (2014) LDA2vec focused on utilizing document-wide feature vectors while simultaneously learning continuous document weights loading onto topic vectors. LDA2vec embedded both words and document vectors into the same space and train both representations simultaneously. Moody (2016)

RBM was proposed to extract low-dimensional latent semantic representations from a large collection of documents Hinton & Salakhutdinov (2009). The architecture of the RBMs is an undirected bipartite graphic, in which word-count vectors were modeled as Softmax input units and the output units were usually binary units. The RBM model can be generalized much better than LDA in terms of both log-probability on the document and the retrieval accuracy. A deeper structure of neural network was developed based on stacked RBMs, which was the Deep Belief Network (DBN). In Hinton & Salakhutdinov (2011), the input layer was the same as RBM mentioned above, other layers are all binary units.

# 3 APPROACH

## 3.1 WORD PAIR BASED RBM MODEL

Current RBM model for topic modeling uses Bag of Words approach. Each visible neuron represents the number of appearance of a dictionary word. We believe that the order of the words also exhibits rich information, which is captured by the bag of words approach. Our hypothesis is that including word pairs (with specific dependencies) helps to improve topic modeling.

In this work, Stanford natural language parser Chen & Manning (2014) Nivre et al. (2016) is used to analyze sentences in both training and testing texts, and extract word pairs. For example, if we have the following sentence, by applying the word pair extraction, we will get 38 word pairs and one root pair as the Figure 1.

*The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.*

- det(rain-3, The-1)
- amod(rain-3, strongest-2)
- nsubj(shut-8, rain-3)
- nsubj(snapped-16, rain-3)
- nsubj(closed-20, rain-3)
- nsubj(forced-23, rain-3)
- advmod(recorded-5, ever-4)
- partmod(rain-3, recorded-5)
- prep_in(recorded-5, India-7)
- ccomp(said-40, shut-8)
- prt(shut-8, down-9)
- det(hub-12, the-10)
- amod(hub-12, financial-11)
- dobj(shut-8, hub-12)
- prep_of(hub-12, Mumbai-14)
- conj_and(shut-8, snapped-16)
- ccomp(said-40, snapped-16)
- nn(lines-18, communication-17)
- dobj(snapped-16, lines-18)
- conj_and(shut-8, closed-20)
- ccomp(said-40, closed-20)
- dobj(closed-20, airports-21)
- conj_and(shut-8, forced-23)
- ccomp(said-40, forced-23)
- dobj(forced-23, thousands-24)
- prep_of(thousands-24, people-26)
- aux(sleep-28, to-27)
- xcomp(forced-23, sleep-28)
- poss(offices-31, their-30)
- prep_in(sleep-28, offices-31)
- xcomp(forced-23, walk-33)
- conj_or(sleep-28, walk-33)
- dobj(walk-33, home-34)
- det(night-37, the-36)
- prep_during(walk-33, night-37)
- nsubj(said-40, officials-39)
- root(ROOT-0, said-40)
- tmod(said-40, today-41)

Figure 1: Word Pair Extraction

The first part in each word pair segment represents the relationship between two words, such as det, nsubj, advmod, etc. And the number after each word gives its position of in the original sentence. The two words extracted in this way are not necessarily adjacent to each other, however, they are semantically related.

Because each single word may have different combinations with other words, the total number of the word pairs will be much larger than the number of word in the training dataset. If we use all word pairs which are extracted from the training dataset, it will significantly increase the size of our dictionary and reduce the performance. So, we only keep the first 10000 most frequent word pairs to be our word pair dictionary.

### 3.2 TWO STEPS TF-IDF PROCESSING

TF-IDF stands for the term frequency - inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Sparck Jones (1972) Salton et al. (1983) Salton & McGill (1986) Salton & Buckley (1988) Wu et al. (2008) The equation of the TF-IDF is as following:

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ docuemnt}{Total\ number\ of\ terms\ in\ the\ document} \tag{1}$$

$$IDF(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ docuemnts\ with\ term\ t\ in\ it} \tag{2}$$

$$TF - IDF(t) = TF(t) * IDF(t) \tag{3}$$

Term Frequency (TF), Equation 1, measures how frequently a term occurs in a document. Inverse document frequency (IDF), Equation 2, measures how important a term is.

There are many words in the training dataset. If we use all words to build the training dictionary, it will contain a lot of high frequency but useless words, like “first” and “name”. Beside removing stop word and processing word tense in the dataset, we also used the TF-IDF algorithm to filter the dataset.

In addition, based on the original TF-IDF algorithm, we proposed a two-step TF-IDF processing method. This method is applied to keep the number of word pairs to a manageable size as shown in Figure 2:

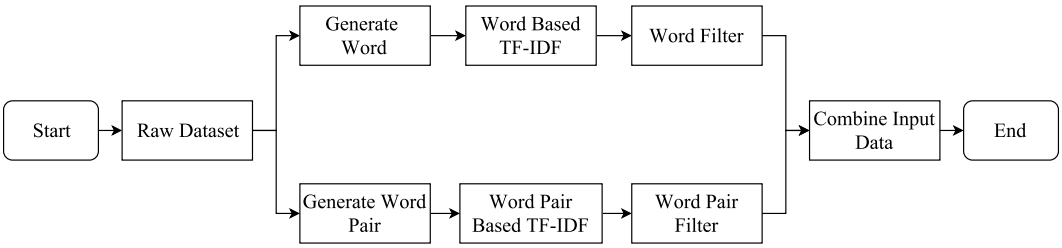



Figure 2: Two Steps TF-IDF Flow Chart

First, word pairs are generated and word-level TF-IDF is performed. The result of word level TF-IDF is used as a filter and a word pair is kept only if the TF-IDF scores of both words are higher than the threshold (0.01). After that, we treat each word pair as a single unit, and the TF-IDF algorithm is applied to the word pairs and further filter out word pairs that are either too common or too rare.

### 3.3 K-MEANS CLUSTERING

Even with the TF-IDF processing, the size of the word pair dictionary is still prohibitively large and selecting only the most frequent ones is a brute force approach. We further cluster semantically close word pairs to reduce the dictionary size. The semantic distance between two words is measured as the distance of their embedding vectors calculated using Googles word2vec model. The only issue is how to determine the cluster centrum.

Our first approach is to use upper level words in WordNet Miller et al. (1990) Miller (1995) as cluster centrum. Our hypothesis is that, since those upper-level words have broader meaning, a relatively small number of these words can provide good coverage of the semantical space. We first build a word level tree based on the relations specified in the WordNet. We then pick those words that are closest to the root as the cluster centrum. Words usually have multiple paths to the root. We first add those synsets that have only one path to the root into the graph, then iteratively examine the concepts

that we have left and add the paths that grow the tree by as little as possible. Based on which path is selected, this approach can be further divided into min-path clustering and max-path clustering. The min-path clustering will add the shortest path and the max-path clustering will add the longest path to the graph

The second approach is K-means clustering. By applying the K-mean algorithm, we group the embedding vector of words into K clusters. Then we use the index of each cluster to represent a group of words. Because the word embedding is a very dense space, the k-mean clustering does not give good Silhouette score Rousseeuw (1987). Our original hypothesis is that the K-mean clustering based approach will not be as effective as the WordNet-based approach. However, our experimental results show that this is not right. So, we pick the K-means clustering algorithm to be our final feature dictionary organization method.

# 4 EVALUATION

## 4.1 EXPERIMENTS SETUP

In our experiment, we generate the topic distribution for each document by using RBM model. Then we retrieve the top N documents by calculating Euclidean distance. Our proposed method is evaluated on 3 datasets: OMDb, Reuters, and 20NewsGroup. For the Reuters and 20NewsGroup dataset, we download them from Cachopo (2007). The OMDb dataset is collected manually by using OMDb APIs (Fritz). All the datasets are divided into three sub-dataset: training, validation, and testing. The split ratio is 70:10:20. For each dataset, a 5-fold cross-validation is applied.

- **OMDb** stands for The Open Movie Database.The training dataset contains 6043 movie descriptions, the validation dataset contains 863 movie descriptions and the testing dataset contains 1727 movie descriptions. We define the class for each movie by applying K-means clustering on category information. The value of K is set to 20.
- **Reuters**, these documents appeared on the Reuters newswire in 1987 and were manually classified by personnel from Reuters Ltd. There are 7674 documents in total which belongs to 8 classes. The training dataset contains 5485 news, the validation dataset contains 768 news and the testing dataset contains 1535 news.
- **20NewsGroup**, this dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The training dataset contains 13174 news, the validation dataset contains 1882 news and the testing dataset contains 3765 news.

## 4.2 METRIC

We use $mean\ Average\ Precision\ (mAP)$ score to evaluate our proposed method. It is a score to evaluate the information retrieval quality. This evaluation method considers the effect of order in the information retrieval result. If the relational result is shown in the front position, the score will tend to 1; if the relational result is shown in the back position, the score will tend to 0. $mAP1$, $mAP3$, $mAP5$, and $mAP10$ are used to evaluate the retrieval performance. For each document, we retrieve 1, 3, 5, and 10 documents whose topic vectors have the smallest Euclidean distance with that of the query document. The documents are considered as relevant if they share the same class label. Before we calculate the $mAP$, we need to calculate the $Average\ Precision\ (AveP)$ for each document first. The equation of $AveP$ is described below.

$$AveP = \frac{\sum_{k=1}^{n}(P(k) \cdot rel(k))}{\text{number of relevant documents}} \tag{4}$$

where $rel(k)$ is an indicator function equaling 1 if the item at rank $k$ is a relevant document, 0 otherwise.Turpin & Scholer (2006) Note that the average is over all relevant documents and the relevant documents not retrieved get a precision score of zero.

The equation of the $mean\ Average\ Precision\ (mAP)$ score is as following:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{5}$$

where $Q$ indicates the total number of queries.

### 4.3 RESULTS

#### 4.3.1 WORD/WORD PAIR PERFORMANCE COMPARISON

In this experiment, the total feature size is a fixed number. Then, we compare the performance between two RBM models. One of them only considers words as the input feature, while the other has combined words and word pairs as the input feature. The total feature size varies from 10500, 11000, 11500, 12000, 12500, 15000. For the word/word pair combined RBM model, the number of word is 10000, and the number of word pairs is varied to meet the total feature requirement.

Both models are applied to the OMDb dataset, and the results are shown in Figure 3 and Table 1, the word/word pair combined model almost always performs better than the word-only model. For the $mAP1$, the $mAP5$ and the $mAP10$, the most significant improvement shown in Feature Number = 11000, about 10.48%, 7.97% and 9.83%. For the $mAP3$, the most significant improvement shown in Feature Number = 12000, about 9.35%.

Table 1: OMDb fix total feature number word/word pair performance evaluation

| **mAP** | **F = 10.5K** | | **F = 11K** | | **F = 11.5K** | | **F = 12K** | | **F = 12.5K** | | **F = 15K** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair |
| mAP 1 | **0.14772** | 0.14603 | 0.13281 | **0.14673** | 0.13817 | **0.14789** | 0.13860 | **0.14754** | 0.14019 | **0.14870** | 0.13686 | **0.14708** |
| mAP 3 | 0.09381 | **0.09465** | 0.08606 | **0.09327** | 0.08933 | **0.09507** | 0.08703 | **0.09517** | 0.09054 | **0.09657** | 0.09009 | **0.09537** |
| mAP 5 | 0.07453 | **0.07457** | 0.06835 | **0.07380** | 0.07089 | **0.07508** | 0.06925 | **0.07485** | 0.07117 | **0.07635** | 0.07175 | **0.07511** |
| mAP 10 | 0.05273 | **0.05387** | 0.04862 | **0.05340** | 0.04976 | **0.05389** | 0.04900 | **0.05322** | 0.05019 | **0.05501** | 0.05083 | **0.05388** |

The two models are further applied on the Reuters dataset, and the results are shown in Figure 4 and Table 2. Again the word/word pair combined model outperforms the word-only model almost all the time. For the $mAP1$, the most significant improvement happens when Feature Number = 12500. Under this feature size, the combined model improves the $mAP$ score by approximately 1.05%. For the $mAP3$, the $mAP5$ and the $mAP10$, the most significant improvement happens when Feature Number = 15000. Under this feature size, the combined model gives about 1.11%, 1.02% and 0.89% improvement.

Table 2: Reuters fix total feature number word/word pair performance evaluation

| **mAP** | **F = 10.5K** | | **F = 11K** | | **F = 11.5K** | | **F = 12K** | | **F = 12.5K** | | **F = 15K** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair |
| mAP 1 | 0.94195 | **0.95127** | 0.94277 | **0.95023** | 0.94407 | **0.95179** | 0.94244 | **0.94997** | 0.94277 | **0.95270** | 0.94163 | **0.94984** |
| mAP 3 | 0.92399 | **0.93113** | 0.92448 | **0.93117** | 0.92604 | **0.93276** | 0.92403 | **0.93144** | 0.92249 | **0.93251** | 0.92326 | **0.93353** |
| mAP 5 | 0.91367 | **0.92123** | 0.91366 | **0.91939** | 0.91589 | **0.92221** | 0.91367 | **0.92051** | 0.91310 | **0.92063** | 0.91284 | **0.92219** |
| mAP 10 | 0.89813 | **0.90425** | 0.89849 | **0.90296** | 0.90050 | **0.90534** | 0.89832 | **0.90556** | 0.89770 | **0.90365** | 0.89698 | **0.90499** |

The results for 20NewsGroup dataset are shown in Figure 5 and Table 3. Similar to previous two datasets, all the results from word/word pair combined model are better than the word-only model. For the $mAP1$ the $mAP3$, the $mAP5$ and the $mAP10$, the most significant improvement happens when Feature Number = 11500. And the improvements are about 10.40%, 11.91%, 12.46% and 12.99%.

Observing the results for all three datasets, we found that they all reflect the same pattern. The word/word pair combined model consistently outperforms the word only model, given that both of them are constructed based on the same number of input features.

#### 4.3.2 DIFFERENT K VALUES

In the second experiment, we focus on how the different K values affect the effectiveness of the generated word pairs in therms of their ability of topic modeling. First, we average the performance


(a) Total feature = 10500

(b) Total feature = 11000

(c) Total feature = 11500

(d) Total feature = 12000

(e) Total feature = 12500

(f) Total feature = 15000

Figure 3: OMDb fix total feature number word/word pair performance evaluation

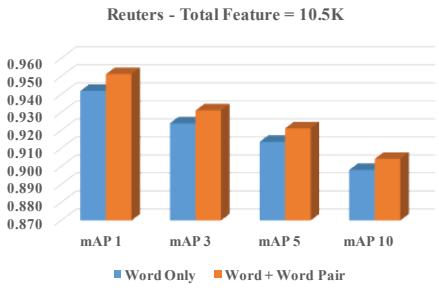
Table 3: 20NewsGroup fix total feature number word/word pair performance evaluation

| mAP | F = 10.5K | | F = 11K | | F = 11.5K | | F = 12K | | F = 12.5K | | F = 15K | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair | word | word pair |
| mAP 1 | 0.73736 | **0.77129** | 0.73375 | **0.76093** | 0.68720 | **0.75865** | 0.73959 | **0.75846** | 0.72280 | **0.76768** | 0.72695 | **0.75583** |
| mAP 3 | 0.65227 | **0.68905** | 0.64848 | **0.68042** | 0.60356 | **0.67546** | 0.65530 | **0.67320** | 0.63649 | **0.68455** | 0.63951 | **0.66743** |
| mAP 5 | 0.60861 | **0.64620** | 0.60548 | **0.63783** | 0.56304 | **0.63321** | 0.61115 | **0.62964** | 0.59267 | **0.64165** | 0.59447 | **0.62593** |
| mAP 10 | 0.55103 | **0.58992** | 0.54812 | **0.58057** | 0.51188 | **0.57839** | 0.55338 | **0.57157** | 0.53486 | **0.58500** | 0.53749 | **0.56969** |

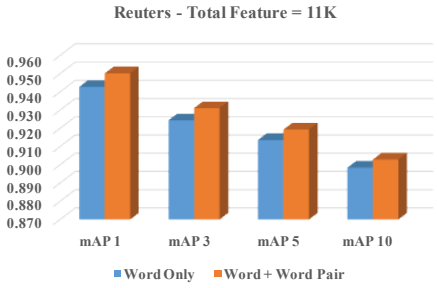
word/word pair combination all K value. The potential K values are 100, 300, 500, 800 and 1000. Then we compare the $mAP$ between our model and the baseline model, which consists of word only input features.

The OMDb dataset results are shown in Table 4. As we can observe, all the K values give us better performance than the baseline. The most significant improvement shown in K = 100, which are about 2.41%, 2.15%, 1.46% and 4.46% for $mAP1, 3, 5, and\ 10$ respectively.

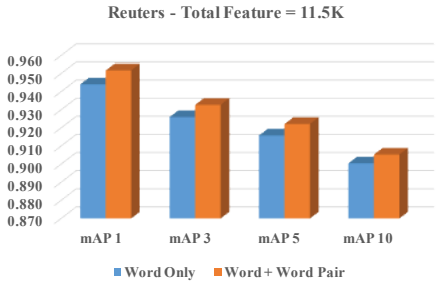
The results of Reuters dataset results are shown in Figure 6 and Table 5. When the K value is greater than 500, all $mAP$ for word/word pair combination model are better than the baseline. Because the



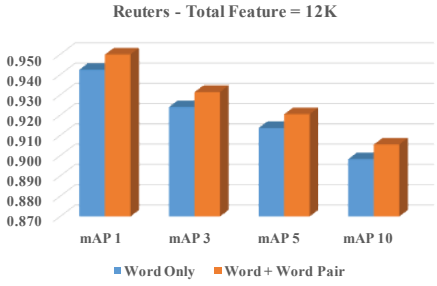

(a) Total feature = 10500



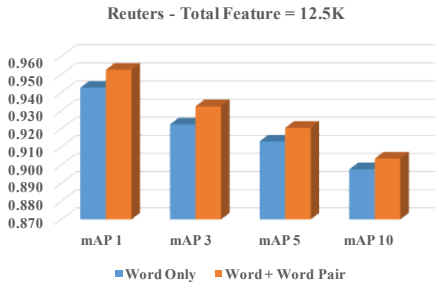

(b) Total feature = 11000



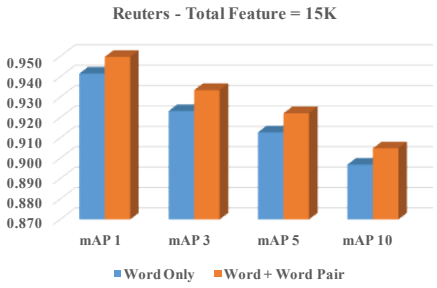

(c) Total feature = 11500




(d) Total feature = 12000




(e) Total feature = 12500




(f) Total feature = 15000

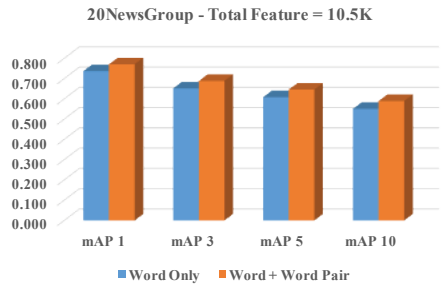
Figure 4: Reuters fix total feature number word/word pair performance evaluation

Table 4: Different K Values for OMDb

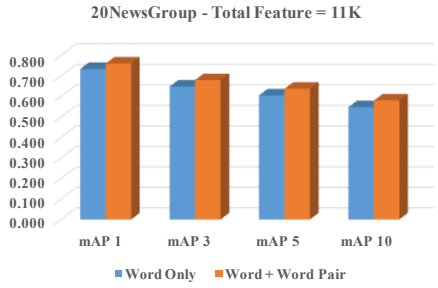
| mAP | Baseline | K = 100 | K = 300 | K = 500 | K = 800 | K = 1000 |
|---|---|---|---|---|---|---|
| mAP 1 | 0.14134 | **0.14474** | 0.14318 | 0.14312 | 0.14212 | 0.14275 |
| mAP 3 | 0.09212 | **0.09410** | 0.09222 | 0.09324 | 0.09219 | 0.09223 |
| mAP 5 | 0.07312 | **0.07419** | 0.07313 | 0.07374 | 0.07293 | 0.07310 |
| mAP 10 | 0.05113 | **0.05341** | 0.05254 | 0.05320 | 0.05243 | 0.05274 |

mAP score for Reuters dataset in original model is already very high almost all of them higher than 0.9, it is hard to get the improvement as large as OMDb dataset. For the $mAP1$, the most significant improvement happens when K = 500, which is 0.31%. For the $mAP5$, the $mAP5$ and the $mAP10$, the most significant improvement shown in K = 800, about 0.50%, 0.38% and 0.42%.

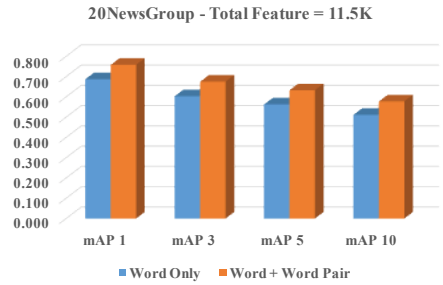
The results for 20NewsGroup dataset results are shown in Figure 7 and Table 6. Similar to the Reuters dataset, when the K value is greater than 800, all $mAP$ score for word/word pair combination model are better than the baseline. For the $mAP1, 3, 5, and\ 10$, the most significant im-



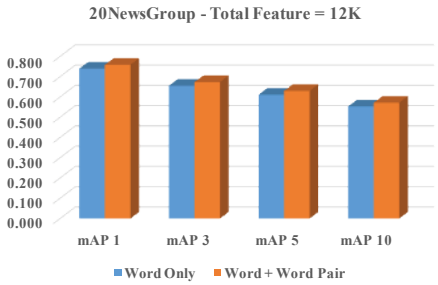

(a) Total feature = 10500



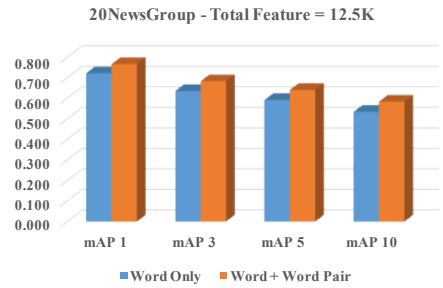

(b) Total feature = 11000



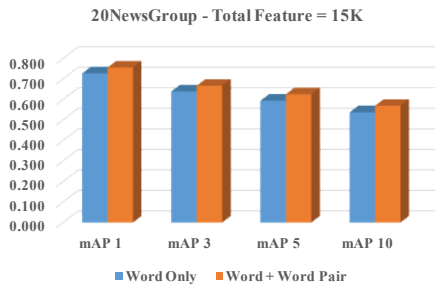

(c) Total feature = 11500




(d) Total feature = 12000




(e) Total feature = 12500




(f) Total feature = 15000

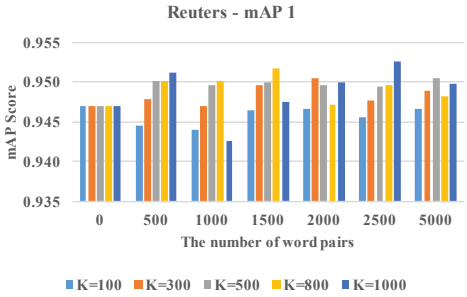
Figure 5: 20NewsGroup fix total feature number word/word pair performance evaluation

Table 5: Different K Values for Reuters

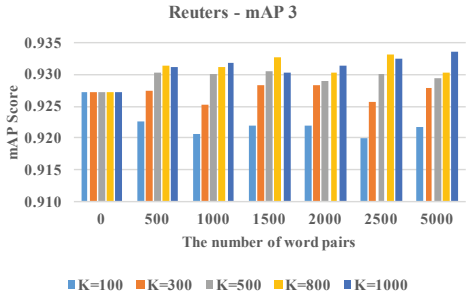
| mAP | Baseline | K = 100 | K = 300 | K = 500 | K = 800 | K = 1000 |
|---|---|---|---|---|---|---|
| mAP 1 | 0.94692 | 0.94565 | 0.94860 | **0.94988** | 0.94955 | 0.94898 |
| mAP 3 | 0.92719 | 0.92149 | 0.92715 | 0.92987 | 0.93146 | **0.93179** |
| mAP 5 | 0.91740 | 0.90859 | 0.91577 | 0.91881 | 0.92064 | **0.92088** |
| mAP 10 | 0.90078 | 0.88975 | 0.89869 | 0.90179 | 0.90403 | **0.90460** |

provements about 2.82%, 2.90%, 3.2% and 3.33% respectively, and they all happened when K = 1000.

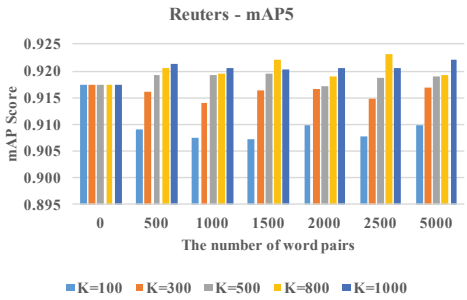
In summary, a larger K value can give us a better result. As we can see from the Reuters dataset and the 20NewsGroup dataset, when K is greater than 800, the combined model outperforms the baseline model in all four $mAP$ evaluation scores. However, the best results appear when K = 800 or K = 1000. For the OMDb dataset, because the $mAP$ score for the baseline model is not very high, the combined model gives better result than the baseline model with any K value that we tested.



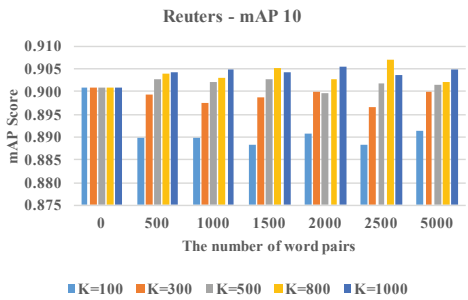

(a) mAP 1 for Reuters dataset




(b) mAP 3 for Reuters dataset




(c) mAP 5 for Reuters dataset




(d) mAP 10 for Reuters dataset

Figure 6: Reuters dataset mAP score evaluation

Table 6: Different K Values for 20NewsGroup

| mAP | Baseline | K = 100 | K = 300 | K = 500 | K = 800 | K = 1000 |
|---|---|---|---|---|---|---|
| mAP 1 | 0.73582 | 0.68122 | 0.71698 | 0.72272 | 0.75207 | **0.75659** |
| mAP 3 | 0.65447 | 0.58838 | 0.62933 | 0.63745 | 0.66834 | **0.67343** |
| mAP 5 | 0.61153 | 0.54296 | 0.58551 | 0.59431 | 0.62492 | **0.63118** |
| mAP 10 | 0.55651 | 0.48424 | 0.52789 | 0.53714 | 0.56724 | **0.57505** |

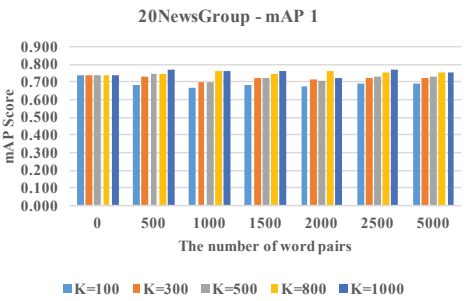
#### 4.3.3 Word Pair Generation Performance

In this experiment, we compare different word pair generation algorithms with the baseline. Similar to previous experiments, the baseline is the word-only RBM model whose input consists of the 10000 most frequent words. The "semantic" word pair generation is the method we proposed in this paper. By applying the idea from the skip-gram Mikolov et al. (2013b) algorithm, we generate the word pairs from each word's adjacent neighbor, and we call it "N-gram" word pair generation. And the window size we used in here is N = 2. For the Non-K word pair generation, we use the same algorithm as the semantic except that no K-means clustering is applied on the generated word pairs.

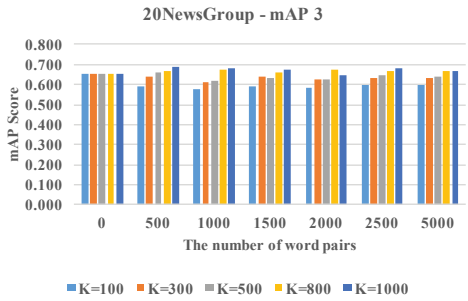
Table 7: Different Algorithms for Word Pair Generation for OMDb

| mAP | Baseline | Semantic | N-gram | Non-K |
|---|---|---|---|---|
| mAP 1 | 0.14134 | **0.14870** | 0.13202 | 0.14302 |
| mAP 3 | 0.09212 | **0.09657** | 0.08801 | 0.09406 |
| mAP 5 | 0.07312 | **0.07635** | 0.07111 | 0.07575 |
| mAP 10 | 0.05113 | **0.05501** | 0.05132 | 0.05585 |

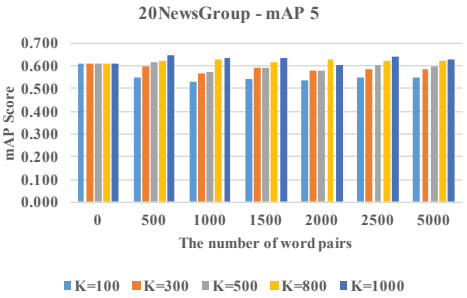
The first thing we observe from From the Table 7 is that both "semantic" word pair generation and "Non-K" word pair generation give us better $mAP$ score than the baseline; however, the $mAP$ score



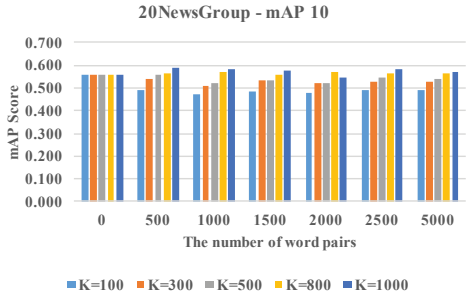

(a) mAP 1 for 20NewsGroup dataset




(b) mAP 3 for 20NewsGroup dataset




(c) mAP 5 for 20NewsGroup dataset




(d) mAP 10 for 20NewsGroup dataset

Figure 7: 20NewsGroup dataset mAP score evaluation

of the semantic generation is slightly higher than the $mAP$ score of the Non-K generation. This is because, although both Non-K and semantic techniques extract word pairs using natural language processing, without the K-means clustering, semantically similar pairs will be considered separately. Hence there will be lots of redundancies in the input space. This will either increase the size of the input space, or, in order to control the input size, reduce the amount of information captured by the input set. The K-mean clustering performs the function of compress and feature extraction.

The second thing that we observe is that, for the N-gram word pair gener-ation, its $mAP$ score is even lower than the baseline. Beside the OMDb dataset, other two datasets show the same pattern. This is because the semantic model extracts word pairs from natural language processing, therefore those word pairs have the semantic meanings and grammatical dependencies. However, the N-gram word pair generation simply extracts words that are adjacent to each other. When introducing some meaningful word pairs, it also introduces more meaningless word pairs at the same time. These meaningless word pairs act as noises in the input . Hence, including word pairs without semantic importance does not help to improve the model accuracy.

## 5 Conclusion

In this paper, we proposed a few techniques to processes the dataset and optimized the original RBM model. During the dataset processing part, first, we used a semantic dependency parser to extract the word pairs from each sentence of the text document. Then, by applying a two way parallel TF-IDF processing, we filtered the data in word level and word pair level. Finally, K-means clustering algorithm helped us merge the similar word pairs and remove the noise from the feature dictionary. We replaced the original word only RBM model by introducing word pairs. At the end, we showed that proper selection of K value and word pair generation techniques can significantly improve the topic prediction accuracy and the document retrieval performance. With our improvement, experimental results have verified that, compared to original word only RBM model, our proposed word/word pair combined model can improve the $mAP$ score up to 10.48% in OMDb dataset, up to 1.11% in Reuters dataset and up to 12.99% in the 20NewsGroup dataset.

## REFERENCES

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

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

Ana Margarida de Jesus Cardoso Cachopo. Improving methods for single-label text categorization. *Instituto Superior Técnico, Portugal*, 2007.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 740–750, 2014.

Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

B Fritz. Omdb api. URL `http://www.omdbapi.com/`.

Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

Geoffrey Hinton and Ruslan Salakhutdinov. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91, 2011.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pp. 1607–1614, 2009.

Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 289–296. Morgan Kaufmann Publishers Inc., 1999.

Intenet Movie Database Inc. The intenet movie database, 1990. URL `http://www.imdb.com/`.

Thomas K Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

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

Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*, pp. 101–110. ACM, 2008.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013b.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4): 235–244, 1990.

Christopher E Moody. Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv:1605.02019*, 2016.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *LREC*, 2016.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 11–18. ACM, 2006.

Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 448–456. ACM, 2011.

Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. In *Advances in neural information processing systems*, pp. 1577–1584, 2008.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008.