Data Science Toolbox Assessment 4

Linlan Wang	Xiaomeng Jia	Keyu Long	Yewei Yuan
fa19847	fy19515	xc19821	bf19915

February 2020

Equity

In this experiment, data processing is the foundation and focus, so each of us has completed the exploration of the data processing part by studying and researching the data.
To calculate the eigenvalues, both the CountVectorizer and TfidfVectorizer are common methods. We combined these two methods and classifiers. Here are our respective tasks:

- ☐ Linlan Wang: TfidfVectorize + Naive Bayes
- ☐ Xiaomeng Jia: CountVectorizer + Naive Bayes
- ☐ Keyu Long: TfidfVectorize + SVM
- ☐ Yewei Yuan: CountVectorizer + SVM

Contents

1 Introduction

With the development of the Internet, mobile phones have become an essential part of everyone's life. Mobile phone text messaging is not only an important channel for sending information, but also an important way for people to obtain information. The verification code we use when registering and the confirmation information we receive when making an appointment, need to be passed to us via SMS.
As text messages continue to grow in popularity and importance, there are more and more spams. Spam is a triple threat: [1]

- It often uses the promise of free gifts or product offers to get you to reveal personal information;
- it can lead to unwanted charges on your cell phone bill;
- and it can slow cell phone performance.

Spam is one of the most controversial by-products of the Internet and its proliferation has overwhelmed the entire Internet. We have to spend a lot of time dealing with spam in mailboxes and the development of a practical anti-spam program is undoubtedly an important subject. Based on the **SMS Spam Collection v.1** data set, we analyzes the messages' text and combines the characteristics of the training data set to select an appropriate classification algorithm to classify spam.

2 Data set

2.1 Brief Introduction

The **SMS Spam Collection v.1** is one of the public sets of SMS labeled messages.[2] This data set was composed by **5,572** English, real and non-enconded messages, tagged according being ham or spam and it has been collected for mobile phone spam research.

2.2 Composition

This small corpus has been collected from free research sources at the Internet:

- 425 SMS spam messages were manually extracted from the Grumbletext website.
- 3,375 SMS ham messages were randomly chosen from the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research.
- 450 SMS ham messages were collected from Caroline Tag's PhD Thesis.
- 1,002 SMS ham messages and 320 spam messages were Combined from the SMS Spam Corpus v.0.1 Big.

The table below counts the composition of the data set, giving the amount of samples in each class and the total number of samples.

Table 1: Data composition

File format	Spam	Ham	Total
Plain text	747	4,825	5,572

2.3 Data content

The collection is composed by just one file, where each line has the correct class (ham or spam) followed by the raw message.

ham What you doing?how are you?
ham Ok lar... Joking wif u oni...
ham dun say so early hor... U c already then say...
ham MY NO. IN LUTON 0125698789 RING ME IF UR AROUND! H*
ham Siva is in hostel aha:-.
spam Sunshine Quiz! Win a super Sony DVD recorder if you canname the capital of Australia? Text MQUIZ to 82277. B
spam URGENT! Your Mobile No 07808726822 was awarded a L2,000 Bonus Caller Prize on 02/09/03! This is our 2nd attempt to contact YOU! Call 0871-872-9758 BOX95QU

3 Data Processing

3.1 Stop Words

In information retrieval, we automatically filter out certain words before or after the natural language processing data (or text) in order to save storage space and improve search efficiency. These words are called Stop Words.

Any set of words can be chosen as the stop words for a given purpose. There is no single universal list of stop words used by all natural language processing tools. In general, stop words can be divided into two broad categories.[3] One is the short function words, such as **"the", "is", "at", "which", and "on"**. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as **"The Who", "The The", or "Take That"**. The other one is some of the most common words—including lexical words, such as "want"—from a query in order to improve performance.

	message	stopwords
0	Go until jurong point, crazy.. Available only ...	4
1	Ok lar... Joking wif u oni...	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	5
3	U dun say so early hor... U c already then say...	2
4	Nah I don't think he goes to usf, he lives aro...	5

Figure 1: Stop words of first five raw text

After removing stop words, symbols and numbers, we did frequency statistics for the words in spam and ham, then generated two word clouds. It can be found that in spam, the words "free", "please", "call", "service" and so on appear frequently, which is consistent with our knowledge of spam, as shown in [Figure 2].

Figure 2: Spam's wordscloud after removing stop words, symbols and numbers

In ham, the words "im", "need", "go" and so on appear frequently, which is also consistent with our cognition of ham in daily life, as shown by [Figure 3].

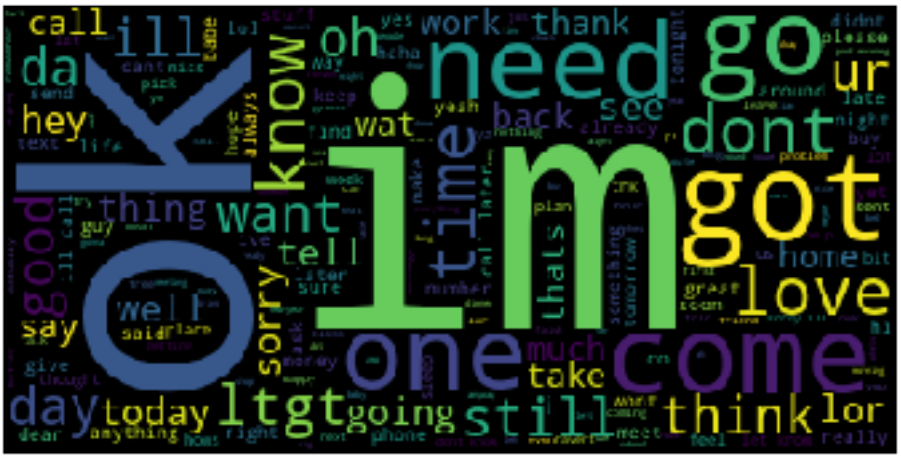

Figure 3: Ham's wordscloud after removing stop words, symbols and numbers

3.2 Stemming & Lemmatization

After removing the Stop Words, we are looking forward to simplify the word form, which is also an important part in data processing. There are two common ways of word simplification, one is stemming and the other is lemmatization.[4]

Stemming is a strictly rule-based process that removes the prefix and suffix of a word to obtain the root. Common prefixes and suffixes are "-s", "-ing", "-ed" and so on. As we can see in [Figure 4], stemming can be basically described as a **"cut"** process.

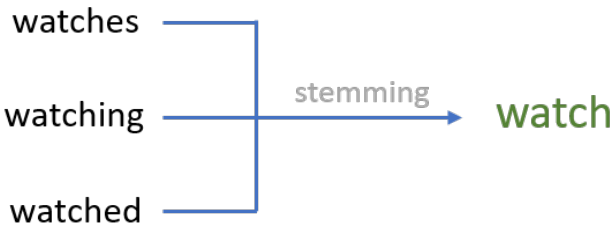

Figure 4: Example of stemming

Lemmatization is based on the dictionary, which transforms the complex form of a word into the most basic form. The lemmatization is not simply removing the prefix and suffix, but converting the word according to the dictionary. More precisely, it is necessary to analyze the form of the word, not only the deletion of the affix, but also the recognition of the part-of-speech(pos), to distinguish the difference between words with the same current form but different original forms. The accuracy of pos tagging also directly affects the accuracy of lemmatization reduction. Compared with stemming, lemmatization is a more complex text processing method.

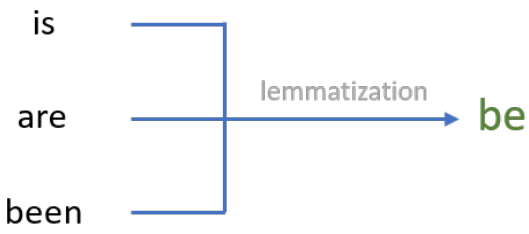

Figure 5: Example of lemmatization

Because the two methods are based on different rules, there are some differences in the outputs of stemming and lemmatization. The result of stemming may not be a complete and meaningful word, but only a part of the word. For example, the result of "revival" stemming is "reviv" and the result of "airliner" stemming is "airlin". On the contrary, the result obtained after the lemmatization is a meaningful and complete word, which is generally a valid word in a dictionary.

3.3 Countvectorizer

CountVectorizer is a commonly used class of feature numerical calculation, which is a text feature extraction method. For each training text, it only considers the frequency of each word in the training text.

CountVectorizer converts the words in the text into a word frequency matrix, which calculates the number of times each word appears through the fit_transform function.

An example sentence [5]: "The weather was wonderful today and I went outside to enjoy the beautiful and sunny weather." You can tell from the output below that the words "the", "weather", "and "and" appeared twice while other words appeared once. That is what Count Vectorization accomplishes.

the	weather	was	wonderful	today	and	I	went	outside	to	enjoy	beautiful	sunny
2	2	1	1	1	2	1	1	1	1	1	1	1

Figure 6: Example of CountVectorizer

3.4 TF-IDF

TF-IDF (term frequency–inverse document frequency) is a commonly used weighting technique for information retrieval and data mining. It is often used to mine keywords in articles, and the algorithm is simple and efficient. It is often used by industry for the first text data cleaning.[6]

We can use TF to filter some high-frequency n-gram(refer to the process of combining the nearby words together for representation purposes where n represents the number of words to be combined together) that do not help the result, such as: and, is, the, etc. When the high-frequency n-grams are filtered, only some n-grams that have practical meaning remain.

However, we will find that some n-grams appear the same number of times, but this does not mean that as keywords, their importance is the same. For some n-grams that are common in other articles, the system will put them under less important. This reflects the importance of IDF(inverse document frequency), IDF will give less weight to common n-grams, and its size is inversely proportional to the commonness of a n-gram.

When we combine TF and IDf, we can get the TF-IDF value of the n-gram. The larger the TF-IDF value of the n-gram in the article, the more important the word is in this article.

In our example, term frequency shows the occurrences of each n-gram in the training SMS messages.We all know that some words are not common in commonly used SMS messages (such as free, urgent, prize, etc.), but these words often appear in spam, so in order to emphasize these words that rarely appear in corpus and often appear in our dataset, we need to downweight the term frequency with inverse document frequency.

TF-IDF formula[7] is :

$$w_{i,j} = tf_{i,j} * \log\left(\frac{N}{df_i}\right)$$

where $w_{i,j}$ = TF-IDF weight for token i in document j, $tf_{i,j}$ = number of occurences of token i in document j, df_i = number of documents that contain token i, N = total number of documents.

Scikit-learn has an off-the-shelf tool called TfidfVectorizer that performs n-gram tokenization and also computes the tf-idf statistic. Two technical details regarding TfidfVectorizer: a) the tf-idf statistic is computed slightly differently to avoid division by zero, and b) the computed tf-idf values for each training example are subsequently normalized.

3.5 Comparison between TF-IDF and Countvectorizer

The choice between CounterVectorize and TF-IDF all depends on what we want to analyze.[5]

If the TF-IDF score is pretty high, it means the words is pretty rare and is good at discriminating between documents. This can be very useful unlike CountVectorizer only counts how many times a word occurs. If your analysis wants to look at the differences between documents, the uniqueness of words then TF-IDF is the tool to use. If you care how frequently a word appears, CountVectorizer is the tool to use.

But in this experiment we found that no matter which classification method is choose, the results obtained using different bag-of-words methods are very similar (we used almost the same preprocessing, control random state = 42, and all passed tuning fit), the performance of TF-IDF is slightly better than Countvectorizor (about 0.2%). We believe that this result is due to the small sample size (about 5500 SMS messages) and the fact that TF-IDF is more practical for SMS processing than Countvectorizor.

4 Classifier

4.1 Naive Bayes

Naive Bayes has been extensively studied in 1950 and it is still a popular (baseline) method for text classification. After proper pre-processing of the text, he can even compete with the more advanced methods (SVM) in the NLP field.

The method used in this project is multinomial Naive Bayes classifier, that is, in the Bayesian formula of this model, we assume a prior that follows a multinomial distribution. The basic Bayesian formula is as follows

$$P(label|feature_1, ..., feature_n) = \frac{P(feature_1, ..., feature_n|label)P(label)}{P(feature_1, ..., feature_n)}$$

What we pay attention in is the numerator equals the joint probability model

$$P(label, feature_1, ..., feature_n)$$

which can be written as,

$$\begin{aligned} P(label, f_1, ..., f_n) &= P(f_1, ..., f_n, label) \\ &= P(f_1|f_2, ..., f_n, label)P(f_2, ..., f_n, label) \\ &= P(f_1|f_2, ..., f_n, label)P(f_2|f_3, ..., f_n, label)P(f_3, ..., f_n, label) \\ &= ... \\ &= P(f_1|f_2, ..., f_n, label)P(f_2|f_3, ..., f_n, label)...P(f_{n-1}|f_n, label)P(f_n|label)P(label) \end{aligned}$$

The notion **"naive"** refers to the assumption that there is strong independence for each feature in the model, and the correlation between features is not considered. Therefore, Naive Bayes is also called simple Bayes or independent Bayes.Under this assumption,

$$P(f_i|f_{i+1}..., f_n, label) = p(f_i|label)$$

Therefore, this joint model can be expressed as

$$\begin{aligned} P(label|f_1, ..., f_n) &\propto (f_1, ..., f_n, label) \\ &= P(label)P(f_1|label)P(f_2|label)P(f_3|label)... \\ &= P(label)\prod_{i=1}^{n} P(f_i|label)P(label) \end{aligned}$$

A well-known application of the Bayesian classifier is the classification of spam emails. By selecting a token (usually a word in the email) to get the association between spam emails and ham emails, Bayesian theorem is used to calculate the probability to classify the emails, which is consistent with the goal of our project.

One of the advantages of Naive Bayes is that it requests only a small amount of training data to make a good estimate of the parameters in the model, and our project also proved that using Naive Bayes classifiers can get quite good classification accuracy. More importantly, under the assumption of strong independence, the computational complexity of the model is greatly reduced, which makes the Naive Bayes classifier show high efficiency in many complex real-world situations.

4.2 Support Vector Machine

Support vector machines (SVM) have been widely used in the field of natural language processing (NLP), such as POS (Part-of-Speech) tagging, word sense disambiguation, NP (Noun Phrase) blocks, information extraction, relationship extraction, and semantic role processing. The above processes take almost the same steps. First, they turned the problem into a multi-class classification task. Then multi-class problems are transformed into several binary classification problems. The SVM classifier is then trained for each binary classification; finally, the results of the classifiers are combined to obtain a solution to the original NLP problem.

In terms of classification, SVM tends to have better inductive ability for invisible data than other distance-based or similarity-based learning algorithms (such as k-nearest neighbors (KNN) or decision trees). Another feature of the SVM is that by using different types of kernel functions, the SVM can explore various combinations of a given function without increasing the computational complexity.[9]

In our NLP tasks usually represent instances with very high dimensional but very sparse feature vectors, which results in ham and spam being distributed in two different regions of the feature space. This is particularly useful for SVM's generalization ability to search classification hyperplanes and classifiers in the feature space. This is the main reason why SVM can achieve good results in various NLP tasks.[8] By using a large number of linguistic features to explicitly form feature vectors from text, and in many cases by exploring so-called kernel functions to map features (in our task, the feature space exceeds 7000) enters a higher-dimensional space.

We've elected to use SVM with a linear kernel because text data contains a large number of features(more than 7000). So using a nonlinear kernel would be computationally expensive. Apart from that, by referring some online materials, we decide to use svm.LinearSVC() instead of svm.SVC(kernel='linear') because the former relies on a library whose algorithmic complexity is $O(n)$ instead of $O\left(n^2\right)$ or $O\left(n^3\right)$, meaning much faster.

Actually, support vector machines construct hyperplanes or hyperplane sets in high-dimensional or infinite-dimensional spaces, which can be used for classification, regression, or other tasks. The hyperplane achieves good separation, because the marginal size usually reduces the generalization error of the classifier.

Given a binary classification training vector, $y \in \{-1, 1\}^n$, SVC addresses the following issues[12]:

$$\min \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

subject to

$$y_i \left(w^T \phi\left(x_i\right) + b\right) >= 1 - \zeta_i, \quad \zeta_i >= 0, i = 1, \ldots, n$$

Its duality is:

$$\min \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

subject to$y^T \alpha = 0, 0 <= \alpha_i <= C, i = 1, \ldots, n$. Here, e is a vector of all 1, $C > 0$ is the upper boundary, and Q is an $n^* n$ positive semi-definite matrix.

$Q_{ij} = y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel function. ϕ maps training vectors to high-dimensional (possibly infinite) space.
The decision function is:

$$\operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

The influence of too many features of text data is also reflected in the influence of adjusting parameters on accuracy. Every small disturbance to parameter C(the penalty parameter) will increase or decrease the accuracy of the result unpredictably

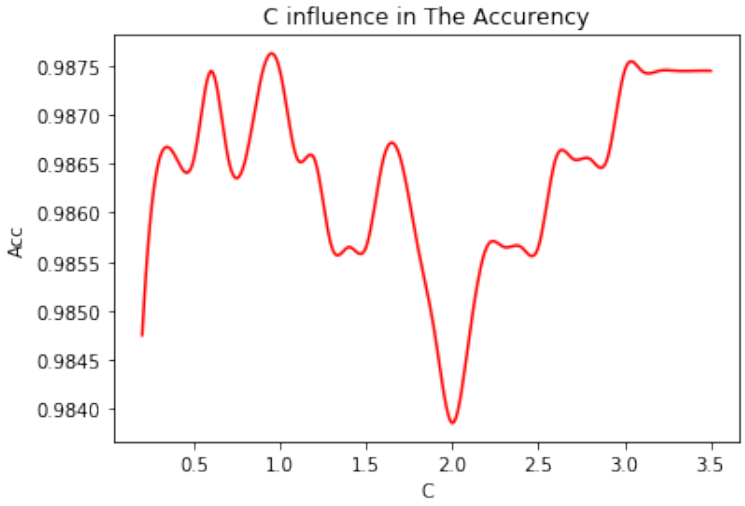

Figure 7: the influence of adjusting parameters

4.3 Comparison between SVM and Naive Bayes

There are many excellent classifiers, such as Random Forest, Bayesian classifier, SVM and so on. In this group task, since the dimension of the data set is very large, and the maximum sustainable dimension of Random Forest algorithm is quite limited, we considered to use Bayesian classifier and SVM to achieve the spam classification.

In terms of Naive Bayes, the algorithm is simple in logic and easy to implement. However, the Naive Bayes assuming that features are independent of each other is often untrue in practice. The greater the correlation between the features, the greater the classification error. Furthermore, multinomial Naive Bayes is usually used in discrete-time models. For example, in spam classification task, we not only check whether words appear in the text, but also count the number of times. If the total number of words is n and the number of a specific word is m, it's kind of like a statistical scenario where a specific event occurs m times in n independent trials.

In general, SVM has a high computational complexity. But because of the limited size of our dataset, and we use a linear kernel, all computations can be done in an instant for the computer, we can't feel the time difference consumed by the two algorithms. In this project, both Naive Bayes and SVM showed great performance, presenting only a 0.5% difference in the accuracy of prediction (SVM was slightly higher than Naive Bayes).

5 Sentiment Analysis

Text sentiment analysis refers to the process of analyzing, processing, and extracting subjective text with sentiment by using natural language processing and text mining techniques[10].

At present, there are two common sentiment analysis methods: sentiment dictionary-based method and machine learning-based method.

Actually, machine learning-based methods require large-scale training sets. Training and prediction are time-consuming and complex and models trained in a certain class of text cannot be generalized in other types of text well. Therefore, in the case of insufficient data and no sentiment annotation, sentiment dictionary is a better choice.

NLTK contains a built-in sentiment analyzer module, **nltk.sentiment.vader**. It can analyze the positive, negative, and neutral polarity classification of emotions in a text or sentence. Among them, "compound" indicates the degree of complexity, "neu" indicates neutrality, "neg" indicates negative emotions, and "pos" indicates positive emotions.

Vader is a method for text emotion recognition based on thesaurus and grammatical rules[11]. Here we briefly introduce the vader thesaurus.

(1) The construction of the vader thesaurus.

- 7000+ common sentiment words (including adjectives, nouns, adverbs, etc.) were manually judged by 10 people to determine the polarity and intensity of emotions.
- Different from other sentiment dictionaries, vader's thesaurus also takes the commonly used characters into account to deal with the sentiment discrimination of non-standard sentences in the network environment such as Twitter. Vader also considers the sentiment of commonly used abbreviations, such as WTF, LOL, etc. and slang, such as nah, giggly, etc.

(2) The impact of vader's grammatical rules on emotional recognition.

- **Punctuation**: Such as "!" will strengthen the emotional intensity of the sentence.
- **Case**: If the sentence contains both uppercase and lowercase words, the emotional intensity of all uppercase words will increase.
- **Degree adverbs**: For example, extreme good is much stronger than good.
- **Conjunctions**: For example, if there exists a transition conjunction "but" in a sentence, the emotional polarity before and after the "but" is reversed. The general intention is to emphasize the semantic emotion after the "but".
- **Negative words**: For example, "isn't" will cause the subsequent emotional reversals.

Through the vader thesaurus, we analyzed 5,572 information's emotional tendencies and made statistics combined with ham and spam.

Table 2: Sentiment classification statistics

sentiment label	neg	neu	pos	All
ham	46	4511	268	4825
spam	0	737	10	747
All	46	5248	278	5572

Through analysis, we found that whether it is spam or ham, neutral emotions are the most and negative emotions are the least. Spam is almost entirely neutral, with no negative emotions. [Figure 8] is the intuitive quantity histogram.

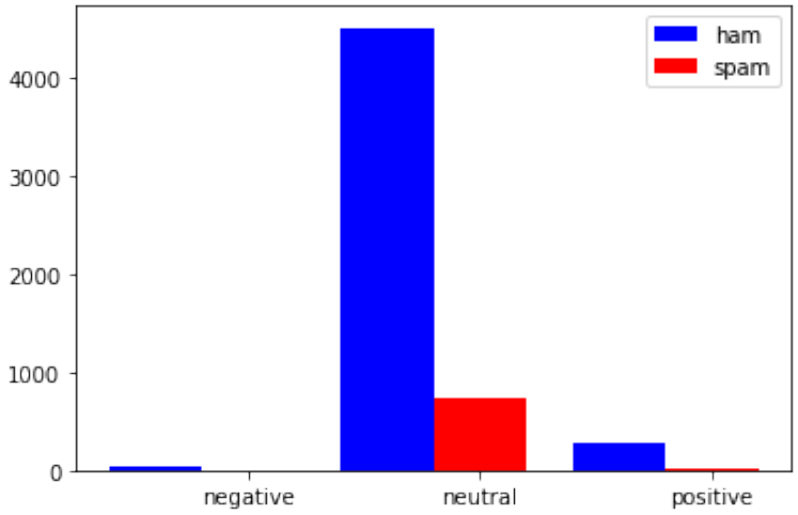

Figure 8: Sentiment analysis histogram

The sentiment distribution of spam and ham in this data set is not much different, and we cannot get accurate classification through sentiment analysis. But sentiment analysis is still a very important part of text analysis.

6 Conclusion

In this experiment, we made a simple attempt on natural language processing. First, we preprocessed 5,572 pieces of SMS text data and extracted features. Then we performed benign and malignant classification and sentiment analysis on the text by using the functions contained in the nltk package in Python. Finally, we compared the differences between the methods we used and visualized some of the results. The whole process is very clear, and this project paved the way for our future research on natural language processing.

In fact, in addition to identifying spam and sentiment analysis, there are many other applications of natural language processing, such as machine translation, information extraction, automatic question answering, and recommendation systems. However, the basic text processing methods are similar no matter where they are applied. Only by continuously improving the text processing method can we have more application directions and higher accuracy.

References

[1] Text Message Spam
https://www.consumer.ftc.gov/articles/0350-text-message-spam

[2] SMS Spam Collection v. 1
http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/

[3] https://en.wikipedia.org/wiki/Stop_words

[4] http://www.chanpin100.com/article/109260

[5] https://medium.com/@joshsungasong/natural-language-processing-count-vectorization-and-term-frequenc

[6] https://zh.wikipedia.org/wiki/Tf-idf

[7] https://towardsdatascience.com/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef677

[8] https://github.com/NLP-LOVE/ML-NLP/blob/master/Machine%20Learning/4.%20SVM/4.%20SVM.md

[9] *Adapting SVM for Natural Language Learning : A Case Study Involving Information Extraction*, Yaoyong Li and Kalina Bontcheva and Hamish Cunningham, 2006.

[10] *Opinion mining and sentiment analysis[J]*, PANG B, LEE L. Foundations and Trends in Informationetrieval, 2008, 2 (12) :130135

[11] *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text[C]*, Hutto C J, Gilbert E. Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media. 2015.

[12] https://www.jianshu.com/p/e3b2b289bd96