國立政治大學資訊科學系

Department of Computer Science

National Chengchi University

碩士論文

Master’s Thesis

深度學習對於中文句子的表示

Sentence Representation in Chinese

研 究 生：管芸辰

指導教授：蔡銘峰

中華民國 一百零六 年 十一 月

November 2017

105
碩士論文

深度學習對於中文句子的表示

政治大學資訊科學系

管芸辰

深度學習對於中文句子的表示
Sentence Representation in Chinese

研 究 生：管芸辰　　Student：Yun Chen Kuan
指導教授：蔡銘峰　　Advisor：Ming-Feng Tsai

國立政治大學
資訊科學系
碩士論文

A Thesis
submitted to Department of Computer Science
National Chengchi University
in partial fulfillment of the Requirements
for the degree of
Master
in
Computer Science

中華民國 一百零六 年 十一 月
November 2017

Abstract

The paper demonstrate the popular method in recent years to construct the semantic embedding, and use classification to verify the accuracy of these models on Chinese.

Content

Figure Content

Table Content

Chapter 1

Introduction

1.1 Abstract

How to make the sentence embedding with its own semantics more precisely is study of interest, since it's beneficial for several NLP tasks like machine translation, sentiment analysis. Since the internet text volume grows so enormously and rapidly, and the new derivative or new word keep growing as well. How to make the information can be extracted more efficiently and precisely become more critical for many applications. Chinese forums, blogs or microblog expand especially rapidly. The studies tried to vectorize the sentences with deep learning approach with more general way to make it invariant to the languages properties.

Recently word2vec[13] is considered to work for evaluating word semantics in general cases. Additionally, the character is invariant to the language. Nevertheless, the embedding in sentence or pharse level is more complicated, it's related to the sentence structure, intention or context. There are several methods raised in recent years, like Siamese-CBOW, FastText ...etc. Most of them are able to train batch of text to construct sematic vectors.

1.2 Purpose

So far, most the studies are conducted in English or more general way to applied in various languages ,since the most platforms are contributed by the worldwide users. Most approaches also are aimed at being invariant to language properties or appliable to multilanguage environment. However, few of them evaluate the effectiveness of these approaches to other languages, or they evaluate the multiligual dataset without considering the characteristic of other languages. we are also interested if those models also works in

Chinese or other languages, and if the algorithm is invariant to the language grammar or language property.

In this paper, we will demostrate the modern methods on practical data, and compare it with traditional methods.

Chapter 2

Related Work

2.1 Traditional Approach

[3] summarized both corpus-base and lexicon-base techniques purposed recently and listed the languages those techniques aiming at, and there are some innovative mothods combined both approaches. The advantage of corpus-base is that it is dictionary-free, but it requires relatively larger corpus to build the model, while lexicon-based approach depends mainly on existing resources to detect the sentiment. When lexicon-based approach comes to the informal articles contributed by netizens, it may suffer some troubles like misspelling, abbreviation ,words or metaphor...etc, neither can it take the sequence of words into consideration.

The basic corpus-based approach like TF-IDF is considered to be able to generate relatively good precision. However, both approaches may utilize some keywords in the sentences rather than sentiment of sentence itself. In real world, we often use negation or irony to present our feeling rather than solely keywords. To solve the problems from rapidly-evolved languages, there are both semi-supervised and unsupervised approaches introduced as well.

The another problem from the approach is the sparse matrix and high dimension vector due to the complexity of the language nature. Sparse matrix results in inefficience of classifiers and difficulty of scaling up. Additionally, the relationship of synonymy can not be modeled. A classic method for generating dense vectors is to utilize singular value decomposition, SVD. SVD is part of a family of methods that can approximate an N-dimensional dataset using fewer dimensions, including Principle Components Analysis (PCA), Factor Analysis, and so on. Nonetheless, there is a significant computational cost for the SVD for a large cooccurrence matrix, and performance is not always better than using the full sparse vectors.

2.2 Chinese Related Sentiment Analysis

In recent years, most models are aimed at English or more general way.

When it comes to multiligual environment, the preprocess approach may differ in languages. The traditional ways to counter the variation of words like stemming or lemmatization are appliable to most Latin languages. However, in Chinese and Japanese, segmentation may also be invloved. In the example of FastText[7], they also demostrated to convert character into pinyin, which make the subword infomation can be obtained.

Though most approaches are tested and verified by English dataset, there are some work to test in Chinese dataset as well.

[17] performed the basic way to classify the articles from WeiBo with Naive Bayes and smoothing with Laplace smooth. In this work, the authors also use emoticon as the ground truth to verify the approach, it also applied some imcrement learning.

2.3 Advanced Approach

Besides the tradition approaches, which try to extract the effective features. With deep approach, it may be possible to learn more in continuous representations. Additionally, sentiment analysis with typical deep learning models are conducted. Multiple tasks are performed like parsing (Socher et al., 2013a), language modeling (Bengio et al., 2003; Mnih and Hinton,2009) and NER (Turian et al., 2010). For sentiment analysis, the legendacy models were also tested, like CNN [9], RNN [1], but most of them are applied in English dataset only.

Recently, word2vec(Mikolov et al. 2013)[13] is considered to perform efficiently to vecctorize the meaning of single words. It is a log-bilinear model to learn continuous representations of words on very large corpora efficiently. Therefore, the same concept can be adapted to phrase or sentence level as well. Mikolov also purpose it with sentence level [11] called PVDB, and claimed it can applied to both short text and long article. These approaches are unsupervised, but it can be conducted to sentiment analysis with proper transformation.

The other way to model the sentimatic is using encoder-decoder model, which come from statistical machine translation. Skip-thoughts[10] employs the GRU encoder-decoder models. And it also combines ”vocabulary expansion” from [12], they used vocabulary expansion to map word embeddings into the RNN encoder space. Therefore, it made it possible to use less vocabulary to build model, and reuse pre-trained model.

To detect the sentiment polarity of short text has attracted the interest of study as well, the most used dataset is informal tweet, which is contributed by the netizens from different background. For the task of sentiment classification, an effective feature learning method is to compose the representation of a sentence (or document) from the representations of the words or phrases it contains (Socher et al., 2013b; Yessenalina and Cardie, 2011).

Pang et al. (2002)[14] already used bag-of-word representation, and present the word as one-hot vector, and they get the better classification results. However, it is still not enough to represent the complex meaning or linguistic characteristics. The following works, like [15], the paper purpose to use deep learning to do sentiment analysis directly.

There is also a work[16] to evaluate the multiligual approach and monoligual one. However, it used the Spanish and English as target, both two are belongs to Indo-European languages. It also addressed the culture difference, ”dragon” mean harmful in English but it’s opposite in Chinese.

Chapter 3

Methods

3.1 The model introduction

Here are some models we tested, including TF-IDF, fasttext, PVDB, and siamese-CBOW.

3.2 TF-IDF + SVM

TF-IDF represent as ”term frequency–inverse document frequency”. The conventional way to evaluate the semantics based on the occurrence of words and term, and it also takes the occurrence of word in global context into consideration, which means the more entry the word present, less meaningful it is.

It is simple and effective, but it still suffers from some disadvantages like data sparsity and high dimension, which may slow down the classifiers, and inability to model synonymy.

The SVM we conducted is LinearSVC in sklearn, which uses linear kernel. It used one-vs-rest strategy to handle the multiclass cases. In the other word, it generates less models than SVC. We used default parameters. The penalty is l2, and loss function is squared hinge.

3.3 FastText

The approach is purposed by [7]. The structure of FastText can be considered as extension of word2vec as well, and it uses the hierarchical softmax to compute the probabilities for predefined classes. However, the key difference of FastText from word2vec is that it employed bag of N-grams feature. For example the word vector “apple” is a sum of the vectors of the N-grams “¡ap”, “app”, ”appl”, “apple”, “apple¿”, “ppl”, “pple”, “pple¿”, “ple”, “ple¿”, “le¿”. This approach tries to take the local word order into consideration and evaluate the partial information from spelling. It may also vectorize the rare words better, which have less neighboring words to model.

While the computation and space complexity increase, it employs the hashing trick like word2vec as well. The word representation is looked up through a table and finally averaged into the text representation. For the words absent from word embeddings, it uses subword infomation[2] to guess the meaning of the word. Another trick applied is pruning some of the vocabulary elements. Feature selection among N-gram is very inefficient and complicated. Here it used online parallelizable greedy approach: To check if the document is covered already, if not, add one with highest norm.

Finally it uses the linear classification to classify the data. The classifier called ”Product quantization” uses some tricks to speed up, which utilizes compressed-domain approximate nearest neighbor search (Jegou et al., 2011) [6]. The compression technique approximates a real-valued vector by finding the closest vector in a pre-defined structured set of centroids. The original PQ has been concurrently improved by Ge et al. (2013)[5] and Norouzi & Fleet (2013), who learn an orthogonal transform minimizing the overall quantization loss. The technique may scrifices some accuracy to gain much more performance and efficiency of memory. It is proved by their experiment that with normalization, both PQ and OPQ are almost lossless with 4 subquantizers.

We used the released version from Facebook github.

3.4 Paragraph Vector

This is method is purposed in [11]. The idea is obtain the summary of paragraphs, sentences or documents. There are 2 different algorithms we tested, which are DM(distributed memory) and DBOW(distributed bag of words). The DM model in figure 3.2a is quite similar with Word2Vec. Figure 3.2b show the model archeture. Compared with DM model, DBOW conceptually simple, this model requires to store less data.

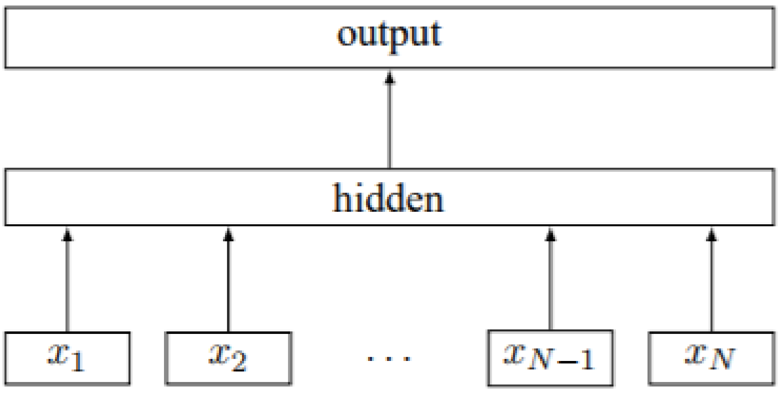


Figure 3.1: The architecture of fasttext

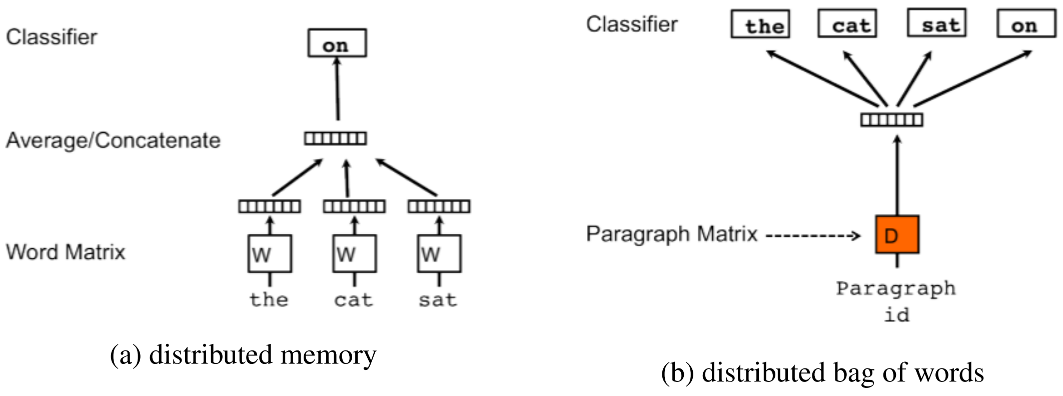


Figure 3.2: Paragraph vector

The paragraph vectors are asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph. The contexts are fixed-length and sampled from a sliding window over the paragraph. Therefore, DM take the sequence into consideration.

However, in DBOW model, it ignore context words from input and force the model to predict words randomly sampled from the paragraph in the output. It is quite similar to Skip-gram in word2vec.

The author suggested the DM may be better in general cases, while DBOW may took less resources. Two models are also can ne concatenated, which means combine both models together. The author claimed it is appliable to both short sentence and long paragraph.

We use the implementation of Gensim and use SVM with linear kernel to classify.

3.5 Siamese-CBOW

The Simese-CBOW[8] computes a sentence embedding is to average the embeddings of its constituent words, instead of using pre-trained word embedding.

It applied the concept of bag of word from Word2Vec. It used the average from the words composing the sentence and use it to evaluate the possibility to predict the sentence around. The architecture shows as Figure 3.3

We used the implementation (https://bitbucket.org/TomKenter/siamese-cbow/overview) from the author, and made it compatible with python3 for better compatibility with unicode.

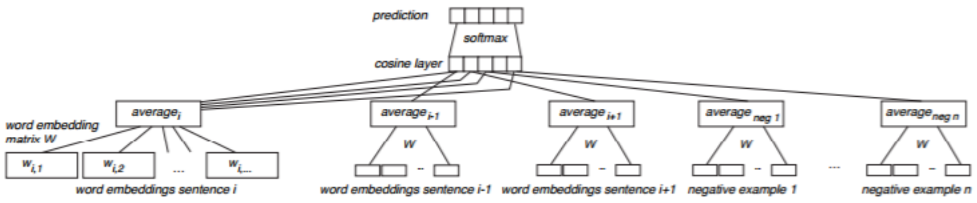


Figure 3.3: The architecture of Siamese-CBOW

Chapter 4

Experiment

4.1 Set Up

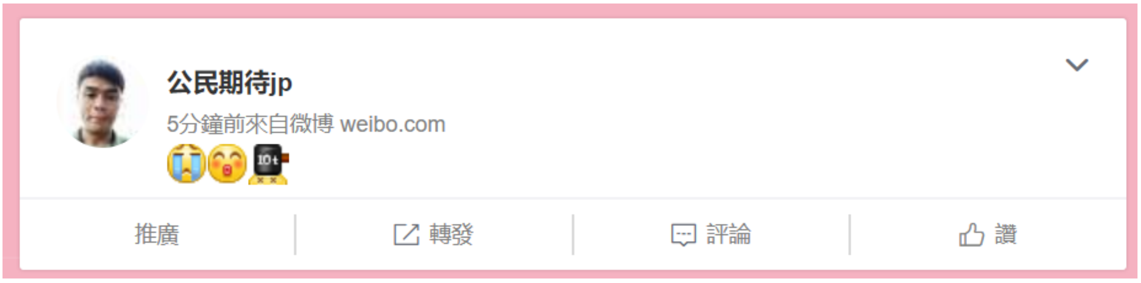


Figure 4.1: The emoticons in WeiBo

The dataset we chose is Open WeiboScope[4], which is collected WeiBo randomly with API by researchers at the Journalism and Media Center of the University of Hong Kong in 2012. It contains 226 millions posts distributing evenly over the year. The most weibo users come from the different province of China. There are also some users from Hong Kong or oversea. The content of weibo contains both simplified Chinese and traditional Chinese.

It's a Weibo feature to allow the user to use emoticon, and the emoticon in raw data

Table 4.1: Tag Category

JOY	呵呵 酷 贊 鼓掌 耶
DISGUST	黑 汗
SAD	可憐 淚 衰 失望 心 生病 囧 鄙 淚 衰 失望 心 生病 囧 鄙
FEAR	委屈 可憐
SURPRISE	吃驚 吃惊
ANGER	怒 抓狂

Table 4.2: Number for categories

ANGER	331091
DISGUST	261955
FEAR	151564
JOY	717059
SAD	788492
SURPRICE	191974

expressed as [笑](smile),[淚](tear). It displays as images like the Figure 4.1. We used the tags in posts as the indicators of sentiment, and removed some duplicated posts or some posts without any tags, or too many tags. We evaluated the accuracy of the classification for different algorithms. We used the TF-IDF and SVM (Joachims, 1998). as baseline.

4.2 Preprocess

For the data preprocessing and cleansing, most posts contains more than 1 tag. To avoid ambiguity, we only preserve those with single tag. we removed the posts containing too many tags, or without any tag. We also removed the duplicated posts by their post id roughly because it is a property of Chinese microblog [4] for Chinese netizens to post repeatedly. Besides, we only chose the posts over certain length (over above 10 characters). Finally, we used jieba and dictionary to segment to post.

We used most-used 6 emotion which most social network support: JOY, SAD, ANGER, FEAR, SURPRICE, DISGUST. We classify these tags into these classes manually. Like [17], we also suffered the problem that the numbers of emoticon classes skewed. The numbers of JOY and SAD are more than 50% of posts. We only selected some specific tags from JOY and SAD to make the whole dataset more balance. The mapping table shows in Table 4.1. The JOY contains the tags like 呵呵(haha), 贊(excellent)...etc. The SAD contains the tags like 失望(disppointed), 淚(tear).

We removed the tags from the original post, and there are so many tags. The posts left for 6 categories display in Table 4.2. The majority classes are still JOY and SAD.

After initial round, we found some special string or tokens like username or url may affect the result. Therefore, we also removed those special tokens from the posts as well.

Table 4.3: FastText Dataset

no segmentation	弊喇,好似有少少喉嚨痛添!
segmentation	弊 喇 , 好似 有 少 少 喉嚨痛 添 !
segmentation + pinyin	bì lǎ , hǎosì yǒu shǎo shǎo hóulóngtòng tiān !

4.3 PVDM

In the Paragraph vector experiment, we tested both DM and DBOW. Additionally, there are 2 different DM supported by gensim to use average or concatenation. We use DM/C and DM/M to represent concatenation and average separately, and used the parameters suggested for 3 models.

4.4 FastText

In the FastText experiment, we tried 3 formats, including non-segmented dataset, segmented dataset and the dataset with pinyin. We tested the non-segmented dataset, since some training data in demostration is Japanese without segmentation. We wonder if the segmentation matters. Additionally, we also want to test if the subword information work for Chinese. So we convert the dataset to pinyin as well. We can see the differences from the table4.3.

For converting to pinyin, we use jieba + pinyin (https://www.npmjs.com/package/pinyin) npm package to convert the charaters to pinyin, which also includes the tone. We used built-in classifier to classify the test set.

We also tested the both dimesion size from 8-300, and loss function including hs (Hierarchical Softmax), ns (negative sampling), softmax.

4.5 Siamese-CBOW

In the siamese-cbow, we use siames-cbow with the default parameters that the author suggested. The dimension is 100, and update algorithm is ada delta. We ran it with epoch 5, 10 separately without any pre-trained word embedding, so it generates the word embedding from scratch. Due to low performance of the trial without pre-trained word embedding, we also tried to use gensim to generate the pre-train word embedding from our dataset. With pre-trained word embedding, we tried epoch 10 to run.

After the vectors are generated, we use both multinomial Naive-Bayes and Linear SVC to classify the results.

Chapter 5

Conclusion

5.1 Experiment Settings

We used baseline TD-IDF plus SVM with linear kernel as baseline. Since the original distribution for classes is a little skewed, most the test sample is classified into 2 major classes. We compared it with other models with different settings.

For PVDB, we use 3 different models dm/c and dm/m and dbow. All of them, we choose most commonly-used parameters,dm : dimension:100, window size:10, negative:5, hs:0 and we tested both dm with concatenation of context vectors (dm/c) and average of context vectors(dm/m). The other model dbow, we chose the same parameters.

In FastText experiment, we iterated through the parameters like window size from 8 to 100, loss function ns,hs,softmax. Since the result did't indicate significant difference between these parameters, we only display 1 of them as reference.

Additionally, we also tried to convert data set to pinyin to evaluate if the pinyin improve the sematic recognition for FastText, which support vocabulary expansion with

Table 5.1: Results

Tf-IDF	0.44 ($\pm$ 0.04)
PVDM(dbow)	0.40
FastText	0.51
FastText(Pinyin)	0.51
Siamese-CBOW(5)	0.41 ($\pm$ 0.04)
Siamese-CBOW(10)	0.39 ($\pm$ 0.03)
Siamese-CBOW(10) with pre-train	0.45 ($\pm$ 0.02)

Table 5.2: FastText

	8	12	16	32	64
no segmentation	0.369	0.375	0.389	0.372	0.368
segmentation	0.515	0.515	0.514	0.516	0.513
segmentation + pinyin	0.513	0.518	0.516	0.517	0.51

Table 5.3: Result of PVDB

	Test set	Training Set
dm/c	0.384	0.384
dbow	0.404	0.457
dm/m	0.38	0.436

subword information [2].

5.2 Result

It took around 2 days with GPU NVIDIA TITAN X (PASCAL) to finish Siamese-CBOW word-embedding training. With both 5,10 epoch the perforamance are below the baseline. We would discuss in the Discussion. The other models can be complete with CPU within hours.

We also look into the classification result with confusion matrix in Figure 5.1 and Figure 5.2a.

That we can see that the 2 major classes get best accuracy while the test results skewed into these 2 major classes as well. The ANGER, DISGUST, and FEAR are more likely to be classified as SAD. It is reasonable that those posts contain more negative feeling. The tendancy in the Siamese-CBOW result is more obvious. And it classify no entry into some rarely-used classes.

In the confusion matrix for fasttext in Figure 5.3b, it shows different tendancy for Pinyin dataset classification. The accuracy for SURPRICE is higher than SAD. It indicates that the SURPRICE may be modeled better than SAD, which contains more sample.

Figure 5.3b is quite similar with other confusion matrix, where most test data are classified as JOY and SAD.

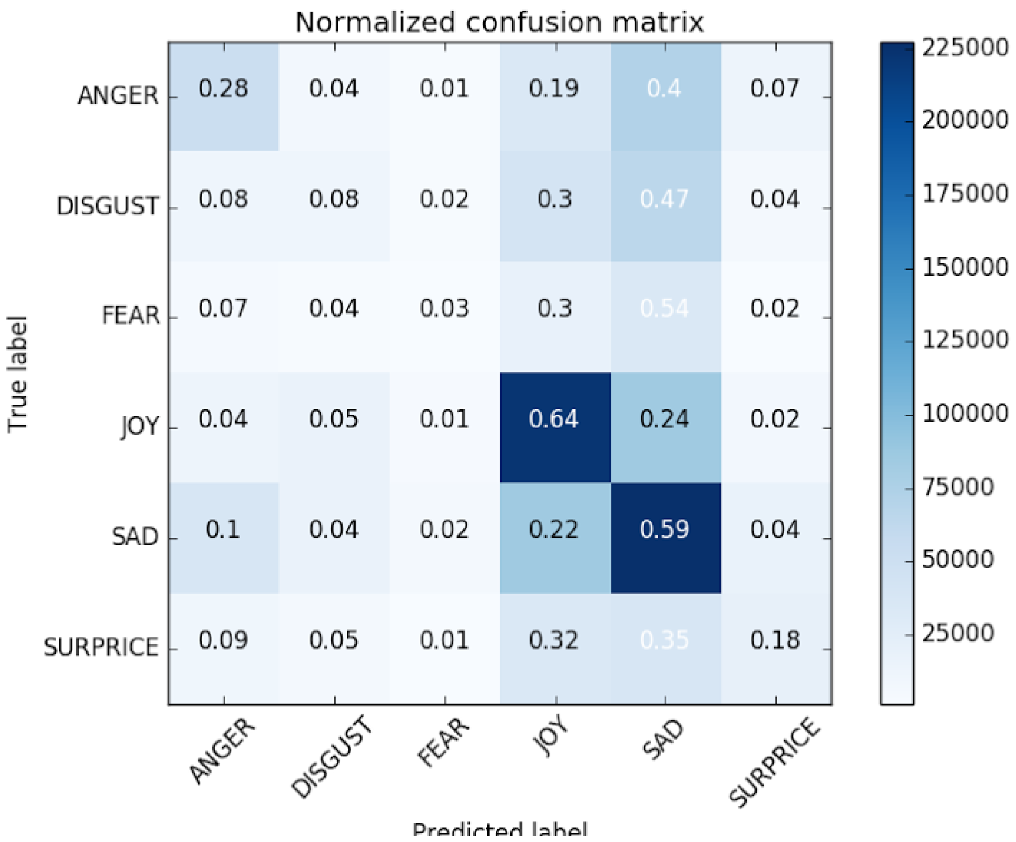
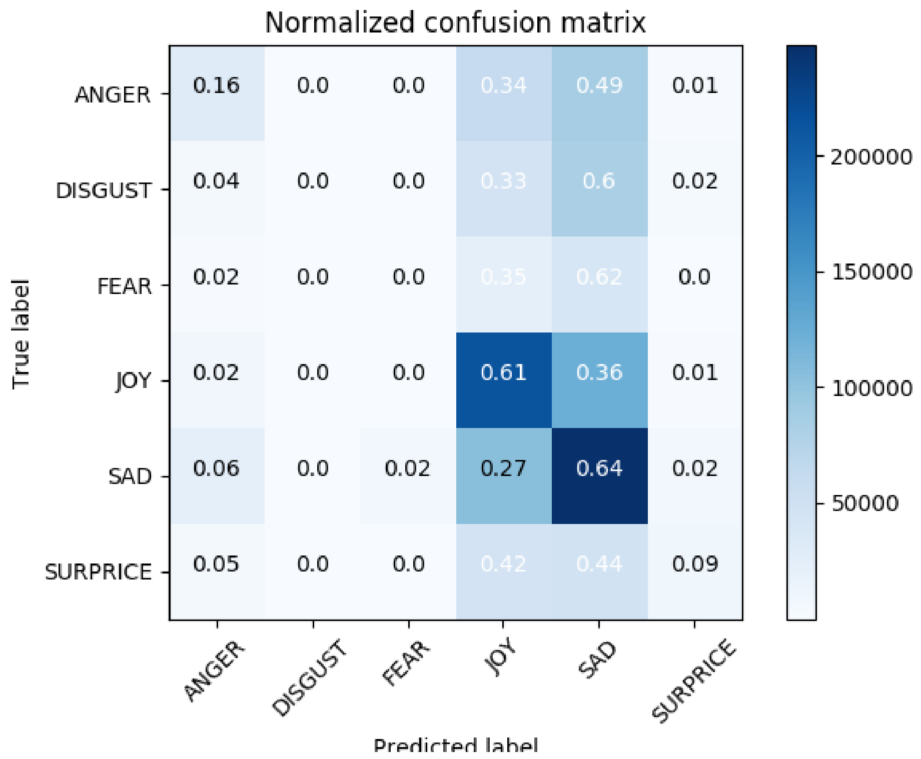
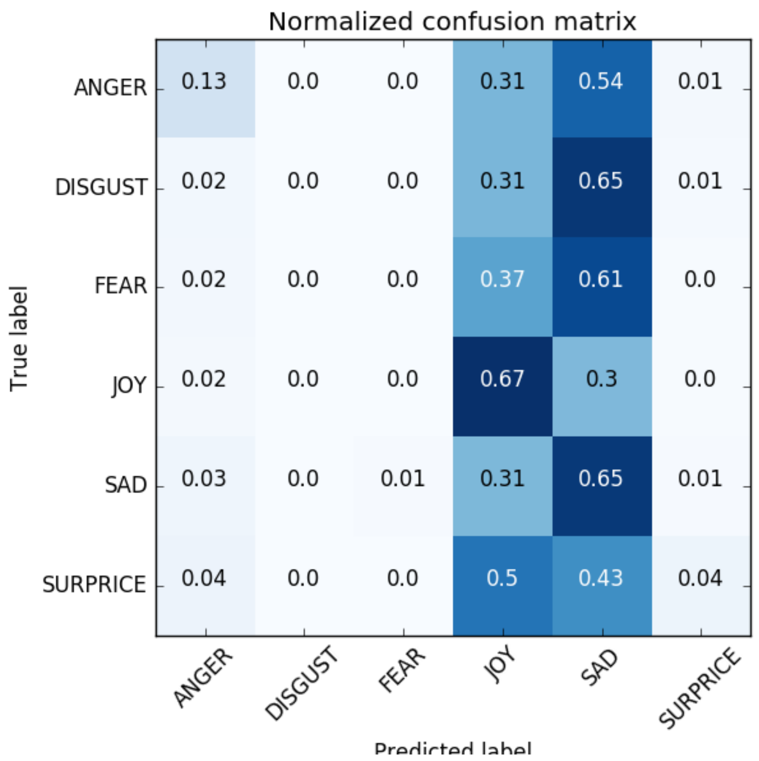


Figure 5.1: The confusion matrix for TF-IDF+ SVM

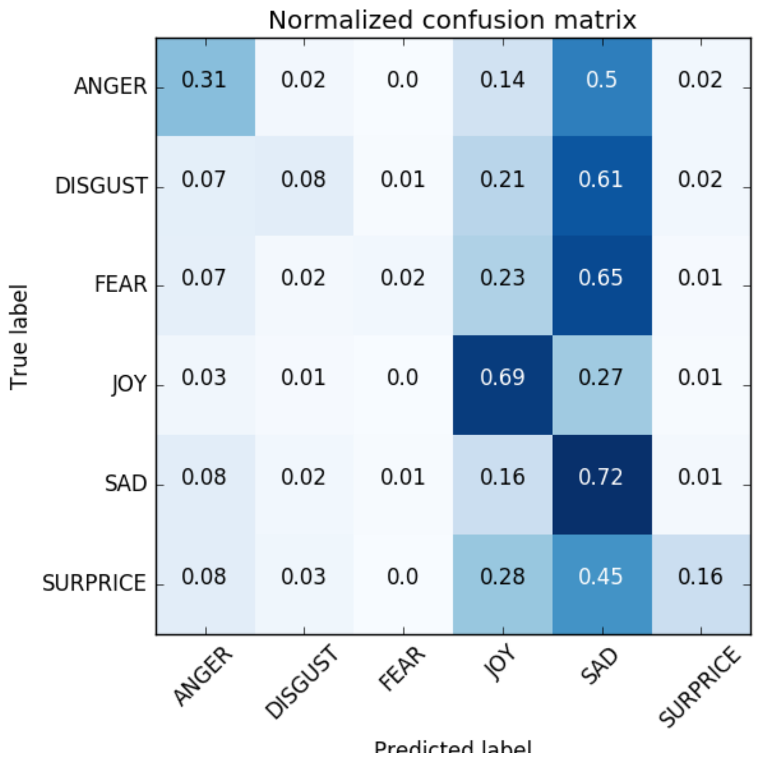


(a)

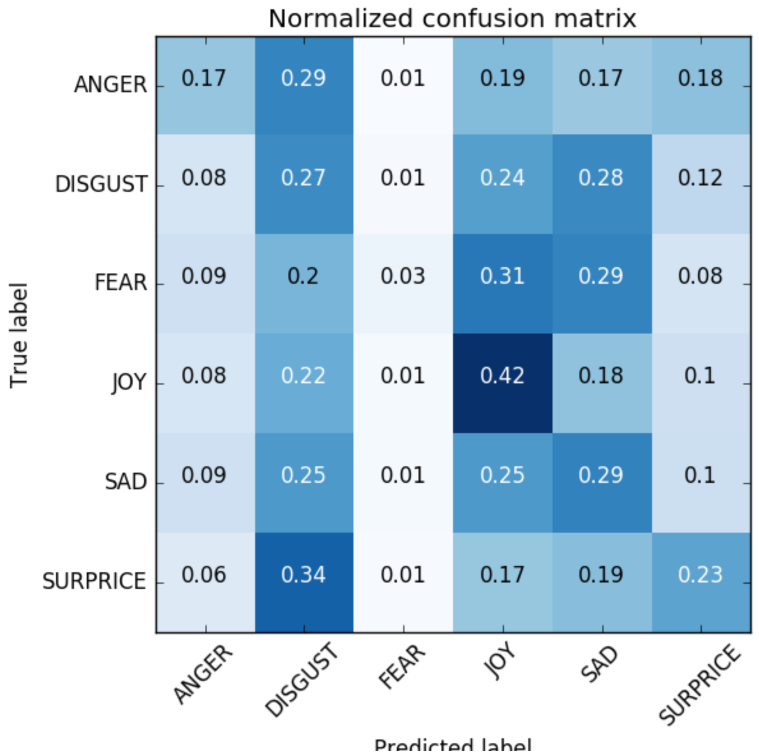


(b)

Figure 5.2: (a) The training set without pre-trained embedding, (b) The training set with pre-trained embedding



(a)



(b)

Figure 5.3: (a) segmented dataset trained with dimension=300, window size=20, loss function hierarchical softmax, (b) pinyin dataset with dimension=200, window size=15, and loss function = softmax

Chapter 6

Discussion

The result shows that FastText can archive better accuracy in general way. We also compare the result between different models.

6.1 Discussion

For the baseline, though TF-IDF it can archive the accuracy about 0.44($\pm$0.04). The most distinguishable features they use are some rarely used terminology. Since we only removed the duplicated post roughly, it may still suffer from the duplicated post from different sources with certain rarely-used words. In general, the model is not general enough, it may not be applicable when the data set changed.

Additionally, compared to other methods, those approaches convert the sentence to vectors with lower dimensions. Theoretically, it may benefit the classifier with more dense presentation. Sparse matrix cause much more time and space to train. With such high dimension vectors, it is unlikely to apply the non-linear classifier.

Generally, FastText can retrieve the better accuracy with segmentation inclusion, and is able to train the large-scale dataset more efficiently with its own classifer. With conversion to Pinyin, it also achieves the similar accuracy. Though, we tried the different settings for FastText, the accuracy is not different significantly despite of the various settings of loss function, window size and dimensions. In the comparison set, segmented dataset outperforms the one without segmentation. It suggested that the term itself may be more meaningful than a single character. And it also took much less time than that of other implementatons. Although we did not perform it with linear classifier like that in other experiments, we can consider that the accuracy is not lossless.

Table 6.1: The feature extracted from TF-IDF

ANGER	DISGUST	FEAR	JOY	SAD	SURPRICE
想瘦	抬杠	含	余	grunewald	印机
造謠	款	害命	符离	省	窑
浪	由此看	表同情	守	春	54:03.7
缸	淫	体	神明	前	清福
行利息	觀天象	可憐見	asce	已閱	任免
冕冠	解除	突如其	三元里	q1050505041	karei
落水狗	上刑	万念俱灰	何苦哉	伐	一
剖腹自	矢	供站	太妙了	离世	sikucd
下	超主	劳資	面三刀	噴火龍	touchsmart610
多吉	美國使館	深	slient	查無此人	翻筋斗

6.2 Baseline

We used TF-IDF as baseline, the result is 0.44(±0.04), which is statistically greater than the random result in 6 classes problem, even we consider the skewed classes. We tried to evaluate the feature that TF-IDF use to classify as Table **??**. Though some term here seemed related like, 2nd place in ANGER is 造謠 (create humor),3rd one is 浪(violent billow). The 2nd place in FEAR is 害命(commit murder for money). The most term is not so related intuitively.

Most of words in each classes actually are not used so commonly. It is contributed by the problem of IDF, which may overweight some rarely-used words. The other problem is that we observed the netizen keep posting something quite similar like advertisement, joke or news..., Though their content are not the same exactly, the words or terminology they used may be quite common. Besides TF-IDF, most algorithms assume that the articles themselves are distinct from others. In real world, it is not a common case except some well-organized website articles. We may require some extra preprocess procedure to handle to prevent intentionally repeated posts.

Some publications[**?**] shows that the posters may use some morphs to avoid censorship, which we can't evaluate how these words contributes to the sentiment analysis. However, the similarity of morphs is quite easy to identified by Word2vec with proper context. It is relatively difficult to identify in TF-IDF model.

6.3 Siamese-CBOW

The Siamese-CBOW, the performance is below the baseline. We tried evaluate the model it trained, it seemed it is not converged enough. The word embedding is not converted correctly. For example, we use ”我” (I) to query in the word-embedding generated by Siamese-CBOW 5 epoch. The related words it show are 扙(hurt, rarely-used word),第四 (the fourth quarter) and 賈寶玉(the name). With 10 epoch word embedding, the related words are 几家 (Some homes), (peek), and 速成班 (rapid-archieve class). It seemed there is no sign of converge.

In the confusion matrix, we found the most tested result fall into two major classes. It is quite similar as other models due to data imbalance.

We tried to use pre-trained word-embeddings with 10 epoch to improve it, and the result is improved to the level of baseline. However, when we assessed the embeddings it generated, the embeddings are still far from converging. According to original paper, the proper embedding can be trained properly. We are not sure if the property is not available in Chinese dataset.

In the original paper, the dataset they conduected is Toronto Books, which contains novels, therefore the sematics of the sentences may be more highly coherent with previous sentence and next one. The property of the dataset should not affect the word embedding it trains. However, it may affect how it determines the relationship between sentences in our cases.

Using some pre-trained embeddings may help to improve the performance. Another drawback is that Siamese-CBOW does not support the feature like subword information. It means if the words is absent from its training dataset, it would be consider non-existent at all. Conducting the vocabulary expansion like that in Skip-Thoughts may assist the problem.

6.4 PVDM

In paragraph vector experiment, the result shows that DBOW produced best accuracy among 3 models. In the original paper, the author suggested that the DM is consistently better than DBOW , and that the sum version of DM is often better than concatenation. So far, it is not clear under what condition that DBOW outperform the DM model. We tried leverge the model it built. We fetched most similar word of I (我) as Table 6.2. Surprisingly, the similar words of DBOW are all not related words. Both DM/C and

Table 6.2: The most similar 5 words of I (我) in 2 models of PVDM

	DM/C	DBOW	DM/M
1	俺	三	偶
2	偶	田徑運動	他
3	老子	暖人	俺
4	哀家	youtudou	我
5	皮下	化妝水	她

Table 6.3: Similar words with ”nǐ”(you) in Pinyin dataset

word related	Chinese
wǒ	我(I)
nǐzìjǐ	你自己(yourself)
nǐmén	你們(you)
,”nǐ	,”你(,”you)
shuí	誰(who)
shuítāmā	誰他媽(who in the hell)
wǒhuì	我會(I can)
,”shuí	,”誰(who)
,”shuítāmā	,”誰他媽(who in the hell)
biérén”	別人(others)

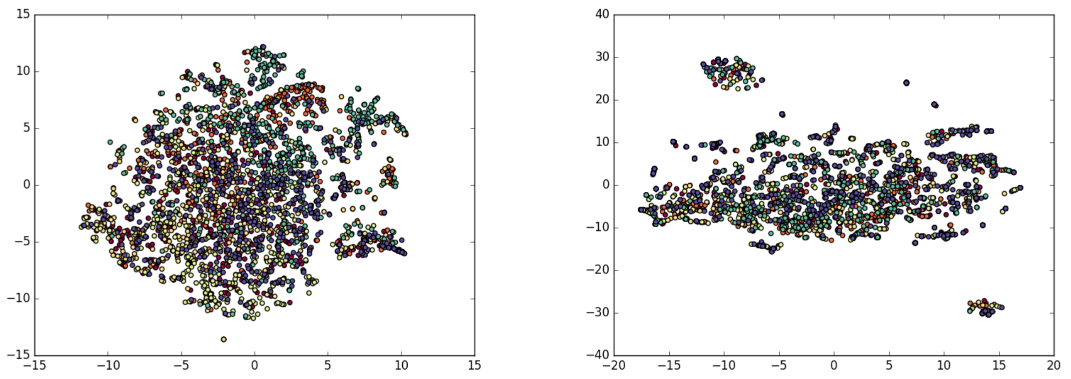
DM/M generated better results, which top 10 related words are synonyms of I. It seemed predictable that DBOW stored less data to train. DM/C and DM/M actually model the word in more proper way though the accuracy their accuracy is not good.

6.5 FastText

In fasttext, we tried to evaluate the 3 different datasets, segmented, non-segmented and pinyin. Although pinyin dataset archieve the same overall accuracy similar to segmented one, the confusion matrix show different tendancy. Despite of the various settings of loss function, window size, both segmented dataset and non-segmented one classified most entries into 2 major classes. While with some specific parameters, the classifier on pinyin dataset can classify the minor class as well.

We tried to validate the property of vectors generated by pinyin dataset with FastText cbow and skip-gram as Table 6.3. It approves that both cbow and skip-gram can generate the pinyin word-embedding efficiently. However, for the non-segmented dataset, the word-embeddings consists vectors of single characters, they are not able use subword information neither.

We tried to use T-SNE to visualize the vector space in Figure 6.1. As we can see, the



(a) segmented dataset, dimesion=200 , window size=15, loss function= hierarchical softmax

(b) pinyin dataset, dimesion=200 , window size=15, loss function= softmax

Figure 6.1: Visualization of vector space for 2 dataset

Table 6.4: words with low norm in 3 dataset in fasttext

dataset	words
segmented	旳, , 硬, 財, 索取, 像, 巾
non-segmented	喝水那么麻, 我好么@瑟小狼狗....etc
pinyin	GG,MK, :, 😈,☀,KG,PS.

vector space with some ambiguous boundary.

We tried to analyze the vectors it generate. The number of vectors generated with 3 dataset with 100,000 posts are 25,743, 3,137 and 22,233 for segmented, non-segmented and pinyin dataset separately. We listed the vectors with lowest norm in Table [**?**]. As the paper indicated that the vector with lower norm, it contains less infomation. Therefore, most of them are stop words. Surprisingly, the non-segmented dataset contains much less words than other 2. Additionally, the most words it contains are like the sentences rather than the terms. It implied that the fasttext itself can not handle segmentation properly, and explains why it performs poorly. For the segmented dataset, the vectors with lowest norm are some rare used words, which can not be modeled properly with low occurance and few neighboring words around. For the pinyin dataset, the vectors with lowest norm are those special abbreviation and icons. Compared with segmented dataset, those rare used word may have the the same pronunciation with other words or can be inferred from subword information.

The other property of fasttext is that it is insensitive to sequence of word order in the sentence.

6.6 Subword Information

According to the paper, the feature subword information can compensate the insufficience of word embeddings. We tried to evaluate if the features work in pinyin as well, which means sentimatics can be inferred from the pronunciation. Although we converted the dataset to pinyin, the accuracy is not significantly different from the original accuracy. Intuitively, pinyin is less readible to native speaker and inreversible to the characters, since the multiple characters own the same pronunciation. Chinese contains less syllables and more homonyms than English do. In the original paper[2], they evaluate the effectiveness in various languages like Arabic, Czech, German, English...etc. All of them belong to phonography, and most languages belong to phonography. While Chinese belongs to logogram specially. So far, we are still hard to quantize the effectiveness conducted in Chinese.

In the [12], it provides similar function to compensate the words absent from training set. It employs similarity of pre-trained word-embedding, rather than the similarity of n-gram characters features. There are also some differnet approaches like morphologically annotated data, whcih were introduced by Cotterell and Schütze (2015). It may be a good topic to evaluate the difference of these approaches.

6.7 Conclusion

We demonstrated the various modern methods on the Chinese corpus, and it indicated that some models like FastText are invariant to language property. In general, most models improve the sematic analysis compared with tranditional TFIDF, and it is more efficient to extract the informance with more dense vectors.

Most methods are developed with English property, so segmentation plays a crucial role to make the Chinese posts look like English. But the segmentation may also contribute something wrong. Though FastText also can be conducted with non-segmented sentences, it performed worse due to improper segmentation.

We can see fasttext demostrate excellent property in both performance including training and testing time and memory utilization. Besides the accuracy of the sematic conversion, both the performance and the efficiency of memory also become the interest of study.

Reference

[1] G. Arevian. Recurrent neural networks for robust real-world text classification. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 326–329. IEEE Computer Society, 2007.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] K. Dashtipour, S. Poria, A. Hussain, E. Cambria, A. Y. A. Hawalah, A. Gelbukh, and Q. Zhou. Multilingual sentiment analysis: State of the art and independent comparison of techniques. *Cognitive Computation*, 8(4):757–771, Aug 2016.

[4] K.-w. Fu and M. Chau. Reality check for the chinese microblog space: a random sampling approach. *PloS one*, 8(3):e58356, 2013.

[5] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013.

[6] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 861–864. IEEE, 2011.

[7] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fast-text.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[8] T. Kenter, A. Borisov, and M. de Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. 2016.

[9] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[10] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[11] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents icml. 2014.

[12] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

[14] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[15] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565, 2014.

[16] D. Vilares, M. Alonso Pardo, and C. Gómez-Rodríguez. Supervised sentiment analysis in multilingual environments. 53, 05 2017.

[17] J. Zhao, L. Dong, J. Wu, and K. Xu. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1528–1531. ACM, 2012.