Investigation of feature selection methods and classifiers on Text data

Zhaozhi Zhang, Shuo Yang

School of Informatics and Computing, Indiana University, USA

1 Introduction

One of the most critical challenges in the text classification by machine learning algorithms is the feature dimension. Since there are enormous different words in the text documents, the useful features for the classification are usually buried in the non-related ones. Besides that, even we can pick out all the words related to the classification task, and only use them as the features to perform the machine learning classifiers, the computation is usually intractable, since most of the machine learning algorithms are ineffective when dealing with the high dimensional data set [1]. Because of the above reasons, feature selection is an essential element for training an efficient and accurate machine learning model [2]. There are various measures which could be used as the templates for defining the features of a text document, such as N-grams, part of speech tagger and parsing. We used N-grams specifically unigram, bigram and trigram as the templates to define the features, and used the frequency of the gram as the value of the corresponding feature. The working flow is showed in Fig 1. Firstly, the training set went through the feature extractor, and we got 5123 instances with 2728 unigram features, 11266 bigram features, and 16570 trigram features; secondly, the feature selection algorithms assigned a unique ranking to each of the selected features groups, i.e. unigrams, unigrams & bigrams, unigrams & bigrams & trigrams, based on the training set; thirdly, both the training set and testing set were filtered by only keeping the first 1000 features in each group (which returned a 566097*1000 testing set); Finally, the testing set were classified by different machine learning models learned from the training set. We tried three different feature selection algorithms and three classifiers on the recipe rating classification task based on the review documents. And the results from these algorithms are evaluated by three different measures: mean absolute error, F-measure and area under ROC.

2 Pre-processing

There are totally 26765 recipes in the experimental data set, and each of them has a total rating and multiple reviews with different individual ratings. The task is to training a machine learning model so that the rating of a recipe can be predicted based on the features selected in the feature selection step. We used

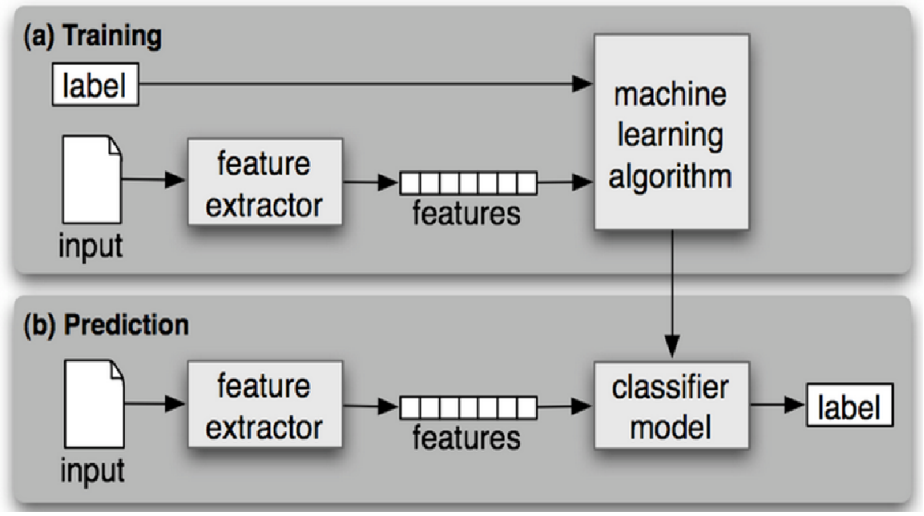

Fig. 1: Flow Chart of Classification Process
Picture from online source: *http : //nltk.org/book/ch*06.*html*

the first 1000 recipe as the training sets. After pre-processing, Features are extracted with Python. Some tools provided by Natural Language Toolkit (NLTK) are utilized to simplify the work of preprocessing the corpus.

2.1 Extracting Uni-gram Features

To extract uni-gram features, each of the reviews is separated into a sequence of sentences by PunktSentenceTokenizer, and then tokenized into tokens by TreebankWordTokenizer. Some word tokenizer like WordPunctTokenizer can separate the sentences and tokens simultaneously, but since the TreebankWordTokenizer is selected, PunktSentenceTokenizer is needed.

TreebankWordTokenizer is selected since its rules of tokenization is considered better matching the goal of this experiment: evaluating the effects of feature selection approaches on machine learning algorithms with unigram, bigram, and trigram features. A glance of the tokenization rules of four different tokenizers can be provided with as table 1 shows.

WordPunctTokenizer is used at the very beginning for this experiment, since some punctuation combinations like ":)" can be considered meaningful, but is latter replaced since TreebankWordTokenizer seems working better on the cases like "cannot" and "don't", which are supposed to be representatives of effective bigrams.

Text Sample	“cannot”	“don’t”	“!!!!”	“:)”	“computers.”
TreebankWordTokenizer	“can” “not”	“do” “n’t”	“!” “!” “!” “!”	“:” “)”	“computers” “.”
WordPunctTokenizer	“cannot”	“don” “”’ “t”	“!!!!”	“:)”	“computers” “.”
PunktWordTokenizer	“cannot”	“don” “’t”	“!” “!” “!” “!”	“:” “)”	“computers.”
WhitespaceTokenizer	“cannot”	“don’t”	“!!!!”	“:)”	“computers.”

Table 1: Comparison of Different Tokenizers

After the tokenization, WordNet Lemmatizer is manipulated to recognize the different inflected forms of the words. The following table is showing the effects of the stemmers / lemmatizer on a sample generated by TreebankWordTokenizer.

Text Sample	“The cat doesn’t like playing computers. ”
TreebankWordTokenizer	“The cat does n’t like playing computers . ”
Porter	“The cat doe n ’ t like play comput .”
Lancaster	“the cat doe n ’ t lik play comput .”
WordNet Lemmatizer	“The cat doe n ’ t like playing computer .”

Table 2: Comparison of Porter, Lancaster and WordNet Lemmatizer

As table 2 shows, WordNet Lemmatizer is selected, since it can differentiate the words “compute” and “computer” which play different roles in the context while the other two stem both of them as “comput”. Stemmers are usually more aggressive than Lemmatizers. The length of the stemmed and lemmatized tokens should not affect the runtime performance of the algorithms so much, due to the indexing operations mentioned in the next section. Thus, the unigram features are extracted.

At this step, the reviews are converted to sequences of lemmatized tokens.

2.2 Indexing and Encoding

Since thousands of features are generated intentionally for this experiment, and all the original features are in text format which is supposed to be quite expensive in time and memory cost when being processed by computing systems, before being filtered and passed to the specific algorithms, these features are replaced by integer indices by the following steps:

1. Build a dictionary of the features, and assign distinct integers to unigram features as their values. Since elements in dictionary are not ordered, an inverted dictionary is needed for fetching elements from their values. Thus, the values of the features are their keys as well.
2. Encode the reviews, which are currently sequences of lemmatized tokens, with the keys of the tokens, to sequences of integer keys.

3. Generate a feature set consist of feature keys and occurrences for each of the encoded reviews, with stop words filtered out.
4. Extract adjacent tokens from encoded reviews. In this case, they are tuples containing two or three integer keys.
5. Assign distinct integer keys for the integer tuples. Thus, features besides uni-grams get integer keys as well.
6. Attach the keys and occurrences of the bigram and trigram features to the feature set of each of the encoded features.
7. Filter the feature sets as needed. This step can be done together with Step 6. Currently the feature sets can be considered as an extremely sparse matrix with many feature columns filtered out.
8. Re-generate new keys for the remained columns in the feature sets matrix.

Thus, a Coordinate-wise (COO) sparse matrix indicating the feature sets of the reviews is made for the next phases. Please note that, at Step 3, the stop words are filtered out when generating unigram feature sets for reviews, which means word like "with" may not appear in the feature set as a unigram, but it may be combined with some other words, and appear in the feature sets as a part of bigram or trigram features, e.g. "with a fork". Another point is if Step 8 is omitted, the filtering effects may not present with some of the algorithm implementations in the next phases, which means they still operate on the filtered out columns. The process described in 2.1 and 2.2. are showed in Fig 2.

3 Feature Selection Methods

For multi-class data, there are two ways to perform feature selection methods: globally and locally. The one selects the same feature set for every class is called Global feature selection, and the other selects a particular feature set for each class by reassigning the labels through 'one Vs all' approach [3]. The labels include 4 different ratings, which is a multi-classification task. We used three different algorithms to perform global dimensionality reduction: support vector machine, Chi-square and information gain. SVM belongs to the category of wrapper method which can get more accurate results yet takes extensive amount of time. The other two algorithms are under the category of filter method which can finish the task in shorter time by sacrificing the accuracy. In our experiments, the run time for Chi-square and information gain is in the measure of seconds while the SVM is in hours. The top 40 features picked out by these three algorithms are showed in the appendix. As it shows, most of the features contain strong sentiment (either positive or negative) which is a favorable measure as to classify the reviewers' rating. It is worth to mention that SVM picked out the bigram 'not good' and trigram 'good, but' as one of the most important features, which shows that in those cases, bigram and trigram are more reliable than unigram as features to predict the reviewers' attitudes.

In this report, top 1000 features in each group are selected. Top 2000 features were tried as well, but no significant improvements have been found.

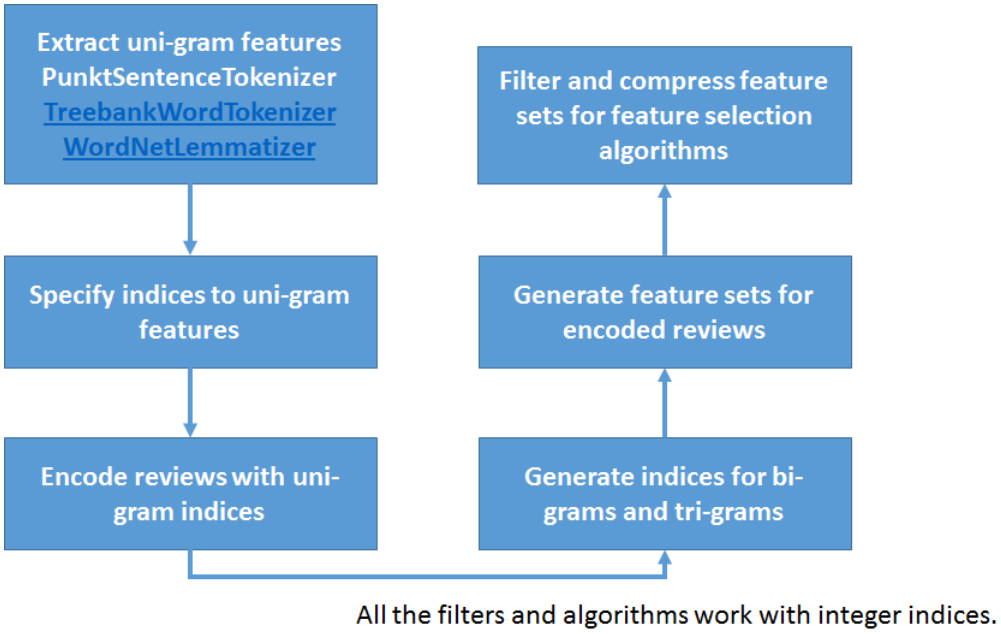

Fig. 2: Flow Chart of Pre-processing

4 Classification Methods

We tried three different machine learning classifiers to predict the ratings of the reviews, which are decision tree, naive Bayes and SVM. The results are shown in Fig 3 to Fig 5. Each cluster represents a different classifier, which are naive Bayes, decision tree and SVM from left to right, and each bar in every cluster represents a different feature selection method, which are information gain, Chi-square and SVM from left to right. As the results showed, there is no much difference on the performance among different feature selection algorithms (three bars in every cluster). However, naive Bayes as a classifier (first cluster) presents better results in all three measures. So we choose naive Bayes as the classifier, and draw the different grams in Fig. 6 where each cluster represents different feature selection algorithm and each bar in every cluster represents different N-gram. As the picture shows, there is no much difference among unigram, unigram+bigram and uigram+bigram+trigram as the feature template.

5 Conclusion

As our experiment results showed, different feature selection algorithms have not shown much difference on the performance in the recipe rating task. However SVM as a wrapper method took a significant long time to get the feature ranking. Naive Bayes as a classifier shown a slightly better performance compared

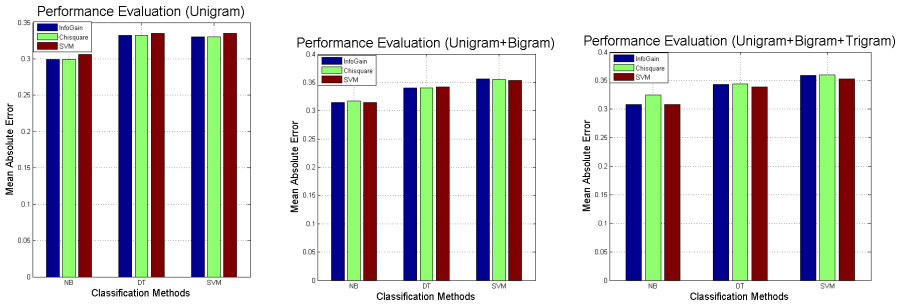

Fig. 3: Mean Absolute Error

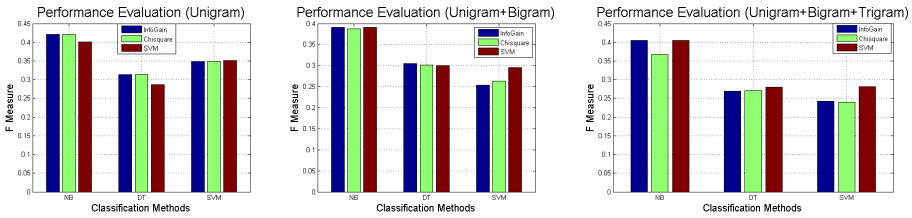

Fig. 4: F-measure

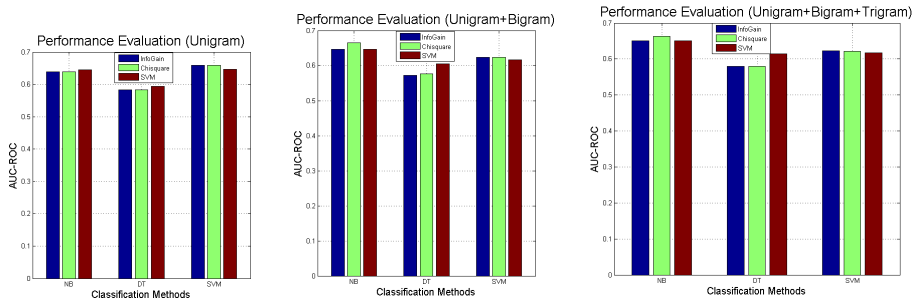

Fig. 5: AUC-ROC

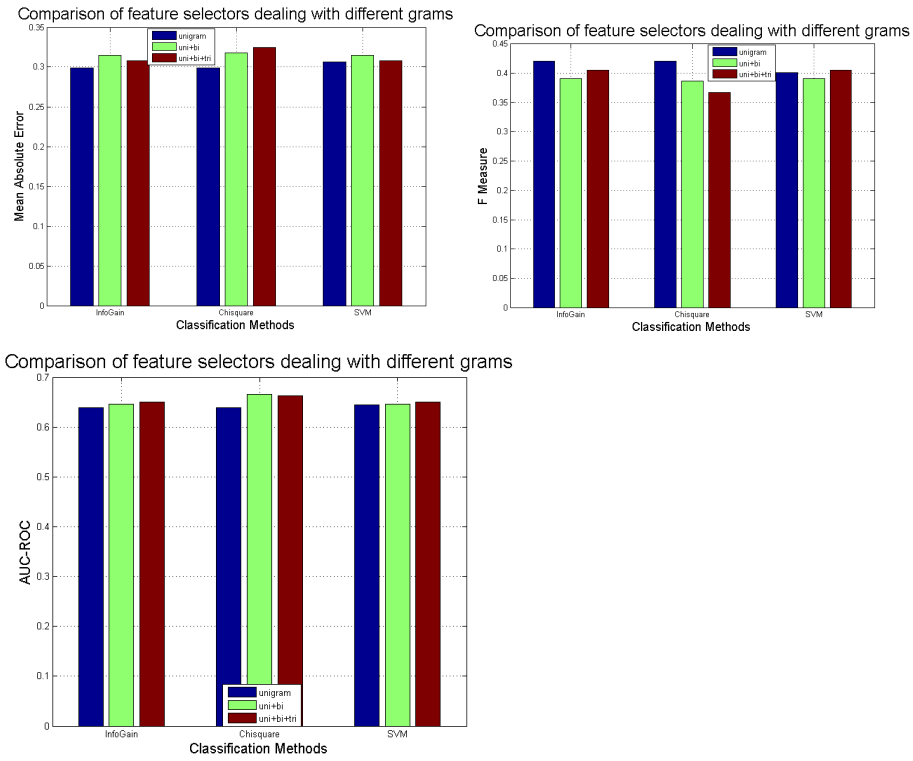

Fig. 6: Comparison of N-grams

to decision tree and SVM. It is worth to mention that, in our preliminary experiments where we used 3416 samples as training set and 1708 samples as test set, the algorithms showed much better results in all measures of mean absolute error, F-measure and AUC-ROC, especially for F-measure. This difference could be caused by not enough training instances compared to the enormous amount of testing samples.

References

1. Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.
2. George Forman, Isabelle Guyon, and Andr Elisseeff. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.
3. Fei Xia. Feature selection slides, January 2013.

Appendix: Top 40 features selected by different algorithms

Feature Ranking	Unigram	Unigram+Bigram	Unigram+Bigram+Trigram
1	great	great	great
2	orleans	was+good	onto
3	fourth	used	good+’,’+but
4	blah	head	body
5	alright	not+good	a+recipe
6	bland	two+fork	blandness
7	japanese	very+good	everyone+in
8	horrible	delight+.	so+good
9	absorb	disappointment	combination+of+the
10	used	was+interesting	seder
11	blackberry	not+very	were+pretty
12	loved	rating+is	wow
13	notch	horrible	shorter
14	tasty	sitting	a+kid
15	wow	really+liked	an+easy+recipe
16	yummy	four+fork	fabulous
17	wing	bland	’,’+go
18	liked	difference+in	of+thyme
19	yum	strip	bland
20	delicious	reviewer+who	rating+is
21	ok	’–’+’–’	change
22	candy	this+many	the+milk
23	bouillon	shorter	should+have
24	bleu	added+’2’	perfect
25	concept	i+tried	wonderful
26	spot	the+cupcake	grocery+store
27	ever	a+recipe	weight
28	mild	cheese+sauce	myself+ ’,’
29	altitude	csa	loved
30	added	delicious+with	le+’.’
31	kinda	added	not+very
32	waste	alright	is+not
33	delight	very+tasty	year+and
34	good	was+not	a+meat
35	meh	blah	not+worth
36	tweak	blandness	and+the+texture
37	semisweet	still+turned	excellent
38	’!’	become+a	last+minute
39	raved	a+little	them+’!’+i
40	enjoyed	’2’+fork	never

Table 3: Feature Ranking from SVM

Feature Ranking	Unigram	Unigram+Bigram	Unigram+Bigram+Trigram
1	delicious	delicious	delicious
2	easy	easy	easy
3	loved	loved	loved
4	'!'	'!'	'!'
5	used	used	used
6	great	great	great
7	bland	bland	bland
8	waste	waste	waste
9	perfect	i+used	i+used
10	added	loved+it	loved+it
11	made	waste+of	waste+of
12	tasty	perfect	perfect
13	awful	added	added
14	also	delicious+ '.'	delicious+ '.'
15	ok	','+but	','+but
16	disappointing	'.'+i	'.'+i
17	served	made	made
18	everyone	tasty	tasty
19	wonderful	a+waste	a+waste
20	next	awful	awful
21	instead	also	also
22	liked	easy+and	a+waste+of
23	little	ok	easy+and
24	disappointment	disappointing	ok
25	nice	a+little	disappointing
26	'?'	instead+of	a+little
27	quick	served	instead+of
28	excellent	everyone	served
29	much	'!'+I	everyone
30	rich	a+great	'!'+I
31	bit	next+time	a+great
32	worst	wonderful	next+time
33	good	will+definitely	wonderful
34	disappointed	wa+good	will+definitely
35	enjoyed	next	'.'+i+used
36	something	delicious+'!'	was+good
37	ever	very+tasty	next
38	away	and+delicious	delicious+'!'
39	','	definitely+make	very+tasty
40	dinner	bland+'.'	and+delicious

Table 4: Feature Ranking from Chisqure

Feature Ranking	Unigram	Unigram+Bigram	Unigram+Bigram+Trigram
1	delicious	delicious	delicious
2	loved	loved	loved
3	easy	easy	easy
4	used	used	used
5	great	great	great
6	'!'	'!'	'!'
7	bland	bland	bland
8	waste	waste	waste
9	added	i+used	i+used
10	perfect	loved+it	loved+it
11	awful	waste+of	waste+of
12	made	delicious+ '.'	delicious+ '.'
13	tasty	added	added
14	also	perfect	perfect
15	disappointing	a+waste	a+waste
16	served	awful	awful
17	ok	easy+and	easy+and
18	everyone	made	made
19	next	','+but	','+but
20	liked	tasty	a+waste+of
21	wonderful	also	tasty
22	disappointment	'.'+i	also
23	quick	a+little	'.'+i
24	nice	disappointing	a+little
25	instead	served	disappointing
26	disappointed	instead+of	served
27	little	ok	instead+of
28	'?'	everyone	ok
29	much	delicious+'!'	everyone
30	excellent	next+time	delicious+'!'
31	worst	definitely+make	next+time
32	bit	bland+'.'	definitely+make
33	enjoyed	next	'.'+i+used
34	good	and+delicious	bland+'.'
35	change	a+great	next
36	something	will+definitely	and+delicious
37	ever	liked	a+great
38	dinner	not+worth	good+','+but
39	','	wonderful	will+definitely
40	simple	very+easy	liked

Table 5: Feature Ranking from Information Gain