Literature Review on Text Mining: Semantic-Unit Mining and Embedding

Hao Wang, Shuo Feng

University of Illinois Urbana-Champaign
Email:{haow4, sfeng15}@illinois.edu

Abstract

Human cognitive system uses semantic-unit(words and phrases) as basic unit to analyze text. While, current embedding methods uses bag-of-word, which means it uses single word as basic unit, to embed text data. To improve the quality of embedding and make it closer to reality, we can change basic unit of embedding from single word to semantic-unit. To do that, we should do semantic-unit mining and modify embedding methods to adapt to new basic unit type. And for semantic-unit mining, there are many phrase mining approaches to help us obtain quality semantic-unit data. Thus, our survey focus on these topics or technologies related to semantic-unit mining and embedding methods. We would introduce the basic concept and representative works in these fields to elaborate semantic-unit mining and embedding.

1 Introduction

For now, the embedding method is based on bag-of-words, which means it uses single word as basic unit to analyze text data. But in fact, human cognitive system uses semantic-unit, which is not only including single words, but phrases, as basic unit to do semantic cognition. Because semantic-unit based embedding is closer to reality, it can improve the quality of embedding by changing basic unit from single word to semantic-unit. Since the method of embedding is improved, we can obtain better quality vectors to improve the effect of embedding applications such as comparing sentences similarity or papers similarity, which computes the similarity of sentences or papers based on the vectors generated from embedding. It is obvious that the quality of sentences or paper similarity can be improved by changing the basic unit of embedding from single words to semantic-unit because using semantic-unit as basic unit on embedding is closer to reality; Besides, semantic-unit based embedding can handle many problems causing by single words level embedding such as phrase paraphrase. For example, bag-of-words based embedding can not paraphrase phrase such as "support vector machine", because current embedding methods generate three vectors for "support vector machine" and each vector represents one word in this phrase. But semantic-unit based embedding regards "support vector machine" as a unit and it generates only one vector to represent this phrase. We can use cos distance or other methods to compute similarities of vectors.Then, we can find similar semantic-unit using these similar vectors. After knowing the motivation, we should analyze approaches to achieve that. The most common way is that pack multi-words which constitute phrases before run embedding method. To pack expected multi-words to phrases, we have to do phrase mining in advance to help identify semantic-unit. Since semantic-unit based embedding is a integration of phrase mining and embedding, we conduct survey on these two topics.

2 Phrase Mining

Comparing with single word, semantic-unit is more effective to manipulate unstructured text data. As an important part of semantic-unit, phrases can help us eliminate ambiguity when we analyze text data. So it is necessary that extracting quality phrase from unstructured massive text data. While, before we introduce phrase mining technology, we should know what kind of phrases we want, more specific, what are high quality phrases? According to (Liu et al. 2015), high quality phrases have these four properties: popularity, concordance, informativeness and completeness.

1. Popularity: The frequences of occurrences of multi-words should be high enough to make these multi-words be regarded as phrases. For example, "data structure" is an important phrase in computer science field, but it is not a meaningful phrase in many other fields such as medical science. So when should we regard "data structure" as a phrase? It depends on the frequences of occurrences of "data structure". In other words, it depends on the input text data. That's why we should do both phrase mining and topic modeling when we extract quality phrases. In general, to avoid this ambiguity problem, we usually do phrase mining on specific domain.
2. Concordance: Concordance is similar to syntagmatic relation. Multi-words can be regraded as phrase must have syntagmatic relation. There are many approaches to verify whether they have concordance property. The principle of most of approaches is that the observed frequence

of collocation of these multi-words are higher than their expected frequence. Mutual information and conditional entropy we learned are typical approaches to check the concordance of text data.

3. Informativeness: It is obvious that the phrases we generated should be meaningful. For example, "this is" have popularity, concordance and completencess. But we are not supposed to generate this collocation as a phrase because it is not informative.
4. Completeness: Completeness, as literally meaning, is the properity to make sure the phrases we generated are complete. For example, when we extract quality phrases from machine learning topic, we are supposed to generate "support vector machine" as a phrase rather than "support vector" or "vector machine".

After knowing what kind of phrase we want to extract, let's focus on the phrase mining technology. Since phrase mining is a very important process for analyzing text data. There are many approaches or methods to accomplish this task. Overall, they can be divided into two fields: natural language processing approaches and data mining methods.

2.1 NLP Approaches

In natural language processing field, phrase mining is an essential part of chunking, which identifies basic elements of a sentence then combine them to generate meaningful multi-words groups such as phrases, noun groups and verb groups. While, it turns phrase mining probelm to label elements of a sentence problem.

The basic idea of chunking is that first, obtain tarining data by annotating massive documents; then, use training data we obtain from first step and part-of-speech features such as noun, verb, etc to train a supervised model. After that, we can use the supervised model on test data to get final result. With the mature of other technologies, more and more technologies are applied on chunking to improve the quality of chunking. Recent trend is that build supervised model using training data and distributional features based on web n-grams.(Bergsma, Lin, and Goebel 2009) The qualities of phrases generated by natural language processing approaches are pretty good. The accuracy of chunking can even reach 95% and F-score of phrase level can reach 88%.

Although the performance of chunking is wonderful, but the limitations are still difficult to solve:

Figure 1: Accuracy on different confusion set(Bergsma, Lin, and Goebel 2009)

Set	BASE	TRIGRAM	SUMLM	SUPERLM
among	60.3	80.8	90.5	92.8
amount	75.6	83.9	93.2	93.7
cite	87.1	94.3	96.3	97.6
peace	60.8	92.3	97.7	98.0
raise	51.0	90.7	96.6	96.6
Avg.	66.9	88.4	94.8	95.7

1. Since it is a supervised model generator, it is inevitable that we have to generate training data based on human effort. That's a huge work. Besides, we can not guarantee the quality of training data since it is based on human effort.
2. Another major limitation is that it is hard to extend to support multi-languages. Because different kind of language have different grammar and the approach of chunking can not handle this problem, it is impossible to support multi languages. If we want to use chunking on multi-languages, we have to generate multi supervised models, whcih means We also have to annotate massive documents in multi-languages as training data.
3. Although we can annotate documents as much as possible, it is still impossible to cover every domain or topic. So it may not fit specific domain or topic such as social media.

2.2 Data Mining Approaches

In data mining field, there are many approaches to do phrase mining. Most of approaches of phrase mining in data mining field are unsupervised. It means we no long need to do huge work on annotating documents. Besides, most of methods are scalable to another language. The base technology of phrase mining in data mining field is frequent pattern mining and topic modeling. In general, we can obtain candidate phrases sets by using frequent pattern mining, which ensures popularity and concordance property; and we can filtrate phrases by using topic modeling, which ensures informativeness and completeness property. We can divide phrase mining methods into three kinds according to the way they combine frequent pattern mining and topic modeling technologies:(1) Infer phrases and topics simultaneously (2) Topic modeling then phrase mining (3) Phrase mining then topic modeling

1. Infer phrases and topics simultaneously:

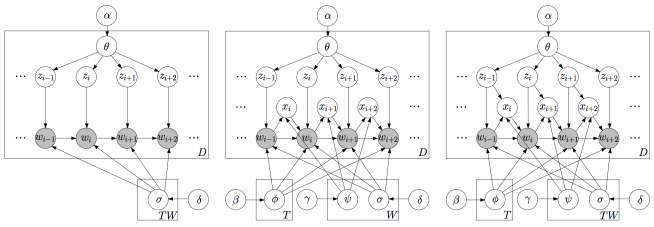
   The processes of phrase mining and topic modeling execute at same time. These two technologies rely on each other and enhance mutually. While, since they rely on each other, we can not guarantee the running time. Besides, over-fitting is another major problem. Classical Models: Bigram Topic Model, Topic N-Grams, and Phrase-Discovering LDA.

   The Bigram Topic Model incorporates n-gram statistics and latent topic to generate a hierarchical probabilistic model. The hyperparameters of model are inferred suing Gibbs EM algorithm, which is similar as ER algorithm we leaned from class. In short, We can treat this model as an extension of unigram topic model to include properties of a hierarchical Dirichlet bigram language model.(Wallach 2006)

   Topical N-Grams model is another probabilistic model. But it pays more attention on text order. It generates words in textual order and creates n-grams by concatenating successive bigrams. In general, the framework of Topic N-Grams model sampling a topic first; then sampling its status a unigram or bigram; after that, sampling

the word from topic-specific unigram or bigram distribution.(Wang, McCallum, and Wei 2007) Thus, Topical N-Grams model has ability to decide whether to form a bigram for the same two consecutive word tokens depending on their nearby context.(Wang, McCallum, and Wei 2007)

Figure 2: Differences of three n-gram models

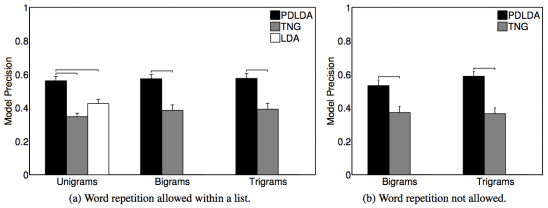


(a) Bigram topic model (b) LDA-Collocation model (c) Topical n-gram model

D: # of documents; T: # of topics; W: # of unique words

Phrase-Discovering LDA(PDLDA) runs phrase mining on every single topic. The basic idea is that the model regards sentences as time series of words and generates the parameter and topic periodically in accordance with the change point indicators.(Lindsey, Headden III, and Stipicevic 2012) In other words, each word is drawn based on previous words and current phrase topic. The model simultaneously infers the location, length, and topic of phrases within a corpus and relaxes the bag-of-words assumption within phrases by using a hierarchy of Pitman-Yor processes.(Lindsey, Headden III, and Stipicevic 2012)The results of PDLDA shows in Figure 3 with contrast among PDLDA, TNG and LDA.

Figure 3: Result of Phrase-Discovering LDA:



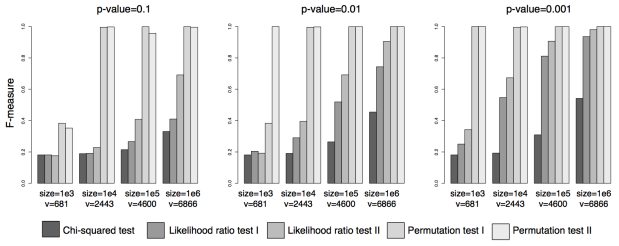
(a) Word repetition allowed within a list. (b) Word repetition not allowed.

2. First topic modeling then phrase mining:

   Comparing with inferring phrases and topics simultaneously, posting topic modeling phrase construction no longer need to do phrase mining and topic modeling at same time. Thus, the running time of most models using strategy that do topic modeling first then phrase mining are much faster than models inferring phrases and topics simultaneously. The quality of extracted phrase relies on topic labels for unigrams. Overall, we have high accuracy for labeling unigrams on topic and we can get high quality phrases and topics. The main idea of topic modeling then phrase mining is using post bag-of-words model inference, visualizes topics with n-grams. Classical models: TurboTopics and KERT(Keyphrase Extraction and Ranking by Topic).

   TurboTopics model performs Latent Dirichlet Allocation on corpus to assign each token a topic label. Then, it merges adjacent unigrams with the same topic label by a distribution-free permutation test on arbitrary-length back-off model. End recursive merging when all significant adjacent unigrams have been merged.(Blei and Lafferty 2009) Basic steps of TurboTopics model:first, run bag-of-words model inference and assign topic labels to each token; second, extract candidate key phrases within each topic with frequent pattern mining; third, rank the key phrases in each topic. The result of TurboTopic shows in Figure 4

Figure 4: Result of TurboTopics:



KERT(Keyphrase Extraction and Ranking by Topic) uses phrase construction as a post-processing step to Latent Dirichlet Allocation. This approach can be divided into two main steps: execute frequent pattern mining on each topic; rank phrases based on the properties of quality phrases we talked in first section.(Danilevsky et al. 2014)The result of KERT shows in Figure 5

Figure 5: Result of KERT for machine learning topic:

Method	Top 10 Topical Keyphrases
kpRelInt*	learning / classification / selection / models / algorithm / feature / decision / bayesian / trees / problem
kpRel	learning / classification / learning classification / selection / selection learning / feature / decision / bayesian / feature learning / trees
KERT$_{cov}$	effective / text / probabilistic / identification / mapping / task / planning / set / subspace / online
KERT$_{pur}$	support vector machines / feature selection / reinforcement learning / conditional random fields / constraint satisfaction / decision trees / dimensionality reduction / constraint satisfaction problems / matrix factorization / hidden markov models
KERT$_{phr}$	learning / classification / selection / feature / decision / bayesian / trees / problem / reinforcement learning / constraint
KERT$_{com}$	learning / support vector machines / support vector / reinforcement learning / feature selection / conditional random fields / vector machines/ classification / support machines / decision trees
KERT	learning / support vector machines / reinforcement learning / feature selection / conditional random fields / classification / decision trees / constraint satisfaction / dimensionality reduction / matrix factorization

3. First phrase mining then topic modeling:

   There is a huge problem which tokens in the same phrase may be assigned to different topics when we first do topic modeling then phrase mining. While, switch the order of phrase mining and topic modeling can avoid this problem. First phrase mining then topic modeling strategy uses prior bag-of-words model inference, mines phrases and imposes to the bag-of-words model. Although we can solve phrase topic ambiguous problem by switch the executer order of phrase mining and topic modeling. It

would raise another problem. The key issue for phrase mining first then topic modeling strategy is that topic inference relies on segmentation of documents, which means we must segment documents correctly.

ToPMine is a representative model using first phrase mining then topic modeling strategy. Comparing with KERT(Keyphrase Extraction and Ranking by Topic), ToPMine do phrase construction first then topic mining. The main idea of ToPMine is that combines a novel phrase mining framework to segment a document into single and multi-word phrases and a new topic model that operates on the induced document partition.(El-Kishky et al. 2014) ToPMine employs PhraseLDA, which is a generative model extended from LDA. The difference between PhraseLDA and LDA is that PhraseLDA incorporates constraints obtained from the bag-of-phrases input rather than bag-of-words. Basic idea of ToPMine framework: first, extract candidate phrases and their counts with frequent contiguous pattern mining; second, perform agglomerative merging of adjacent unigrams as guided by a significance score; third, The newly formed bag-of-phrases are passed as input to PhraseLDA, that constrains all words in a phrase to each sharing the same latent topic. Further more, ToPMine model segments each document into a bag-of-phrases in second step. We can generate high quality phrases.

3 Embedding Method

In this section, we summarize the main concepts and algorithms of mainstream embedding methods. Word embedding is essentially a dense representation of lexical distribution. It can be used to support a variety of natural language processing tasks. (Ling, Song, and Roth 2016). We first introduce the basic word embedding methods, Word2Vector. Next, We will cover more advanced topics such as text embedding, knowledge graph embedding, and sentence embedding.

3.1 Neural Language Model

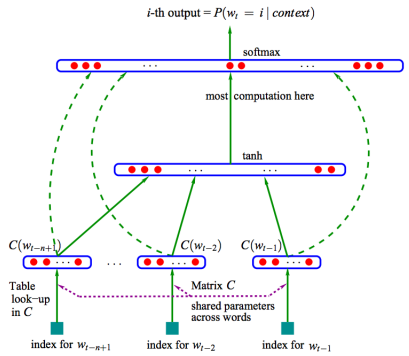
Proposed by (Bengio et al. 2003), the original word embedding uses a shallow neural network for the task of word prediction in a sequence as shown in Figure 6. The way it works is to train the model to maximize the training corpus penalized log-likelihood:

$$L = \frac{1}{T}\sum logf(w_t, w_{t-1}, ..., w_{t-n+1}; \Theta) + R(\Theta),$$

where $R(\Theta))$ is a regularization term. i.e. the probability $p(w_t|w_{t1},,w_{tn+1})$ as computed by the softmax, where n is the number of previous words fed into the model.

However, the invention did not catch much attention until Collobert and Weston defined the unified architecture for natural language processing(Collobert and Weston 2008), in which they used word embeddings for a host of predictions. It paved the way for using neural networks for more applications that many of todays models are built upon.

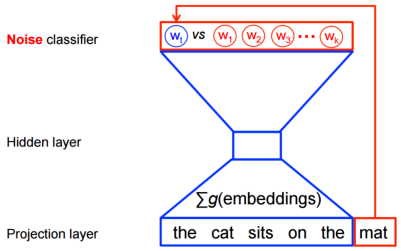
Figure 6: Neural architecture



3.2 Word2vec

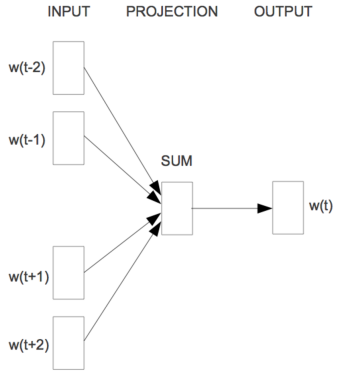
(Mikolov et al. 2013) came up with the well-known and powerful Word2Vec in contrast to traditional ways of representing text information. There are two reasons why Word2Vector has advantages over one-hot vector to represent text. One is that the one-hot vector usually have very high dimensions and could be very sparse, which brings a lot of space complexity. The other one is that the one-hot vector could not use similarity measures such as cosine similarity to measure the distance directly. For instance, two synonyms are mapped to different positions in one-hot vectors which should not happen. Word2Vec uses neural networks as the model while it is a shallow learning model. The basic structure is shown in Figure 8. The network consists of three layers. The input layer reads word one-by-one in a sequential manner. The hidden layer perform the logistic regression. The output is handled by softmax functions to generate predicted word. The output is essentially a multi-class classification problem. After the training, the hidden layer can be used as the feature vector to represent words. In practice, it has two variations: Continuous bag-of-words (CBOW) model and Skip-Gram model. CBOW uses both the n words before and after the target word w_t to predict it. The process is shown in Figure 8. The objective function of CBOW is to maximize the probability of the word given the context

Figure 7: Structure of Word2Vec



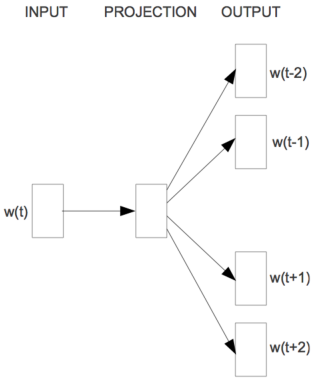
$$J_\Theta = \frac{1}{T}\sum_{T}^{t=1} logp(w_t|w_{t-n}, .., w_{t-1}, w_{t+1}, ...w_{t+n}).$$

Figure 8: CBOW



Skip-gram uses the center word to predict the surrounding words as can be seen in Figure 9. In order to get the proba-

Figure 9: Skip-gram



bilities of the surrounding n words to the left and to the right of the target word, the skip-gram uses the following objective functions:

$J_\Theta = \frac{1}{T}\sum_T^{t=1}\sum_{-n\leq j\leq n,\neq 0} logp(w_{t+j}|w_t)$

Both CBOW and Skip-gram have their own advantages and can be used in different settings. As shown in Figure 10 , one of the interesting facts about Word2Vec is that it efficiently capture the semantic meaning of the words. The semantic difference of meaning can be modeled by math equations. For example, Man-Woman=King-Queen. Besides working as a preprocessing tool, Word2Vector itself can be used to perform other tasks such as clustering and topic modelings.

3.3 RNN Language Model

RNN is a popular model in NLP tasks because it keeps track of the time-series information even when it is far from the current time stamp. The basic structure is shown in Figure 11. Different from the general embedding methods, RNN is a deep learning method which captures more local and global information. (Kim et al. 2015) proposed a simple neural language model that relies only on character-

Figure 10: PCA

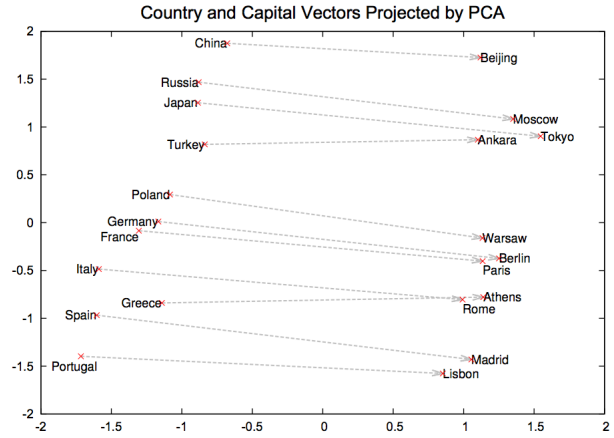
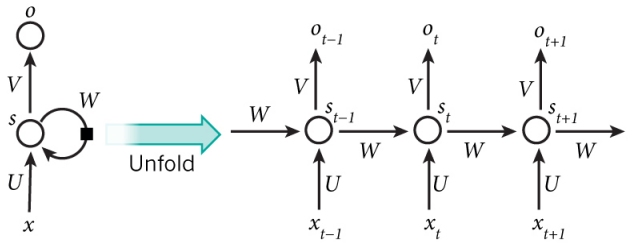


Figure 11: Recurrent Neural Networks



level inputs whereas the predictions are made at the word-level. Their model employs a convolutional neural network (CNN) along with a long short-term memory (LSTM) recurrent neural network(RNN-LM). The LSTM extends RNN by introducing a forget gate to disregard stuff too far from the current stage. (Jozefowicz et al. 2016) summarized advanced strategies in Recurrent Neural Networks for large scale Language Modeling. It is widely accepted that ensemble methods can achieve better results since they combine different models and boost the performance. The authors experimented different ensemble methods and compared their results. Some examples techniques such as character Convolutional Neural Networks or Long-Short Term Memory, on the One Billion Word Benchmark. The evaluation results are represented in Figure 13.

Figure 12: RNN ensemble architecture

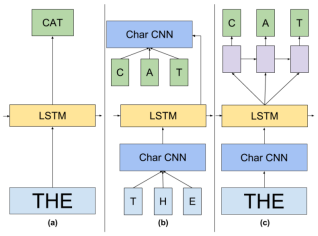


Figure 13: Ensemble Evaluations

Table 2. Best results of ensembles on the 1B Word Benchmark.

MODEL	TEST PERPLEXITY
LARGE ENSEMBLE (CHELBA ET AL., 2013)	43.8
RNN+KN-5 (WILLIAMS ET AL., 2015)	42.4
RNN+KN-5 (JI ET AL., 2015A)	42.0
RNN+SNM10-SKIP (SHAZEER ET AL., 2015)	41.3
LARGE ENSEMBLE (SHAZEER ET AL., 2015)	41.0
OUR 10 BEST LSTM MODELS (EQUAL WEIGHTS)	26.3
OUR 10 BEST LSTM MODELS (OPTIMAL WEIGHTS)	26.1
10 LSTMS + KN-5 (EQUAL WEIGHTS)	25.3
10 LSTMS + KN-5 (OPTIMAL WEIGHTS)	25.1
10 LSTMS + SNM10-SKIP (SHAZEER ET AL., 2015)	**23.7**

Figure 14: Knowledge Graph Embedding

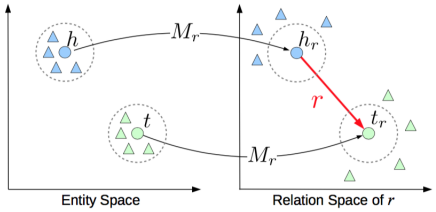


Figure 1: Simple illustration of TransR.

3.4 Knowledge Embedding

Embedding can be extended to areas beyond words as well. Knowledge graph embedding is especially an interesting research area since it encodes the information about entities and their relationships. For example,(Bordes et al. 2011) discusses how to integrate Knowledge Base into statistical learning systems. Different from previous work which maps both entities and relationships into the same space, (Wang et al. 2014) proposed a novel method to map entities and relations into different spaces and then model the projections between them. This design can be illustrated in Figure 14.

3.5 Sentence Embedding

Although word embedding achieves good results, there are inherent weaknesses of it. It is based on the bag-of-words assumption. Bag-of-words representation lacks the order of word and ignores the semantic meaning of words. Therefore, phrase-level and sentence-level representations are researched by numerous people. The most basic way is to take the weighted average of all the words in the document. Another rudimentary approach is to combine the word following the order of a parse tree. These methods are still not efficient to capture the order of the words and model varied length word.

(Le and Mikolov) propose the Paragraph Vector, an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The biggest advantage Paragraph Vector has it that it can be applied to variable-length pieces of texts, anything from a phrase or sentence to a large document. The paragraph vectors can perform different tasks, ranging from producing predictions of the next

Figure 15: Paragraph Embedding

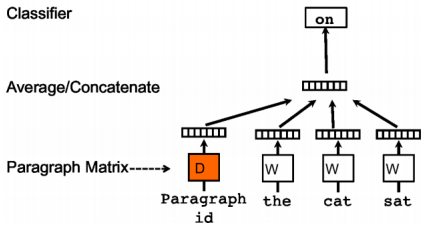


Figure 16: The performance of our method compared to other approaches on the Stanford Sentiment Treebank dataset

Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

word given many contexts sampled from the paragraph, to working as a preprocessing step to other machine learning tasks such as Spam detection, sentiment analysis, and information retrieval. The basic architecture of the Paragraph Vector framework is as follows. As shown in Figure 15, every paragraph is mapped to a unique vector, represented by a column in matrix D and every word is also mapped to a unique vector, represented by a column in matrix W. The paragraph vector and word vectors are averaged or concatenated to predict the next word in a context.

The model is evaluated on sentiment analysis and information retrieval tasks. Figure 16 shows that paragraph vector not only outperforms the traditional machine learning algorithms such as Naive Bayes and Support Vector Machine but also has better results compared to advance RNN-based approaches. This is also illustrated well on another dataset IMDB shown in Figure 17. In terms of information retrieval tasks, the authors perform a lot of queries and compare the similarity of retrieved paragraphs. The result is shown in Figure 18. It turn out that the Paragraph Vector has much more advantages over simple bag-of-words representations. Therefore, it is a great breakthrough with regard to forms of text representations.

(Palangi et al. 2016) develops a model to further improve the performance of Paragraph Vector model in web document retrieval scenarios using recurrent neural networks (RNN) and Long Short-Term Memory (LSTM). The proposed LSTM-RNN model sequentially takes each word in a sentence, extracts its information, and embeds it into a se-

Figure 17: The performance of Paragraph Vector compared to other approaches on the IMDB dataset

Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20 %
BoW (bΔt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

Figure 18: The performance of Paragraph Vector and bag-of-words models on the information retrieval task.

Model	Error rate
Vector Averaging	10.25%
Bag-of-words	8.10 %
Bag-of-bigrams	7.28 %
Weighted Bag-of-bigrams	5.67%
Paragraph Vector	**3.82%**

mantic vector. The biggest advantage of it is that it is able to preserve information far from the current place. After the training process, the hidden layer of the network provides a vector representation of the whole sentence. The structure of the model in this paper is illustrated in Figure 19. Their evaluation result is demonstrated in the Figure 20:

4 Conclusion

In this paper, we explored different topics in text mining and natural language processing. Especially, we focus on the semantic mining and embedding methodologies. We reviewed both the classic models as well as cutting-edge algorithms. These work gave us a deeper understanding of the text mining domain and serves as fundamentals for our future research and work.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Bergsma, S.; Lin, D.; and Goebel, R. 2009. Web-scale n-gram models for lexical disambiguation. In *IJCAI*, volume 9, 1507–1512.

Blei, D. M., and Lafferty, J. D. 2009. Visualizing topics with multi-word expressions. *arXiv preprint arXiv:0907.1013*.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Conference on artificial intelligence*, number EPFL-CONF-192344.

Figure 19: Paragraph Vector Space Model

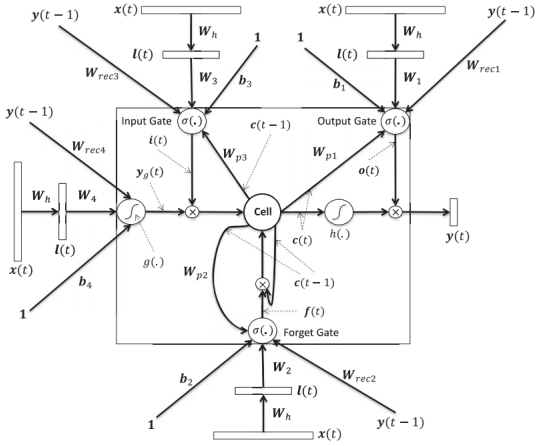


Figure 20: Comparisons of ndcg performance measures

ARE THE BEST RESULTS.

Model	NDCG @1	NDCG @3	NDCG @10
Skip-Thought off-the-shelf	26.9%	29.7%	36.2%
Doc2Vec	29.1%	31.8%	38.4%
ULM	30.4%	32.7%	38.5%
BM25	30.5%	32.8%	38.8%
PLSA (T=500)	30.8%	33.7%	40.2%
DSSM (nhid = 288/96) 2 Layers	31.0%	34.4%	41.7%
CLSM (nhid = 288/96, win=1) 2 Layers, 14.4 M parameters	31.8%	35.1%	42.6%
CLSM (nhid = 288/96, win=3) 2 Layers, 43.2 M parameters	32.1%	35.2%	42.7%
CLSM (nhid = 288/96, win=5) 2 Layers, 72 M parameters	32.0%	35.2%	42.6%
RNN (nhid = 288) 1 Layer	31.7%	35.0%	42.3%
LSTM-RNN (ncell = 32) 1 Layer, 4.8 M parameters	31.9%	35.5%	42.7%
LSTM-RNN (ncell = 64) 1 Layer, 9.6 M parameters	32.9%	36.3%	43.4%
LSTM-RNN (ncell = 96) 1 Layer, n = 2	32.6%	36.0%	43.4%
LSTM-RNN (ncell = 96) 1 Layer, n = 4	33.1%	36.5%	43.6%
LSTM-RNN (ncell = 96) 1 Layer, n = 6	33.1%	36.6%	43.6%
LSTM-RNN (ncell = 96) 1 Layer, n = 8	**33.1%**	**36.4%**	**43.7%**
Bidirectional LSTM-RNN (ncell = 96), 1 Layer	**33.2%**	**36.6%**	**43.6%**

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

Danilevsky, M.; Wang, C.; Desai, N.; Ren, X.; Guo, J.; and Han, J. 2014. Automatic construction and ranking of topical keyphrases on collections of short documents. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, 398–406. SIAM.

El-Kishky, A.; Song, Y.; Wang, C.; Voss, C. R.; and Han, J. 2014. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment* 8(3):305–316.

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Le, Q. V., and Mikolov, T. Distributed representations of sentences and documents.

Lindsey, R. V.; Headden III, W. P.; and Stipicevic, M. J. 2012. A phrase-discovering topic model using hierarchical pitman-yor processes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 214–222. Association for Computational Linguistics.

Ling, S.; Song, Y.; and Roth, D. 2016. Word embeddings with limited memory. In *The 54th Annual Meeting of the Association for Computational Linguistics*, 387.

Liu, J.; Shang, J.; Wang, C.; Ren, X.; and Han, J. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1729–1744. ACM.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Palangi, H.; Deng, L.; Shen, Y.; Gao, J.; He, X.; Chen, J.; Song, X.; and Ward, R. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(4):694–707.

Wallach, H. M. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, 977–984. ACM.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph and text jointly embedding. In *EMNLP*, volume 14, 1591–1601. Citeseer.

Wang, X.; McCallum, A.; and Wei, X. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, 697–702. IEEE.