Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1405 – 1421
© TÜBİTAK
doi:10.3906/elk-1907-46

The impact of text preprocessing on the prediction of review ratings

Muhittin IŞIK[1,*], **Hasan DAĞ**[2]

[1]Department of Computer Engineering, Institute of Science and Engineering, Kadir Has University, İstanbul, Turkey
[2]Department of Management Information Systems, Faculty of Management, Kadir Has University, İstanbul, Turkey

Received: 08.07.2019 • **Accepted/Published Online:** 15.11.2019 • **Final Version:** 08.05.2020

Abstract: With the increase of e-commerce platforms and online applications, businessmen are looking to have a rating and review system through which they can easily reveal the feelings of customers related to their products and services. It is undeniable from the statistics that online ratings and reviews attract new customers as well as increase sales by means of providing confidence, ratification, opinions, comparisons, merchant credibility, etc. Although considerable research has been devoted to the sentiment analysis for review classification, rather less attention has been paid to the text preprocessing which is a crucial step in opinion mining especially if convenient preprocessing strategies are found out to increase the classification accuracy. In this paper, we concentrate on the impact of simple text preprocessing decisions in order to predict fine-grained review rating stars whereas the majority of previous work focused on the binary distinction of positive vs. negative. Therefore, the aim of this research is to analyze preprocessing techniques and their influence, at the same time explain the interesting observations and results on the performance of a five-class–based review rating classifier.

Key words: Text preprocessing, sentiment analysis, opinion mining, review rating, text mining

1. Introduction

Especially over the past decade, fast-growing e-commerce platforms have begun to dominate the entire business world. Thanks to the many options provided by these platforms, customers started to feel more comfortable with e-commerce than with traditional commerce by finding products experienced by others, which are reviewed and rated by many people who are expressing and sharing their own feelings and thoughts about any products. Thus, customers' opinions began to play a major role in purchasing decisions, business intelligence, and keeping any product or service available. Many studies and surveys have been conducted by companies and they have proved that sentiment analysis has been a constantly growing area in recent years [1]. Holleschovsky and Constantinides [1] show that 98% of the sample research population read reviews before making a purchase and 60% of them read often or quite often. Last ReviewTrakers online survey shows that 6 out of 10 consumers search Google for online reviews before visiting a business [2]. Tripadvisor indicates that travelers rely on reviews and opinions from other travelers before booking their trip [3]. Therefore, the field of sentiment analysis, which

[*] Correspondence: mhttnsk@gmail.com

[1] The Amazon Shopper Behavior Study (2018). How shoppers will browse and buy on amazon in 2018 [Online]. Website http://learn.cpcstrategy.com/rs/006-GWW-889/images/2018-Amazon-Shopper-Behavior-Study.pdf [accessed 10 September 2018]

[2] Reviewtrackers (2018). Consumer trends in online reviews [Online]. Website https://www.reviewtrackers.com/online-reviews-survey/. [accessed 19 September 2018]

[3] TripAdvisor (2016). TripBarometer, Traveler Trends & Motivations Global Findings [Online]. Website https://www.tripadvisor.com/TripAdvisorInsights/wp-content/uploads/2018/01/TripBarometer-2016-Traveler-Trends-

is also called opinion mining, suddenly became a popular research field because of the opportunities it provides to the companies wanting to know the pros and cons of their products or services to identify new strategies as well as take crucial decisions.

Specifically, sentiment analysis in reviews is the process of analyzing, monitoring, and categorizing thoughts, opinions, or feelings from an unstructured text about a product or a service, especially in e-commerce platforms. Namely, it works on unstructured review text to find useful information for business intelligence. There are a couple of steps for text classification such as preprocessing, feature extraction, feature selection, and classification.

Although sentiment analysis is a relatively new area of computer science, there are considerable researches except for the importance of text preprocessing on classification performance. Therefore, in this study, we specifically focus on the role of various text-preprocessing stages which are the initial processes in sentiment analysis to demonstrate the effects by experimental results on the performance of a five-class–based review rating classifier. Generally, preprocessing consists of some methods such as tokenization, lemmatization, stemming, lowercase conversion, replacing negation, reverting repeated letters, expanding acronyms, and removing stopwords, numbers, URLs, punctuations, and special characters.

There are few types of research on predicting fine-grained rating stars in review texts which is a challenging task because of the low probability of estimation and use of similar words for closed classes by users. Thus, it is important to know which preprocessing method will increase the classification accuracy and how and why it affects the results.

The rest of this paper is organized as follows. After the introduction, Section 2 presents some of the recent work especially focusing on preprocessing techniques for text classification. In Section 3, we explain some details about each preprocessing method and give some specific examples about the related area. Section 4 introduces a real-life dataset used in our experiment and Section 5 reports some experimental outcomes and evaluates the results. Finally, we conclude and discuss our future work in Section 6.

2. Related work

After sentiment analysis really attracted a great deal of attention among data mining researches in the last decades due to the charming commercial returns, researches related to this field started to increase, particularly on classification models aiming to improve the sentiment classification accuracy. In this section, we specifically focus on some recent related researches which deal with different types of preprocessing methods to improve the performance of a classifier.

When we look at the recent studies in general, some of them indicate that certain preprocessing methods have a great effect on the performance of classifiers while some of them state that they are only slightly better or do not show any effects or even worse. Below is a close look at some of these studies.

Sharma et al. [2] investigated the impact of preprocessing on four different Twitter text data i.e. sports, politics, entertainment, and finance. According to the results, removing stopwords, URL links, and punctuations and converting lowercase increase the classification accuracy of the Twitter sample data.

Ghag et al. [3] investigated the impact of removing stopwords on several sentiment classification models using the movie document dataset. According to the results, while removing stopwords has a great effect on the classification accuracy for the traditional sentiment classifier, there is no significant change for the other classifiers such as the average relative term frequency sentiment classifier, sentiment term frequency, inverse document frequency, and relative term frequency sentiment classifier.

Motivations-Global-Findings.pdf [accessed 15 August 2018]

Jianqiang and Xiaolin [4] investigated the impacts of preprocessing techniques for the performance of sentiment classification on five Twitter datasets. According to the experimental results, while removing URLs, numbers, and stopwords has a little effect, expanding acronyms and replacing negation have a huge impact on the classification accuracy and F1 measure for the classification of Twitter texts.

Srividhya and Anitha [5] investigated some preprocessing techniques to find out whether they have an impact on the classification accuracy on the Reuters dataset. According to the results, removing stopwords, stemming, and TF/IDF have a great effect on the performance of classification.

Camacho-Collados and Pilehvar [6] studied the role of simple preprocessing techniques on the performance of neural text classifier using tokenizing, lemmatizing, lowercasing, and multiword grouping. According to the research, using simple tokenization affects more than complex preprocessing techniques such as lemmatization or multigrouping. The research also shows that the effects of preprocessing changes according to the size of the training data used.

Ghag et al. [7] studied some preprocessing techniques for optimizing sentiment classification. For this purpose, they focus on some rules to handle apostrophe and punctuation symbols, unlike traditional preprocessing techniques. According to the results of the research, accuracy of classification increases by the proposed preprocessed data and elimination of the stopwords decreases, unlike traditional sentiment classification.

Jianqiang [8] studied the preprocessing techniques in order to show their effects on Twitter sentiment analysis especially cleaning tweets from URL links, stopwords, repeated letters, negation, acronym, and numbers. According to the author, some preprocessing techniques hardly change the accuracy of sentiment classification such as removing URL links, numbers, and stopwords.

Safeek and Kalideen [9] studied spell correction and emoticon analysis in order to get suitable data for sentiment analysis on Facebook data. According to the authors, writing “happppyyyyy” is more strength than “happy”. Namely strength of the word is defined how many times a character occurs in a word.

Singh and Kumari [10] studied the effects of preprocessing and normalization on the short text like tweets. They especially evaluate the effects of slang words in a tweet to show how they change the accuracy for a better sentiment classification.

Krouska et al. [11] executed some preprocessing techniques on three different Twitter datasets. According to the results, using appropriate feature selection and representation of the dataset may increase the classification accuracy in sentiment analysis such as 1-to-3 grams perform better than other representations and feature extraction.

Zin et al. [12] showed the effects of various preprocessing strategies such as stopwords, numbers, and punctuations with experimental results on online movie reviews. Their study proved that preprocessing affects the performance of the classification in a good way especially on the SVM with nonlinear kernel.

Pomikálek and Řehůřek [13] studied preprocessing parameters such as stopwords list selection, stemmer selection, and tokenizers in order to compare them on three text data sets and they showed how these parameters affect results. According to their results, the term weighter “ntc” (tf.idf) works best with the shorter documents whereas term frequency “atc” performs better with longer documents.

Schofield et al. [14] investigated the effects of preprocessing in sentiment classification. According to the results, the influence of many common preprocessing techniques such as stemming and removing stopwords have a little effect or even negative effects. They suggest that instead of applying the common preprocessing techniques on the text data, it can be more efficient to decide preprocessing techniques according to the application.

Fan and Khademi [15] concentrate on the effects of top frequent words in raw text reviews and top frequent words/adjectives after part of speech analysis results. According to the results, raw data has almost equal power for different feature generation methods whereas determining words and adjectives after part of speech can remove informative features out.

There are not only researches about text preprocessing in English but also other languages. One of them is the research by Duwairi and El-Orfali [16] who investigated the effects of text preprocessing methods on the classifiers' accuracy in the Arabic language. According to the results, stemming and removing stopwords affect the performance of the classification badly for the movie review texts while slightly improve for the political texts. The other one is Saad's research [17], which investigates the effects of text preprocessing on Arabic text classification applying term weighting schemes, morphological analysis, namely stemming and light stemming. According to the experimental results, light stemming with term pruning work very well for feature reduction and weighting schemes affect the accuracy of the distance-based classifier. As in the Arabic language, there are some challenges in some languages because of having very complex morphology as we compare to the English language. For this reason, preprocessing is very important for text mining. Another study on a language other than English is Uysal and Gunal's [18] research which shows the effects of preprocessing techniques on two different text domains and languages, namely Turkish and English. For this purpose, they use all potential combinations of preprocessing strategies by thinking of several ways. According to the results, using proper combinations of preprocessing strategies provides successful accuracy on text classification depending on domains and languages studied.

3. Background and context

Online review websites play a vital role in all aspects in the business world especially with the increase of e-commerce platforms. Nowadays, most of the review-based e-commerce websites like amazon.com, TripAdvisor, booking.com, and alibaba.com extensively dominate the market. The best parts of these kinds of platforms are that customers can comment on products, rate products, and can easily reach other comments about products written by other users. For this reason, it is important to analyze reviews and ratings in order to determine new strategies and provide better service to customers.

Preprocessing is the first step of the sentiment analysis after getting a dataset. We apply this process to clean and prepare texts for sentiment classification because texts particularly written by users are unstructured. Namely, unstructured texts usually have lots of noisy, unnecessary, useless information such as repeated words, numbers, punctuations, html tags, URLs, scripts, advertisements, stopwords, abbreviations, emoticons, slang words, misspelling, shortcuts, and specific terminology. Because each word is treated as a dimension in the feature set, having unnecessary words causes the models to be confused and loss of time. On the other hand, cleaning the text from noise may increase the performance of classifiers and accelerate the classification process.

Although we cannot show all the details in this research due to the space limitation, preprocessing contains very different steps such as tokenization, removing emoticons, punctuations, and URLs, stopwords elimination, stemming, lemmatization, expanding abbreviation, lowercasing, multiword grouping, word correction, strength of words, weighting scheme and removing common words. Although there have been remarkable researches on this field, finding the best preprocessing method is still an open issue. Researchers show that the best preprocessing methods change according to the application. Therefore, in this study, we concentrate on review texts specifically related to a product or a service. Below are some of these preprocessing methods step by step.

3.1. Tokenization

Tokenization can be defined as splitting up a text into the desired list of practical pieces called tokens such as words, phrases, symbols or other units or even whole sentences in order to work on the text more effective. It is considered an important process of natural language processing because of being an input for the next process. We use whitespace, punctuations and sometimes line breaks to get tokens. For most cases, we use whitespace.

There are a couple of tokenizers in Natural Language Toolkit (nltk) which is a platform to work on human language data. One of them is Regexp tokenizer. This tokenizer split a sentence using regular expression for matching tokens. For instance, if we use RegexpTokenizer$(\text{”}[\backslash w'] + \text{”})$ for a sentence like, “We’ll go on a picnic tomorrow.”. We will get a result like:[“We”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”]. The second one is TreebankWord tokenizer. This tokenizer split the sentence according to the regular expression but treats the punctuations as words, so it splits commas, apostrophes, quotation marks, etc. For instance, if we use TreebankWordTokenizer() for the same sentence above, we will get a result like, [“We”, “ ’ ”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”, “.”]. The third one is WordPunct tokenizer. This tokenizer split the sentence according to this $\backslash w + |[{}^{\backslash}w\backslash s]+$ regular expression. For instance, if we use WordPunctTokenizer() for the same sentence above, we will get a result like, [“We”, “ ’ ”, “ll”, “go”, “on”, “a”, “picnic”, “tomorrow”, “.”]. As it is seen we got the same result as TreebankWord tokenizer for the given sentence. There are more tokenizers in nltk tool to use according to the need.

3.2. Effects of emoticons, removing punctuations, and URLs

Most of the time, it does not make sense to treat emoticons and punctuations as a token for the sentiment classification. Thus, removing emoticons (e.g. :-), :), :-), :-(are frequently used in social media and messaging applications), and punctuations $('!\text{”}\#\$\%'() * +, -./\ :; <=>?@[\backslash\backslash]-\text{‘}\{|\}\ ')$ can increase the accuracy of the classification because of being treated as a dimension in the feature set for each word. However, sometimes especially emoticons can have a slightly good effect on the sentiment score according to the searching area [19]. The research in [20] shows that the importance of emoticons on polarity sentiment classification especially in social networks is undeniable and their popularity is getting higher and higher.

In most literature, URLs do not have any information to analyze regarding sentiments in texts. For instance, when considering the following sentence, “I hate all those disgusting meals from www.mydeliciousmeals.com if you want better you can click www.besteverdinner.com” actually the comment is negative but because of the words of links it may become a positive review. Thus, researchers want to remove URLs from texts to avoid such situations. However, for some specific applications URLs can be effective for providing insights about the text in a way that is not easily obtainable from the context.

3.3. Expanding abbreviations and acronyms

We can say that abbreviation is a shortened form of words and most of the time, their full meaning is given at first mention. We widely use abbreviations to avoid repetition of words that are used many times in a text and to save space. Usually, they are formed by getting first few letters such as Aug. for August, CA for California, and univ. for university. Sometimes they are formed by omitting letters such as TX. for Texas, St. for Street, Rd. for road, and Dr. for Doctor.

The difference between abbreviation and acronym is that acronyms are formed by getting the first letters of each word of the phrase such as ASAP for as soon as possible, PA for personal assistant, lol for laugh out loud, TY for thank you, NP for no problem, FBI for the Federal Bureau of Investigation, and AI for artificial intelligence.

Expanding abbreviations and acronyms are important to understand the contexts in text mining. Compared to the past, the problem of abbreviation and acronym has attracted relatively more attention in text mining especially after increasing the number of messaging applications such as WhatsApp, Viber, Tango, and Line, and social media platforms such as Facebook, Twitter, Instagram, and Snapchat. For instance, the acronyms such as Omg (Oh my God), lol (Laughing out Loud), 2moro (Tomorrow), B3 (Blah, Blah, Blah), ASL (age / sex / location), F2F (face to face), BTW (by the way), XOXOXOX (hugs, kisses,…), PAL (parents are listening), BRB (be right back) are just some of them and they are used very much in daily life conversations. The ability to expand abbreviations and acronyms is crucial for many natural language processing applications and to find out the information contained in documents for information retrieval [21].

3.4. Word correction and multiword expressions

Word correction which is also called misspelling checking is a method that identifies misspelled words in order to change them with the most similar words. For this purpose, the misspelled word is checked whether it is presented in the dictionary or not. If it is not, the algorithm tries to provide the word most similar to it [22]. There are some types of misspelling such as keyboard errors ("yur" – "your", "allways" – "always", etc.), cognitive errors ("piece"–"peice", "sipritual"-"spiritual", "freindly"-"friendly" etc.), phonetic errors ("calander"-"calender", "katalog"-"catalog", etc.), etc.

Bertoldi et al. [23] empirically studied the effects of misspelled words to show the performance of machine translation. According to the research results, performance is related to the noise rate and the noise source affects the capability of machine translation.

There are some tools (e.g., nltk, word2vec, python grammer-check) for checking the misspelled words in texts according to the different languages. Some of them provide the best option, while others offer more than one alternative to the users classifying by types of misspelling. Some of them also check grammatical mistakes by examining everything from incorrect use of a person to subject-verb agreement.

Another important challenge in natural language processing is multiword expressions which are generally difficult to trace from their individual words. They can be metaphorical expressions such as "killing time", "broke someone's heart", and "time is a thief" or verbal idioms such as "give away", "made out", "take off", and "come along with" or phrasal verbs or stereotyped comparisons such "as nice as pie", "swear like a trooper", and "cold as stone" [24], or some well-known group of words such as "United Kingdom", "Galaxy note 9", "Citizen of Humanity". Thus, tokenizing such multiword expressions for text mining causes words to lose their meaning in the sentences. Consequently, getting these types of multiword expressions as a single word can increase the performance of the classifier. There are some studies specifically on this topic [6, 24, 25] to show their effects on text mining. The authors in [25] investigated two empirical methods to integrate multiword expressions in a real constituency-parsing context.

3.5. Stopword elimination

In general, stopwords are the most common words in a languagesuch as "and", "an", "at" in English, which are considered unnecessary and useless in text mining applications. These words can be pronouns (I, me, my, mine, myself, etc.), prepositions (on, in, next to, behind, under, around, etc.), conjunctions (once, until, when, why, since, after, etc.), articles (a, an, the), and auxiliary verbs (be, do, have, will, can, may, etc.). Most of the studies show that stopwords should be removed from the corpus without losing valuable information before the feature selection because of their negative effects on the performance of sentiment classifier. However, sometimes removing the stopwords might reduce the accuracy of classification such as documents or texts related to prepositions, conjunctions, and auxiliary verbs. Thus, removing stopwords can make the matching impossible but as we said, generally due to reducing the size of the feature set comparatively, it has a good effect in text mining. Normally, researchers use compiled lists (Rainbow list, Van stoplist, Smart stopwords list, etc.) provided by text mining tools but sometimes researchers create and then use those predefined lists according to their application. In this study, we use nltk tools but also, we modified the stopwords list according to our text structure.

There are some methods to eliminate stopwords from a text. The basic one is using precompiled lists as we mentioned. The other one is finding the most frequent words which are not needed for matching in the texts. For instance, if you study on restaurant reviews, the word “food” or “meal” generally will not give meaningful results because these words are used in both negative and positive reviews. Maybe these words can be used in combination with other words, namely bigrams or Ngrams combinations such as “terrible tasting”, “food tastes bad”, “never had a bad meal”, etc. Actually, this is another preprocessing method but for some researchers it can be under the branch of stopword elimination. Another method is selecting words that occur rarely and are not related to your texts. There are some additional methods that are examined and studied in the following subsections.

3.6. Stemming

The aim of the stemming is to take words as they occur in a text to reduce them to root forms by removing their affixes such as prefixes (cutting off the beginning of the word) and suffixes (cutting off the end of the word) according to some grammatical rules. In this way, they can be used as an indexing unit in the related research area. Although stemming algorithms in most application tools are commonly developed for English, there is a need for appropriate editing according to the language being studied because of differences in language structure. Nowadays, many different algorithms can be also used for some other languages.

Stemming is applied for a couple of reasons. One of them is to reduce the derivatives of the same root words to the common representations to increase the performance of classification. One of the other reasons is to reduce the size of the feature set so that the number of dimensions is reduced.

We can apply stemming to derivatives of the word such as number (cat, cats), tense (play, played, playing), gender (actor, actress), pronouns (I, me, my, mine), person (hate, hates), and aspect (become, became). For instance, the words select, selected, selecting, selects all can be stemmed to the word “select”. As it is seen, we cut off the end of the words which are semantically related to their root form. In this way, we reduce the number of words in memory space and save time.

When applying stemmer, we should consider some points which are important and required for a powerful natural language processing application. One of them is overstemming which occurs when the words have the same root but not have the same meaning. For example, “general” and “generation” can have the same root “gener”. Similarly, “organization” and “organs” have the same root “organ” and this situation decreases the accuracy of the classifier. Another is understemming which occurs in some stemmer algorithms. For example, the stemmer takes the words “cooks” and “cooked”, which can be reduced to “cook”, while “cookery” can be reduced to “cookeri”, or “absorbtion” and “absorbing” are stemmed to “absorpt” and “absorb”. This causes corresponding documents to not be returned.

Natural Language Toolkit platform uses a couple of stemmers such as PorterStemmer, LancasterStemmer, RegexpStemmer, and Snowball Stemmer. For example, while the PorterStemmer reduces the word “cookery” to “cookeri”, LancasterStemmer reduces to “cookery”. However, if you want to use RegexpStemmer, you should determine your affix. For instance, when you use RegexpStemmer(‘ing’), it brings the word “cooking” as “cook” but you need to be careful in case the word has a prefix such as the word “ingrain”, it will be returned as “rain”.

3.7. Lemmatization

Both stemming and lemmatization are language preprocessing methods to provide that different versions of a word are not left out. Although they are closely related to each other, lemmatization is more complex than stemming because it reduces derivationally related word form to its dictionary form categorized by a part of speech as well as by inflected form. Namely, stemming is applied without checking the position of the word in the sentence. Thus, if the user queries the plural and singular form of a word such as “mice” and “mouse”, when the stemmer brings them as “mice”/“mic” and “mous”, the lemmatization brings “mice” and “mouse” both as “mouse”. For lemmatization we need to indicate the position of the word otherwise lemmatization gets the position of the word as “noun” by default. For example, when you use a WordNet lemmatizer like, wordnet_lemmatizer.lemmatize(‘are’). Lemmatizer will bring it as “are”. For that we should write as below, wordnet_lemmatizer.lemmatize(‘are’, pos=’v’) Then, the lemmatizer will bring it as “be”. Let us have look at some words results after executing lemmatization.

As it is seen in Table 1, according to our indication lemmatizer finds a base form of the words. Namely, lemmatizing means that converting the word to its dictionary form or morphologically related form. For example, for the sentence like, ”I loved cats, dogs, frogs, and geese secretly”. Lemmatizer will bring it as, “i love cat dog frog and goose secretly”. We informed the lemma function that “love” is a verb, “cats”, “dogs”, “frogs”, “geese” are noun and “secretly” is an adverb, and then we get the above result, whereas if we use stemming for the sentence, it will return as, “I lov cat dog frog gees secret”. Namely, stemming returns the root of the word, whereas lemmatizing returns dictionary form according to the position of the word in the sentence.

Table 1. Difference between stemmer and lemmatizer.

Words	Porter stemmer	WordNet lemmatization (pos=verb)	WordNet lemmatization (pos=noun)
constructing	construct	construct	constructing
decided	decid	decide	decided
took	took	take	took
clearly	clearli	clearly	clearly
is	is	be	is

3.8. Lowercasing

Lowercasing is one of the first stages of preprocessing for text mining. All letters are converted to the lowercase to prevent case sensitivity. In this way, we can increase the performance of the classifier without considering nonconsistence of texts. Even though this simple preprocessing technique provides the easiest and important help to the classification, sometimes doing this might create some problems by increasing ambiguity. For example, Turkey is a country, but turkey is an animal or Opera is a browser, but opera is a musical play, so getting words in the lower case would cause these types of words to be considered identical entities in text classification.

3.9. Removing common words

Removal of the common words does not guarantee that the accuracy of classifier will be higher, but for most applications, it gives very good results. Common words and stopwords should not be confused with each other. Stopwords can be the most common words but when we say common words it means that they are found in almost each different class documents related to the field studied. Thus, stopwords are almost the same for all the studied field while common words are totally different for each studied field. For instance, the words “meal”, “dinner”, and “menu” can be the most common words for a restaurant corpus while the words “room”, “reception”, and “bed” can be the most common words for a hotel corpus. As you realize, these words are not enough to find the differences between hotel or restaurant rating classes.

3.10. N-gram

Recently, many researches on text mining and natural language processing have focused on n-gram. According to n-gram, it is not a coincidence that the words in a text are found more than once together. In other words, these words together give us a clue about the text summarization. In particular, the surprising effect of the text classification has been proven by many researches. Of course, n-grams’ effect changes from research to research. For instance, according to the results of [26], n-gram works better on the shorter texts since the presence of words in shorter texts are more important than longer texts. Namely, the value of a word loses its significance or value in a long text. What exactly do we mean by n-gram? Let us explain n-gram with an example. If we want to use bigram for a sentence like, ”An n-gram is a contiguous sequence of words in a text”, after removing the stopwords, the output of the program will be as, [(‘ngram’, ‘contiguous’), (‘contiguous’, ‘sequence’), (‘sequence’, ‘words’), (‘words’, ‘text’)].

As it is seen, n-gram takes each word with the adjacent word. According to the frequency of these adjacent words, the content of the text is estimated, and the classification is made according to the result. If we had done the same example for trigram, we would get an output like, [(‘ngram’, ‘contiguous’, ‘sequence’), (‘contiguous’, ‘sequence’, ‘words’), (‘sequence’, ‘words’, ‘text’)]. As it is seen, at this time n-gram takes three consecutive words and again, according to the frequency of these adjacent words, the class of the text is decided. According to the studied field, the number of the N can be changed but, in this study, we use the combinations of the n-grams from one to three.

4. Dataset description

In this paper, the dataset we use to evaluate the preprocessing methods is a real e-commerce dataset extracted from Yelp in June 2018. It is available at https://www.yelp.com/dataset. We work on two datasets. The first one contains full review text data including the user_id of the person who wrote the review and the business_id for which the review is written and the second one contains business data including location data, attributes, and categories.

The Review dataset contains business_id, the date of the review, Review_id, star (between 1 and 5), review text and user_id. There are 229,907 reviews, from which 43,873 users have at last one review. Table 2 shows that the user rLtl8Z... has a review with id fWKv... and gives 5 stars for the business 9yKzy... on the date 2011-01-26. It is a .json file and 206.0 Mb.

The Business dataset shows business location data, category, its name, how many reviews it gets from users, and its average star rating. There are 11537 businesses which have at last one review. Table 3 shows the business qarob... which reviewed by 10 people and got 3.5 star on average and it is in the Restaurants category. It is a .json file and 4.08 Mb. Our algorithms are executed on Jupyter Notebook with python version 3.6.5.

Table 2. Apperance of the review dataset.

business_id	Date	Review_id	Star	Text	user_id
9yKzy...	2011-01-26	fWKvX8...	5	My wife took me here on my birthday...	rLtl8Z...

Table 3. Apperance of the business dataset.

business_id	Categories	Address	Name	Review count	Star
qarob...	[Restaurants]	891 E Baseline RdSuite...	Jersey Mike's Subs	10	3.5

5. Experimental testing results

In this section, we carry out several experiments in order to verify the effects of preprocessing methods on the performance of the classifier. For this purpose, we perform ten different methods, i.e. tokenization, effects of emoticons, removing punctuation and URLs, expanding abbreviation and acronyms, word correction and multiword expressions, stopwords elimination, stemming, lemmatization, lowercasing, removing common words and lastly n-gram effects.

In order to get meaningful results, we created our feature set on 10,000 restaurant reviews. Our aim is to analyze the effects of preprocessing methods when finding the star ratings of restaurants by analyzing the reviews. Star ratings range from 1 to 5. As we mentioned above, the biggest challenge is to find the star rating of close categories because of using very similar words.

We use K nearest neighbors (KNN), decision tree (DT), random forest (RF), logistic regression (LR), stochastic gradients descent (SGD), naïve Bayes classifier (NB), support vector machine (SVM) classifiers of nltk to get accuracy results for all types of preprocessing methods. Over the created feature set, 10,000 reviews are applied for training the classifier (classifiers are applied for each rating category in equal number, namely, 2000 training reviews are selected from each rating category.) and 1000 reviews are applied for testing (200 testing reviews are selected from each rating category) in order to see the performance of the classifier.

To get stable results from the effects of a random selection of reviews, we run each experiment by selecting 20 times shuffled reviews for each preprocessing method. Namely, for each run, a different subset of reviews is selected from the pool of all available five rating categories.

5.1. Performance of the classifiers based on different tokenizers

In this part, we report the results obtained after tokenizing the text of the reviews as we mentioned above. In addition to those techniques, we also used WhitespaceTokenizer which is simply used for tokenizing the text according to the white space between the words. WhitespaceTokenizer method can be considered as the result without any preprocessing. In other words, at least this basic tokenizer form must be applied on the data before the other preprocessing method can be performed. Therefore, for the rest of the preprocessing methods, we choose space tokenizer to create a dictionary with the 1-to-3 n-grams in order to get directly simple effects of the methods.

As shown in Table 4 after running the methods 20 times shuffled reviews, there is no significant difference between RegexpTokenizer(”$[\backslash w'] +$”), TreebankWordTokenizer(), and WordPunctTokenizer()tokenizers, all the tokenizers gave almost the same results. However, we observed that the WhitespaceTokenizer() gives a little bit worse results on general average all the times than others when we run the program multiple times.

Table 4. Performance of the classifiers based on different tokenizers.

Tokenizer	KNN	DT	RF	LR	SGD	NB	SVM
WhitespaceTokenizer() - Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
RegexpTokenizer(”$[\backslash w'] +$”)	0.284	0.318	0.404	0.513	0.486	0.512	0.478
TreebankWordTokenizer()	0.281	0.303	0.366	0.510	0.485	0.518	0.465
WordPunctTokenizer()	0.266	0.338	0.403	0.503	0.478	0.518	0.476

5.2. Classifier performance based on replaced emoticons and removed punctuations

In this part, we report the results obtained after removing emoticons and punctuations separately. We do not apply the removing URLs method due to a lack of the remarkable number of reviews as we observed from review text.

To see the effects of emoticons on the rating stars, we investigate the usage of emoticons in the text reviews. For this purpose, we create a simple word replacer in order to change emoticons to words. In this way, our program can find a relationship with the combination of words. For example, we replace the emoji “:)” as a “smile”, “:(” as a “sad”, or “:-o” as a “surprised”, which are commonly used in messaging applications and the social media.

As shown in Table 5 after running the methods 20 times shuffled reviews, the average accuracy result of replaced emoticons is sometimes worse than without the execution of the method if we compare with the results base form without preprocessing. Actually, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews unlike the effects in messaging applications and social media as proved in some researches. Almost the same result for the removing punctuations even after we execute multiple times for each method.

5.3. Classifier performance based on expanding abbreviations and acronyms

In this part, we try to find some abbreviations and acronyms in the reviews of the restaurants but unfortunately, there is no remarkable number of the most common words of this specific field in the text of the reviews. Thus, we use general abbreviations and acronyms which are the most commonly used in messaging applications and social media. For example, omg (oh my god), lol (Laughing out Loud), 2moro (tomorrow), or B3 (blah, blah, blah), which we got from https://www.smart-words.org/abbreviations/text.html. Again, we use a replacer class for this purpose and choose space tokenizer for the review text in order to get direct simple effects of the method.

The accuracy results are 0.263 (KNN), 0.305 (DT), 0.393 (RF), 0.484 (LR), 0.470 (SGD), 0.498 (NB), 0.453 (SVM) after running the methods 20 times shuffled reviews, the average classifiers accuracy results of expanding abbreviations and acronyms are not better than without execution of the method significantly compared to the results with the base form given Table 4. Actually, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews unlike the effects in messaging applications and social media as proved in some researches.

Table 5. Classifier performance based on replaced emoticons and removing punctuations.

Preprocessing Methods	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Replaced Emoticons	0.281	0.311	0.341	0.474	0.428	0.492	0.438
Removing Punctuations	0.276	0.289	0.379	0.474	0.435	0.491	0.436

5.4. Classifier performance based on word correction

In this part, we report the result obtained after executing the autocorrector of python on each review text for the misspelled words in order to change with their most possible similar words. As we mentioned above, misspelled words are checked according to the English language. We do not apply the removing Multiword Expression method due to a lack of the remarkable number of reviews as we observed from review text.

The accuracy results are 0.243 (KNN), 0.254 (DT), 0.321 (RF), 0.412 (LR), 0.393 (SGD), 0.421 (NB), 0.401 (SVM) after running the methods 20 times shuffled reviews, the average accuracy result of the word correction is much worse than those of other methods. As we observed from the output of the program, the tool which we used is not successful at all for the text of the reviews. Consequently, this result shows us there are no significant effects of this method for categorizing the rating stars of the reviews unlike the effects in messaging applications and the social media as proved in some researches.

5.5. Classifier performance based on stopword elimination

In this part, we report the result obtained after removing the stopwords in each review text using nltk stopwords. As we mentioned above, stopwords are checked according to the English language.

As shown in Table 6 after running the methods 20 times shuffled reviews, the average accuracy results of the stopword elimination is better than those of other methods especially the modified stopwords list (like not removing comparative adverbs such as good, better, best) method according to our text structure is slightly better than directly removing stopwords. As we observed from the output of the program after executing numbers of times, we conclude that this method has a significant effect on the categorizing the rating stars of the reviews.

Table 6. Classifier performance based on stopword elimination.

Preprocessing methods	KNN	DT	RF	LR	SGD	NB	SVM
Base form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Removing stopwords	0.261	0.379	0.389	0.494	0.454	0.512	0.462
Removing stopwords with modified list	0.299	0.293	0.406	0.494	0.461	0.523	0.486

5.6. Classifier performance based on stemming

In this part, we investigate the stemming algorithms such as Porter Stemmer, Lancaster Stemmer, and Snowball Stemmer and their efficiencies on the restaurant reviews. We reduce words to root forms by removing prefixes and suffixes according to some grammatical rules of the nltk stemmers. We execute the stemmer algorithms on space tokenizer base form in order to get directly simple effects of them.

As shown in Table 7 after running the methods 20 times shuffled reviews, all the stemmer algorithms slightly change the average accuracy results, especially for logistic regression and naive Bayes classifiers in a good way. In general, these results show us that there are no significant effects of these methods on categorizing the rating stars of the reviews as we expected.

Table 7. Classifier performance based on stemming.

Stemmer	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
PorterStemmer()	0.248	0.297	0.376	0.512	0.459	0.511	0.463
LancasterStemmer()	0.232	0.297	0.373	0.501	0.453	0.498	0.448
SnowballStemmer()	0.254	0.316	0.363	0.499	0.461	0.517	0.450

5.7. Classifier performance based on lemmatization

This time we investigate effects of the lemmatizer on the restaurant reviews. Again, we execute the lemmatizer algorithms on space tokenizer base form in order to get directly simple effects of them.

As shown in Table 8 after running the methods 20 times shuffled reviews, the lemmatizer without indicating the word position does not change the accuracy results significantly but the lemmatizer with position increases the accuracy results even we execute the program multiple times. As we observed from the output of the program, these results show us there are no significant effects of this method for the categorization of the rating stars of the reviews as we expected but indicating the position of the word for the lemmatizer gives us better results.

5.8. Classifier performance based on lowercasing

In this part, we report the result obtained after executing the lowercasing on each review text to increase the performance of classifier without considering nonconsistence of texts. We execute our lowercasing 20 times shuffled reviews and on space tokenizer base form in order to get directly simple effects of the method.

Table 8. Classifier performance based on lemmatization.

Lemmatizer	KNN	DT	RF	LR	SGD	NB	SVM
Base Form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
WordNetLemmatizer()	0.263	0.309	0.373	0.473	0.421	0.479	0.439
WordNetLemmatizer() with position	0.305	0.349	0.382	0.513	0.467	0.524	0.481

This time the accuracy results are 0.314 (KNN), 0.333 (DT), 0.399 (RF), 0.523 (LR), 0.473 (SGD), 0.522 (NB), 0.497 (SVM) and surprisingly the average accuracy result of the lowercasing method is much better than before compared to the results with the base form given Table 4. As we observed from the output of the program after executing multiple times, this result shows us that there are significant effects of this method for categorization of the rating stars of the reviews. Because of treating each word as a dimension in the feature set, having the same words in different cases cause the models to be confused and loss of time.

5.9. Classifier performance based on removing common words

In this part, we report the result obtained after removing the common words on each review text to increase the performance of the classifier. We execute the algorithm on 20 times shuffled reviews and on space tokenizer base form in order to get directly simple effects of the method.

This time the accuracy results are 0.294 (KNN), 0.324 (DT), 0.374 (RF), 0.500 (LR), 0.471 (SGD), 0.518 (NB), 0.458 (SVM) and the average accuracy result of the removing common words method is much better than before if we compare to the results with the base form given in Table 4. As we observed from the output of the program after executing multiple times, this result shows us that there are significant effects of this method for the categorization of the rating stars of the reviews. Because the classifier confuses the class of rating when it sees those common words in the review text.

5.10. Classifier performance based on n-gram

In this part, we report the result obtained after executing some combinations of n-gram. In the beginning, we apply each n-gram alone, and then we apply a combination of three in order to see the effect of each combination. Same as before, each obtained result is the average of 20 times shuffled restaurant reviews.

As shown in Table 9, we observed from the output of the program after executing multiple times, while the effect of the Bigram is bigger than Unigram, the effect of the Unigram is bigger than Trigram. When it comes to the combination of the n-grams, the effect of the combination Unigram() & Bigram() is more than Bigram() & Trigram() while the effect of the Bigram() & Trigram() is more than Unigram() & Trigram(). We get the best result even after executing multiple times when we apply all the n-grams together.

Table 9. Classifier performance based on n-gram.

N-Gram	KNN	DT	RF	LR	SGD	NB	SVM
Unigram()	0.304	0.332	0.364	0.370	0.371	0.365	0.361
Bigram()	0.332	0.403	0.415	0.426	0.398	0.445	0.435
Trigram()	0.255	0.297	0.307	0.336	0.331	0.344	0.305
Unigram() & Trigram()	0.325	0.325	0.365	0.388	0.371	0.389	0.377
Unigram() & Bigram()	0.324	0.387	0.411	0.486	0.455	0.483	0.458
Bigram() & Trigram()	0.219	0.362	0.406	0.469	0.432	0.458	0.445
Unigram() & Bigram() & Trigram()	0.335	0.420	0.425	0.495	0.469	0.511	0.472

5.11. Classifier performance based on preprocessing order

In this part, we report the results obtained after executing some combinations of preprocessing methods in order to see effects of executing order. For this purpose we use lemmatization, stopwords, and lowercasing preprocessing methods which have a positive effect on chosen classifiers on the review data set as we mentioned above. In order to see the difference between preprocessing orders we execute all the combination of three methods, respectively. This time we do not shuffle the review set to see the effects of executing order of three methods on the same dataset.

As shown in Table 10, we observed from the output of the program after executing multiple times, executing order of the preprocessing methods affect the accuracy results of any classifier by almost 2%. In addition, when we compare the accuracy results of each classifier with the base form, the preprocessing methods applied change the accuracy results up to 10% in some classifiers. These results show us how important applying preprocessing methods are when classifying our data.

Table 10. Classifier performance based on preprocessing order.

Order of the methods	KNN	DT	RF	LR	SGD	NB	SVM
Base form	0.260	0.287	0.374	0.476	0.446	0.491	0.444
Lemmatization - stopwords - lowercasing	0.345	0.394	0.439	0.553	0.512	0.564	0.521
Lemmatization - lowercasing - stopwords	0.358	0.398	0.423	0.535	0.514	0.547	0.524
Stopwords – lemmatization - lowercasing	0.360	0.376	0.423	0.567	0.522	0.567	0.536
Stopwords – lowercasing - lemmatization	0.344	0.371	0.448	0.542	0.512	0.570	0.527
Lowercasing - stopwords - lemmaztization	0.339	0.404	0.445	0.547	0.494	0.561	0.514
Lowercasing –lemmaztization - stopwords	0.344	0.391	0.433	0.545	0.507	0.542	0.515

6. Conclusion and future work

This paper discussed the experiments involving some simple text preprocessing methods that give an impact on the classification performance when we predict fine-grained review rating stars. For this reason, we wanted to show their effects on the five-class–based review rating stars, individually.

Although less attention has been paid to the text preprocessing in the researches, our evaluations highlight that it has a remarkable impact on the performance of classifier but of course not for all the methods. Some of them have a positive effect on classification accuracy, while some have a negative effect, and others have a neutral effect.

In general, a simple stopword elimination, lowercasing, removing common words, and lastly the combination of 1-to-3 n-grams perform better than other preprocessing methods for improving the classification accuracy of the five-class–based review rating stars. As we mentioned before, the challenge of this field is to predict fine-grained review rating stars because of being used almost the same words for the close classes. Otherwise, it might be useful to apply mentioned methods, for instance, for the binary distinction of positive vs. negative. Namely the effects of the preprocessing methods can change on any domain. Thus, all possible preprocessing methods and their combinations should be considered before used in any application. The order of applying the preprocessing methods can also be important. The effects of abbreviations, acronyms, stemming, and lemmatization might be higher after executing lowercasing to the text. It is believed that our study results will help future researchers to carefully select these text preprocessing methods. Finally, as future work, we plan to work on comparing some different algorithms such as top frequent words and generating vectors for text to finding similarities. to increase the accuracy result of the five-class–based review rating stars.

References

[1] Holleschovsky N, Constantinides E. Impact of online product reviews on purchasing decisions. In: Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016); Rome, Italy; 2016. pp. 271-278.

[2] Sharma P, Agrawal A, Alai L, Garg A. Challenges and techniques in preprocessing for twitter data. International Journal of Engineering Science and Computing 2017; 7 (4): 6611-6613.

[3] Ghag KV, Shah K. Comparative analysis of effect of stopwords removal on sentiment classification. In: IEEE International Conference on Computer, Communication and Control; Indore, India; 2015. pp. 1-6.

[4] Jianqiang Z, Xiaolin G. Comparison research on text pre-processing methods on twitter sentiment analysis. IEEE Open Access Journal 2017; 5 (1): 2870-2879. doi: 10.1109/ACCESS.2017.2672677

[5] Srividhya V, Anitha R. Evaluating preprocessing techniques in text categorization. International Journal of Computer Science and Application Issue 2010; 47 (11): 49-51.

[6] Camacho-Collados J, Pilehvar MT. On the role of text preprocessing in neural network architectures. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP; Brussels, Belgium; 2018. pp. 40–46.

[7] Ghag K, Shah K. Optimising sentiment classification using preprocessing techniques. International Journal of IT & Knowledge Management 2015; 8 (2) : 61-70.

[8] Jianqiang Z. Pre-processing boosting twitter sentiment analysis? In: IEEE International Conference on Smart City/SocialCom/SustainCom 2015; Chengdu, China; 2015. pp. 748-753.

[9] Safeek I, Kalideen MR. Preprocessing on facebook data for sentiment analysis. In: Proceedings of 7th International Symposium on Multidisciplinary Research for Sustainable Development; Oluvil, Sri Lanka; 2015. pp. 69-78.

[10] Singh T, Kumari M. The role of text pre-processing in sentiment analysis. In: Twelfth International Multi-Conference on Information Processing (IMCIP-2016); Procedia Computer Science; Nice, France; 2016. pp. 549-554.

[11] Krouska A, Troussas C, Virvou M. The effect of preprocessing techniques on twitter sentiment analysis. In: 7th International Conference on Information, Intelligence, Systems & Applications; Chalkidiki, Greece; 2016. pp. 740-752.

[12] Zin H, Mustapha N, Murad M, Sharef N. The Effects of pre-processing strategies in sentiment analysis of online movie reviews. In: The 2nd International Conference on Applied Science and Technology; Kedah, Malaysia ; 2017. pp. 4575–4587.

[13] Pomikalek J, Rehurek R. The influence of preprocessing parameters on text categorization. International Journal of Applied Science Engineering and Technology 2007; 1 (9): 54-57.

[14] Schofield A, Magnusson M, Thompson L, Mimno D. Understanding text pre-processing for latent dirichlet allocation. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics; Valencia, Spain; 2017. pp. 432-436.

[15] Fan M, Khademi M. Predicting a Business' Star in Yelp from Its Reviews' Text Alone. arXiv 2014; arXiv:1401.0864.

[16] Duwairi R, El-Orfali M. A study of the effects of preprocessing strategies on sentiment analysis for arabic text. Journal of Information Science 2014; 40 (4) :501-513. doi: 10.1177/0165551514534143

[17] Saad M. The impact of text preprocessing and term weighting on arabic text classification. MSc, Computer Engineering Department, The Islamic University, Gaza, 2010.

[18] Uysal A, Gunal S. The impact of preprocessing on text classification. Information Processing and Management 2014; 50 (1): 104-112. doi: 10.1016/j.ipm.2013.08.006

[19] Shiha M, Ayvaz S. The effects of emoji in sentiment analysis. International Journal of Computer Electrical Engineering 2017; 9 (1): 360-369. doi: 10.17706/IJCEE.2017.9.1.360-369

[20] Wegrzyn-Wolska K, Bougueroua L, Yu H, Zhong J. Explore the effects of emoticons on twitter. Computer Science and Information Technology 2016; 6 (1) : 65-77. doi: 10.5121/csit.2016.61006

[21] Park Y, Byrd R. Hybrid text mining for finding abbreviations and their definitions. In: Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing; Intelligent Information System Institute; Pittsburgh; 2001. pp. 126-133.

[22] Kaur A, Singh P, Rani S. Spell checking and error correcting system for text paragraphs written in punjabi language using hybrid approach. International Journal Of Engineering And Computer Science 2014; 3(9): 8030-8032.

[23] Bertoldi N, Cettolo M, Federico M. Statistical Machine translation of texts with misspelled words. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics; Los Angeles, California, USA; 2010. pp. 412-419.

[24] Müller P, Ohnheiser I, Olsen S, Reiner F. Multi-word expressions. An International Handbook of the Languages of Europe, Berlin, Germany: HSK series, 2011.

[25] Constant M, Sigogne A, Watrin P. Discriminative Strategies to integrate multiword expression recognition and parsing. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics; Jeju Island, Korea; 2012. pp 204-212.

[26] Schonlau M, Guenther N, Sucholutsky I. Text mining with ngram variables. The Stata Journal 2017; 17 (4): 866-881. doi: 10.1177/1536867X1801700406