Comparison of Preprocessing and Feature Engineering Methods in Text Classification

Christopher Zheng 260760794
Wenwen Xu 260786231
Hongjian Gu 260772647

Abstract

In this work, we focus on the preprocessing and feature engineering step of text classification rather than improving models. We compare various text preprocessing, feature generation and feature selection methods and we evaluate which methods are effective. We present the best combinations of these methods based on our dataset for machine learning and deep learning models. These combinations are expected to be replicable and applicable to other datasets and classification tasks.

1 Introduction

In recent years, text classification is one salient application of machine learning research because of its promising commercial benefits. In fact, machine learning algorithms need to work on data appropriately processed by a set of operations which make assumptions and choices on the inclusion of features in text representations. This procedure is a fundamental step in order for the whole system to obtain good results. A good overview of the steps and the most known algorithms for each step is explained in (Kotsiantis et al., 2006). An interesting observation is that although the preprocessing phase and feature engineering phase play an important role, both are often underestimated.

In this work, we highlight the importance of preprocessing and feature engineering techniques and show how and to what extent they can improve system accuracy. Preprocessing plays a key role in normalizing social media languages where abbreviations, misspellings, emoticons, and the use of out-of-vocabulary words(oov) are too prevalent to affect classification accuracy negatively. In addition, appropriate feature selection methods is also possible to improve the performance both in accuracy and training time consumption. We aim to configure a combination of techniques that works the best on our dataset and can be replicated to similar tasks and datasets. In this case, we use a dataset of Reddit comments with twenty classes for classification. In particular, we summarize five widely-used text preprocessing methods (lemmatization, stemming, removing stop words, removing irregular tokens, and spell checking), three feature generation techniques (CountVectorizer, term frequency–inverse document frequency (TF-IDF), and latent Dirichlet allocation(LDA)), and two feature selection approaches (chi-squared test and principal component analysis (PCA)). To compare and contrast their influences on a model's classification accuracy, we design rounds of extensive experiments. The purpose of this comparison is to evaluate which techniques are effective and what type of combinations are optimal. We also present the reasons why the accuracy improves, by means of a precise analysis of each method.

As the key findings, proper preprocessing and feature engineering can greatly enhance both machine learning and deep learning models, and their performances are comparable on our dataset. For machine learning models, achieving an accuracy of 61% which surpasses that of the baseline by 11%, the best preprocessing-feature combination includes lemmatization and stop-word removal for preprocessing, Latent Dirichlet Allocation for feature generation, and chi-square for feature selection. For deep learning models, with an accuracy of 62%, the best combination consists of lemmatization, stop-word removal and irregular token removal for preprocessing, TF-IDF for feature generation and no feature selection.

2 Related Work

Text classification is a field of natural language processing under intense studies (Manning et al., 1999). One salient application of text classification is *sentiment analysis*, particularly in the domain of movie reviews, product reviews, blogs, etc (Go et al., 2009). Text classification techniques are also applied to fields such as *spam filtering* and *readability assessment*.

Keeping pace with improvements of classification models, the fields of preprocessing and feature engineering also attract attention. In (Sayeedunnissa et al., 2013), the authors investigate a classifier for sentiment analysis in posts from Twitter. By using various preprocessing techniques and applying various feature selection techniques to the Naive Bayes classifier, they achieve reasonably good performance, and they show that Naive Bayes with application of Information Gain measured using chi-square with minimum value of 3 to select high information features, gives accuracy above

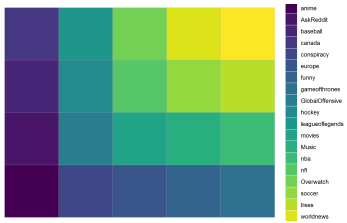

Figure 1: Twenty classes are evenly distributed.

89%. In (Angiani et al., 2016), the authors examine the impacts of preprocessing techniques including basic text cleaning, negation, stemming, lemmatization, etc. They discover that, based on their Tweet dataset, the combination of basic cleaning and stemming with no feature selection yields the best binary classification accuracy. Moreover, the work of A. Balahur (Angiani et al., 2016) also concerns the problem of classification of Twitter posts. She uses a series of interesting preprocessing modules (emoticon replacement, tokenization, punctuation marks, word normalization, etc.). However, the emphasis of her work is not on how each helps improve the accuracy or on the effectiveness of the presented preprocessing techniques.

3 Dataset

Our dataset is gathered from Reddit, a popular online social forum where users post threads. This dataset contains 70,000 threads of comments along with their corresponding subreddit categories. Each category includes exactly 3,500 comments. A comment is a user's typed saying (more or less) about a subreddit theme posted in that subreddit. To avoid overfitting, for each experiment we randomly split the dataset with respect to a 6:1 ratio, leaving 60000 examples for training and 10000 for validation.

4 Approach

As the main goal of this project, we present and explore the pros and cons of a set of preprocessing techniques and feature engineering methods. To compare and contrast their impacts on the final prediction accuracy, we must include certain machine learning classifiers and deep learning architectures which can directly reflect the performance of our methods.

4.1 Methods of Preprocessing

The step of preprocessing decreases the noise level in the comment dataset. By practicing preprocessing, we can better refine the raw data that will be fed to the feature generation procedure.

Lemmatization: Lemmatization refers to preprocessing with the use of morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. Lemmatizers rely on correct language data which makes them considerably accurate. It considers the context of the word to determine what the intended meaning the user is looking for. This process allows to decrease noise. Nonetheless, lemmatization cannot handle unknown words. For example, Porter stems both *iPhone* and *iPhones* to *iPhon* while WordNet lemmatizes both to themselves, respectively. Even for common words, lemmatization can produce weird results as well. Porter stems both happiness and happy to happi whereas WordNet lemmatizes the two words to themselves.

Stemming: Stemming refers to a crude heuristic process that chops off the ends of words and often includes removals of derivational affixes. Stemming shortens the vocabulary space, drastically improving the size of the index (or feature space). Strictly dictionary-based or rule-based stemmers (e.g. Porter Stemmer) are relatively fast compared to lemmatizers. However, stemmers have a much higher error rate (Korenius et al., 2004). Stemming cannot relate words which have different forms from grammatical constructs For instance, *is, am, are* are all conjugations of the same root verb, *be*, but stemming is unable to prune them to a common form. Another example is that the word *better* should be resolved to *good*, but stemmers fail. With stemming, a large amount of ambiguity which may cause several false positives are highly likely to be incurred.

Removing stop words: A stop-word is a commonly used word, such as *the* and *a*, that feature generators should ignore. Stop-words are often useless since they are function words without content-specific information. Thus, removing them can generally reduce the noise level. Conversely, the set of stop words should be carefully scrutinized since some stop words are very suggestive in certain aspects. For instance, negation words, like *not*, are influential in decision making under a classification task and thus removing it negates the meaning of the whole sentence. Moreover, the conventional NLTK stop word set does not include some frequent non-English words such as *la, le* and *les*.

Removing irregular tokens: We consider non-English characters as irregular tokens which contain numbers, punctuations, and other languages. Removing them can generally reduce the noise level. However, certain tokens like Japanese words are in fact suggestive of Japanese-related classes like *anime*.

SpellChecker: We use the SpellChecker that is based on a Levenshtein Distance algorithm to find permutations within an edit distance of 2 from the original word (Perkins, 2010). It then compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list for auto-correction. Running a spell checker ensures that the number of typos can be reduced. Nevertheless, limitations do exist. It is slow. Besides, unless a wrongly-typed word is close to its correct spelling, sensible alternatives are hard to find. Further, a person may correctly spell a word with wrong meaning. Consider the following sentence: *After I had eaten my super I went to bed.* SpellChecker cannot spot that the word *super* should be replaced by *supper*.

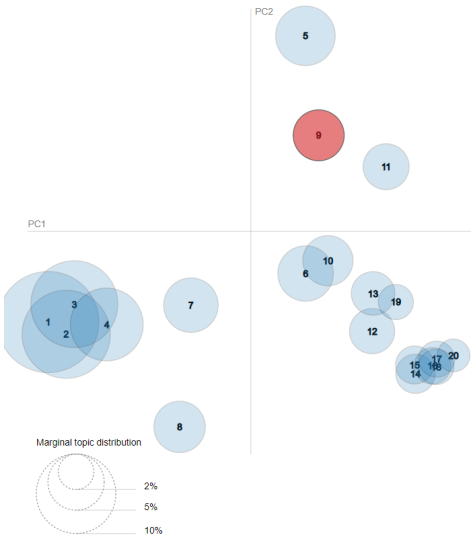

Figure 2: Topics Visualization

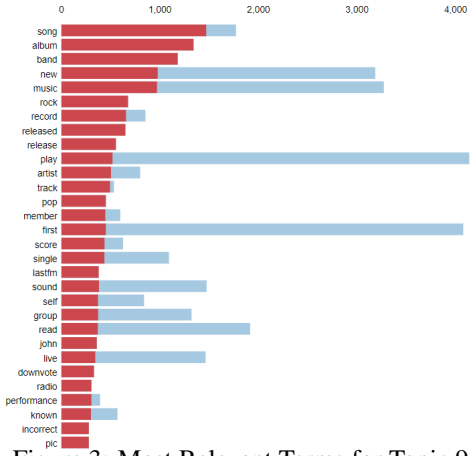

Figure 3: Most Relevant Terms for Topic 9

4.2 Methods of Feature Generation

After cleaning raw data, we generate features that can be used for training by exploring the methods below.

N-grams modeling with CountVectorizer: We can extract feature tokens with CountVectorizer which counts occurrences of words and returns vectors. This leads to a few problems. Common words, like *the*, appear most of the time and other words that carry the topic information of the document become less frequent. Furthermore, the number of occurrences for these non-topic-bearing terms is significantly higher than any other term. This forces them to have the highest weight in the model simply due to their high occurrences and this might skews our model.

TF-IDF transformation: Based on Bag Of Words model(BOW), TF-IDF transformation (Ramos et al., 2003) represents each term with a score rather than its frequency. It balances out the term frequency (how often a word appears in the document) with its inverse document frequency (how often the term appears across all documents in the dataset). It effectively reflects the relevance of a term, however, it inherits the disadvantage of BOW that it cannot capture semantics and co-occurrences in different documents.

Topic modeling with Latent Dirichlet Allocation (LDA): One common disadvantage of using the aforementioned methods as feature representation is that high dimensionality of the feature space may be problematic in terms of time and space complexity and is possible to cause overfitting in later model training. Therefore, we look into LDA(Blei et al., 2003) which is an example of topic modeling and can significantly reduce the dimensionality, increase the speed and improve the efficiency of classification. Instead of representing each comment with a vector with dimension of the whole vocabulary, we can think conversely that building a topic per document model and words per topic model, modeled as Dirichlet distributions.

After training an unsupervised LDA model which recognizes 20 hidden topics, we apply it to each comment and generate a feature vector with exactly 20 dimensions, compared with previously hundreds of thousand dimensions (unigrams). Each dimension of the new feature space corresponds to the probability of a comment to be of one topic. The dimensionality reduction is reflected in the training time for Naive Bayes and Logistic Regression classifiers that are dramatically faster in Feature Generation 2 in Table 2.

To analyze how well LDA works on our dataset, we note that the area of circle in Fig. 2 represents the importance of each topic over the entire corpus, and the distance between the centers of two circles indicates the similarity of two topics. From the most frequent terms in each topic shown in Fig. 3, we can infer the most likely class of it, where the red histogram shows the term occurences in this topic. For instance, from *song, album, band, new, music, rock, record, etc.*, we can infer that this hidden topic is most likely to be *Music*. Also from this graph, we can see some intrinsic problems of the training corpus which affects the performance of methods. The overlaps of some circles indicate that the 20 classes are not independent enough, in other words, the more similar two topics are, the more difficult our classifiers can successfully classify them. For example, the class *league of legends, overwatch* and *wow* are subcategories of *Games*, therefore, they tend to share a lot of semantic similarities related to game playing.

4.3 Methods of Feature Selection

Feature selection is a state-space search where the discrete search space consists of all possible combinations of attributes of the dataset. Feature selection reduces overfitting because less redundant data gives less likelihood to make decisions with noise. It also improves accuracy because of less misleading data. Besides, it saves training time since less feature dimension makes algorithms train faster. For training examples that contain small numbers of features, Pearson correlation analysis is a common choice. In this case, however, the large number of features suggests that we may consider methods such as Pearson's Chi-squared

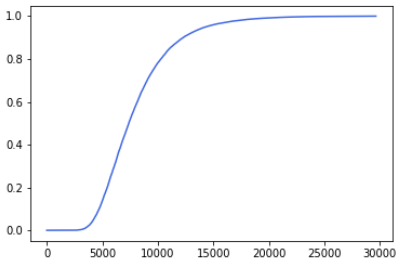

Figure 4: P-value ranking of features. Vertical: p-values. Horizontal: ranking.

test and Principal Component Analysis (PCA).

Chi-square: A chi-square (χ^2) (Lemeshow and Hosmer Jr, 1982) statistic is a test that measures how expectations compare to observed data (or model results). We show this concept by taking a preprocessing result (31539 features) of applying irregular token removal and CountVectorizer as an example. We apply the test, sort the features according to ascending p-values that define rejection regions, and discard the features at the bottom of the ranking as shown in Fig. 4. Specifically, the words ranked after approximately the fifteen thousandth on the list almost get p-values of 1. It is thus reasonable, using the chi-square technique, to delete those words and save only first fifteen thousand tokens.

Principal component analysis: PCA (Jolliffe, 2011) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables. Similarly, the main goal is also to remove correlated features, i.e., tokens. To begin with, we standardize the dataset before implementing PCA, otherwise PCA will not be able to find the optimal principal tokens. After implementing PCA, our original token features are converted into principal components. Principal components are the linear combination of the original tokens. One con is that principal components are not as interpretable as the original features. Since PCA yields very similar results as the chi-square does, we do not present plots due to the limited space.

4.4 Models

Since models are not the focus of this project, we empirically select four widely utilized models (2 ML and 2 DL models) to assess the aforementioned preprocessing, feature generation and feature selection methods by measuring the classification accuracy of these models with our methods. Results are presented in Sec. 5.

Naive Bayes: Multinomial Naive Bayes takes input values that are assumed to be generated with respect to a multinomial distribution. Based on past literature, this event model is typically chosen for document classification (Rennie et al., 2003).

Logistic Regression: Logistic Regression (Walker and Duncan, 1967) estimates parameters of a logit model and classifies examples among classes based on probability. We explicitly denote that the feature value distribution is multinomial to match the real-world scenario.

Table 1: Numbers of unigrams processed.

Preprocessing	Tokens (Unigrams)	Reduction%
No operation	87010	/
Lemmatization	66658	23.4%
Stemming	54040	37.9%
Rm stopwords	86685	0.4%
Rm irregular	31539	63.8%
SpellCheck	70283	19.2%

Convolutional neural network(CNN): We construct a CNN model with five one-dimensional convolutional layers with decreasing numbers of units and followed by five global pooling layers and two dense layers.

Recurrent neural network(RNN): We utilize an LSTM architecture, a special type of RNNs. Our LSTM model has two layers of 128 LSTM units followed by one layer of twenty dense units.

5 Evaluation and Result

In this section, we evaluate the performance of preprocessing methods, feature generation techniques and feature selections approaches. We also present the combination of these methods that performs the best on our Reddit comment dataset. All the results are obtained with 5-fold cross-validation and with GridSearch (Hsu et al., 2003) for fine-tuning [1].

5.1 Preprocessing Evaluation

We first focus on the reduction factors of the preprocessing methods. We tokenize the texts produced by these methods into unigrams and the result is in Table 1. The numbers may convey to what extent the methods are capable of shrinking the feature space, but it is crucial to note that smaller numbers do not necessarily suggest that the corresponding methods are better since they may lose significant features if their shrinking policies are not appropriate. Their impacts on the final classification accuracy are evaluated and discussed in Section 5.3 along with feature generation techniques.

5.2 Feature Generation Evaluation

We design 93 rounds of experimentation. For preprocessing, we exhaust all 31 subsets of the five methods and test them with all three feature generation techniques. (Incompatible combinations are ignored.) We display a representative subset of the experiments in Table 2 where we do not include feature selection. For the preprocessing methods, except for SpellCheck, all have similar performances. We suppose that the autocorrection algorithm mistakenly converts a group of Internet slangs or specific terms into words with other meanings which may explain the overly poor performance. One interesting discovery is that deep learning

[1] Implementation specs: experiments are implemented in Python 3.7.3 and are conducted on Google Colaboratory with 25.5 GB RAM and GPU. Deep learning models are trained for 30 epochs as default and their training time is presented in one epoch.

Table 2: A subset of feature generation evaluation.

Feat. Generation 1	RNN acc,time	CNN acc,time
CountVectorizer	50%, 120s/e	54%, 250s/e
CountV+lemma	53%, 120s/e	55%, 256s/e
CountV+stem	52%, 121s/e	53%, 253s/e
CountV-stopword	52%, 119s/e	55%, 251s/e
CountV-irregular	54%, 120s/e	55%, 249s/e
CountV+Spell	44%, 123s/e	40%, 243s/e
Feat. Generation 2	NB acc, time	LR acc, time
Tf-idfVectorizer	51%, 214s	50%, 792s
Tf-idf+lemma	52%, 256s	52%, 848s
Tf-idf+stem	53%, 285s	53%, 902s
Tf-idf-stopword	53%, 226s	52%, 805s
Tf-idf-irregular	51%, 230s	51%, 799s
Tf-idf+Spell	43%, 2437s	41%, 2996s
Feat. Generation 3	NB acc, time	LR acc, time
LDA	55%, 23s	56%, 34s
LDA+lemma	55%, 24s	57%, 33s
LDA+stem	54%, 23s	54%, 35s
LDA-stopword	55%, 23s	56%, 35s
LDA-irregular	52%, 23s	53%, 33s
LDA+Spell	45%, 1902s	44%, 2701s

models do not show extremely explicit superiority over traditional machine learning models. This may be due to the fact that the Reddit comments are full of informal expressions which make the dataset very noisy.

5.3 Feature Selection Evaluation

Instead of presenting another sheet of result, we assess the two feature selection methods by testing whether they can the experiment results of Table 2. Specifically, we apply either chi-square or PCA to the aforementioned 93 experiments, and we notice that chi-square improves the prediction accuracy of 86% of the experiments whereas PCA enhances only 54% of them. As for the improvements brought by chi-square, the average amount of improvements is approximately 2% while that of PCA is around 1%.

5.4 Best Combinations

After comparing all combinations of preprocessing, feature generation and feature selection, we select two best combinations that yield the best classification results on our dataset. The first combination: we use lemmatization and stop-word removal for preprocessing, Latent Dirichlet Allocation for feature generation, and chi-square for feature selection on a Naive Bayes classifier. Particularly, because of its special mechanism, LDA is applied after chi-square feature selection so that only selected words can be fed into LDA to get the probability distribution of topics. It achieves an accuracy of 61%, the best accuracy among those of all non-deep learning models. The excellence of this combination highlights the positive influence of LDA and that proper preprocessing can provide high-quality information to LDA which pipes generated features to classifiers. Please note that the success of this combination should not be considered as a coincidence as we apply it to Logistic Regression and also gain an accuracy that is very close to 60% (59.6%). The second combination: we lemmatize texts, remove stop words and irregular tokens for preprocessing, and use TF-IDF for feature generation and no feature selection on CNN. It obtains an accuracy of 62%. The success of this combination underscores that feature selection is not very needed for deep learning models. In deep learning, a model can learn by adjusting parameters including weights and bias on its own. Preprocessing, however, still plays an important role in boosting input quality for classifiers as some of the preprocessing techniques remove useless noise in the raw data while others increase the relevance of concepts.

Table 3: Two best preprocessing-feature combinations.

Combination	Model	Acc	Time
lemma,stopword,chi^2,LDA	NB	61%	356s
stopword,irregular,Tf-idf	CNN	62%	269s/e

6 Discussion and Conclusion

To conclude, this work adopts a straightforward method: it basically applies each technique to the raw data. It also assesses every possible subset of the combinations of presented preprocessing, feature generation and feature selection techniques to evaluate their joint contribution. All techniques other than spell checking provide significant improvements to the classifier performances. For text preprocessing, some of the techniques simply remove useless noise in the raw data, while others increase the relevance of some concepts, reducing similar terms and expression forms to their most basic meaning. For feature engineering, all methods are to select fundamental features and delete confounding ones. Having a more precise measure of the impact of these basic techniques can improve the knowledge of the whole data mining process.

This work is conducted over data from Reddit and most of our previous studies are based on Twitter data. A similar analytical work should be performed on different types of datasets to have a more comprehensive understanding of the different preprocessing and feature-related methods. The decision to mix some of these methods is often correct and our presented best combinations are highly recommended for similar text classification tasks. Nevertheless, it should be better motivated by empirical data and result evaluations for various application domains and the peculiar nature of their textual data.

7 Statement of Contribution

Implementation: Christopher takes preprocessing & feature selection. Wenwen takes feature generation. Hongjian takes classifiers. Report: done as a group.

References

Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. 2016. A comparison between preprocessing techniques for sentiment analysis in twitter. In *KDWeb*.

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

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification.

Ian Jolliffe. 2011. *Principal component analysis*. Springer.

Tuomo Korenius, Jorma Laurikkala, Kalervo Järvelin, and Martti Juhola. 2004. Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 625–633. ACM.

SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. 2006. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117.

Stanley Lemeshow and David W Hosmer Jr. 1982. A review of goodness of fit statistics for use in the development of logistic regression models. *American journal of epidemiology*, 115(1):92–106.

Christopher D Manning, Christopher D Manning, and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Jacob Perkins. 2010. *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ.

Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623.

S Fouzia Sayeedunnissa, Adnan Rashid Hussain, and Mohd Abdul Hameed. 2013. Supervised opinion mining of social network data using a bag-of-words approach on the cloud. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, pages 299–309. Springer.

Strother H Walker and David B Duncan. 1967. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.