Feature Extraction for Classification of Recipe Data: Evaluating Types of N-grams as Features. [*]

Alina Banerjee
School of Informatics and Computing
Indiana University
Bloomington, IN, USA
banerjea@indiana.edu

Daniel Whyatt
Department of Linguistics
Indiana University
Bloomington, IN, USA
dwhyatt@indiana.edu

Abstract

With the ever increasing amount of reviews written by users of many types of services,the possibility of extracting information from such reviews is of great interest.Very often such reviews consist of a rating and associated text.In this project,the reviews are responses to recipes on Epicurious.com.Using NLP and machine learning techniques,we evaluate three approaches to predicting ratings from user reviews at the site.The features used in prediction will consist of word n-grams,POS n-grams and syntactic n-grams.

1 Introduction

With a massive increase in online services and content,opinions and feedback from users has correspondingly increased.Such feedback is highly valuable for both newer users of services as well as the service providers themselves.In many cases,with huge amounts of reviews available,the need to use NLP(Natual Language Processing) and machine learning techniques for quality rating or prediction,is in much need.At the same time,memory and speeds of computing have also increased so the process of prediction can potentially be much higher in quality when features for prediction are exacted with care.

In this project,we use a dataset from the website,Epicurious,with individual user reviews linked to user ratings on a scale from 1 to 4 stars,with 1 indicating the lowest,4 the highest.Aggregated ratings of all individual user reviews for a recipe can have values of 0.5.From this data,we seek to extract n-grams which may have words,POS(Parts of Speech) tags of words or tagged non-terminal nodes.With a predetermined baseline of word counts taken from a sentiment lexicon,we add n-grams as features in separate instances and then compare the accuracy of prediction for given data sets.

[*]For submission as the project report for B659/L645 course project

2 Project Steps

We started with the given data where each file represents a recipe and has aggregated individual reviews of the recipe in it.Each individual recipe has a rating of 1 to 4 associated with it,in increments of 0.5.When all these individual ratings for a recipe are averaged,the overall rating is encoded into the file name.The overall rating in the file names had already been converted to the nearest whole number or nearest 0.5 value after averaging the recipe rating within the file.

2.1 Data Preprocessing

The files with recipe ratings ranged from 1 to 4,with 0.5 increases.The number of recipe files with ratings 1 and 2 were far less in number than the number of files with recipe ratings 3 and 4.The files of recipe rating 4 were about 800 times as many as files of rating 1.In order to have the number of files for each rating class as proportional as possible,we removed all recipe files with a non whole number rating.

2.2 Tagging of Data

For tagging the data with POS tags,we used the Stanford Tagger,part of the entire Stanford NLP suite of tools.The recipe files were directly fed to the tagger with the in-built English language models used.

The English language models had been created using the Penn Treebank tag set.The Stanford Tagger uses the a Hidden Markov Model(HMM) along with maximum entropy for determine the POS tags for a given sentence.The model used for the tagger uses one of the bidirectional models in which the HMM is built using the transition probabilities from both the beginning to the end of the sentence and vice-versa.

The English language models are created using the Wall Street Journal Penn Tree Bank.This data set is qualitatively different in that it has professionally edited and has correct spelling and grammar.It is far closer to an ideal standard of the written English language than the sentences likely to be found in our data which is unedited text by users of varying levels of English language proficiency.This point is likely to cause some problems in tagging as there may be words which are spelt incorrectly or places in correctly in a sentence as the tagger will go ahead and add the wrong POS tag to it.Given the bidirectional HMM model,this is likely to affect the correctness of the POS tagging for the correctly written words right next to a misplaced or missing word.

2.3 Parsing of Tagged Data

From the available tagged data,we use the Berkeley Parser to parse the data into parse trees where terminal nodes are linked to POS tags and constituency tags are the non-terminals.The parser repeatedly splits and merges the non-terminal symbols over several iterations to maximize the probability of a sentence given a training grammar.For each iteration,a newer parse is determined where the non-terminals are made sub-symbols or children of the original non-terminals in the sentence.Correspondingly the terminals i.e the words are split in many combinations to enable different combinations of non-terminals.For each iteration,the probability of a parse is determined using the inside and outside probabilities.Using these values,the most likely parsed sentence is output using these same probabilities,the Viterbi algorithm and the Penn TreeBank grammar.

Though it is possible to get more than one parsed output,ranked on decreasing probabilities,we use the default option of getting the most likely parse for each sentence.Before parsing,we ensured that the recipe names were made distinct from the rest of the text by adding a unique string to each recipe name,so that after parsing,processing the text could be done separately from the recipe names.

In practical terms,the Berkeley parser has been able to parse file of size of the order of between 10,000 and 20,000 files merged to together.It has been fairly efficient and fast for such a large number of files concatenated into 1 file.

2.4 Separation of Parsed Data : Train, Test, Evaluation

Apart from the original recipe data,we have been given a file labelling all recipes in either one of the 3 categories: Train,Test,Eval(Evaluation).After getting the parsed data,where we ensured that the recipe names could be made distinct on 1 file,we tag each recipe as a member of the training or test or eval set.

For further processing,we only used data from the training data set to extract features and construct the classifier model and use the test and eval sets for evaluating our model.

2.5 Data Sampling from all Data Sets(Train, Test, Eval)

After all this processing,the recipe data available for the class 1 set was far lower than the class 4 data.Proceeding further with these imbalances in the number of data instances would have meant that the classifier would have been trained on far more class 4 instances than others,raising the possibility of overfitting the training data from classes 3 and 4.In order to avoid this,we randomly sampled for 100 files from each class for each set :train,test,eval.This ensured that while training,testing and evaluating,the data sets weren't skewed and biased towards any one of the four rating classes.

2.6 Feature Extraction

Words: For the training set,we have 4 classes: 1, 2, 3,and 4 For each class,we have 100 files,each file having a recipe name,its overall rating with all the individual reviews or text for the recipe.

From the original training files,we extracted all the terminals or words in all the individual recipe texts.This ensured that we could keep a list of all the words present on the training files.From this list,we sorted and took the top 500 words.These words were matched to a sentiment lexicon,which had both positive and negative sentiment words.The words from our training files were matched to a combined list of the positive and the negative words and then the top 100 words were taken which would form word features that we would look for in our 400 files making up the training data set.The extraction of words was done using regular expressions in Python.

Word bigrams,Word trigrams : From the original training files,we extracted all the word bigrams and trigrams in all the individual recipe texts.This would help us to get the most common bigrams as well as trigrams.Sorting and taking the unique top 100 from each of the list of bigrams and trigrams would give the the final set of bigrams and trigrams to search for in our training data set.The extraction was done using regular expressions in Python with the expression for word bigrams being:

```
r"\(-*[A-Z]*[a-z]*\?*\.
*-*'*,*\"*[a-z]*:*;*\s(-*[A-Z]
*[a-z]*\?*\.*-*'*,*\"*[a-z]*:*;*)\)
```

and the the expression for word trigrams being:

```
(r"\(-*[A-Z]*[a-z]*\?*\.*
-*'*,*\"*[a-z]*:*;*\s(-*[A-Z]*
[a-z]*\?*\.*-*'*,*\"*[a-z]*:*;*)\)
```

Now these word bigrams and trigrams could be used as features for classification of the recipres in the training data set.

POS bigrams,POS trigrams : From the original training files,we extracted all the POS bigrams and trigrams in all the individual recipe texts.giving us most frequent POS bigrams as well as POS trigrams in the text.Sorting and then taking the unique 100 from each of the list of these bigrams and trigrams would give the the set of POS bigrams and POS trigrams to use as features in out training data set.The extraction was done using regular expressions in Python with the expression for POS bigrams being:

```
r"\((-*[A-Z]*[a-z]*\?*\.*-*'*,*
\"*[a-z]*:*;*)\s-*[A-Z]*[a-z]*
\?*\.*-*'*,*\"*[a-z]*:*;*\)"
```

and the the expression for POS trigrams being:

```
\((-*[A-Z]*[a-z]*\?*\.*-*'*,*\"*
[a-z]*:*;*)\s-*[A-Z]*[a-z]*
\?*\.*-*'*,*\"*[a-z]*:*;*\)
```

Non terminal POS bigrams,Non terminal POS trigrams :In this step,the non-terminal POS bigrams and trigrams in all the individual recipe texts were extracted giving us most frequently available POS bigrams as well as POS trigrams for all non-terminals in the text.The extraction for bigrams and trigrams was done using regular expressions in Python.

3 Classification

The process of classification involves using data where each data instance has a number of features and one class label.The pair of the set of features and the class label form the training data set. Separate from the training data set is a test set where the data instances have a set of features and a class label.The goal of classification is to use the features and the model obtaining from the training set and output the class of each label.With a set of class labels for the training data,the percentage accuracy can them be determined from the counts of matches between the predicted class values and the actual class values.

Since we have more than 2 classes for the recipe data set,the category of classification is multiclass instead of binary.

3.1 Use of SVM(Support Vector Machines)

The use of SVMs for classification is very helpful as SVMs use a highly efficient entity called a kernel.A kernel is a transformation function which maps data from a low dimensional vector space to a much more higher dimensional vector space.

In many instance of training data for classification,the features in the original space are not very distinct.This means that classification is likely not going to be very efficient or not possible at all as data may not be linearly separable i.e it may not be possible to draw a line(2-dimensional data),plane(3-dimensional data) or a hyperplane(higher than 3 -dimensional data).This often occurs in neural networks.SVMs get around the problem of non-linearly separable data by constructing a kernel or a function which takes data in one feature space(as a list of vectors where each vector represents a data instance and each member of the vector is a feature and its value) to a much higher dimensional feature space(hyper-dimensional).This transforms the data vector values into suitable values in the higher dimensional space in such a way that now the data instances are distinct enough to be linearly separable. The SVM draws a maximum margin hyperplane in this higher dimension which ensures that the the data instances have a minimum distance between themselves and the separating plane which makes classification valid with a positive degree of confidence.Once this classification is done,the data is transformed back into the original feature space so that the results can be evaluated.

3.2 LIBSVM

LIBSVM is a language agnostic library for creating SVM kernels.We have used the python interface of the library.The "svm-train" binary is the one used for building the training model.The input needs to be fed as:

```
./svm-train input_training_file
```

The program outputs a training model automatically with the same name as the input file and a ".model" extension.This model is subsequently used for prediction as:

```
./svm-predict input_test_file
trained-model-file output_file
```

Data Format: For any input file(training,test or eval),the format needs to be in the following form:

```
<label> <1>:<val1> <2>:<val2> ...
.
.
```

where each line contains an data instance and is ended by a newline. The "$< label >$" is an integer indicating the class label (multi-class classification is supported). Here as we have 4 classes,the values of label range from 1 to 4.The pair "$< index_N >:< value >$" gives a value for a particular index with the feature number N.N is an integer starting from 1(except for pre-computed kernels) and always are in ascending order.$< value >$ is a real number.

For the test data,the labels are only used to calculate accuracy.Since we have labels for all the test data,we have added them to the test file and not used the option of filling in the first column with any random values.

Features	**Accuracy(%)**
Sentiment words	25
Sentiment words + Word Bigrams	41.667
Sentiment words + Word Bigrams + Word Trigrams	48.333
Sentiment words + POS Bigrams	55
Sentiment words + POS Bigrams + POS Trigrams	55
Sentiment words + Bigrams for Non-terminals + Trigrams for Non-terminals	-

Table 1: Accuracy rates for separate feature combinations in the test data set

4 Execution and Results

Execution:For the data tagging and the parsing steps,we ran batch jobs on the IU Mason cluster which sped up the tagging and training a great number of files.

For each of the features shown in the two tables below,we extract the counts for each feature for all the instances in our training data.This is used to train models for each feature combination shown.Then the same process is run over the test and eval data so that with the features count from these,the individual models can be used to gauge the accuracy in classification.

All the scripts for dealing with data: creating the subsets of training,test and eval data,making the list of bigrams and trigrams as features for both words,POS tags and non-terminal POS tags,and the final counting of feature values from all data sets was done with scripts written in Python 2.7.The standard Python modules for regular expressions(re),for array processing(NumPy) and functions such as "bigrams","trigrams" from the NLTK(Natural Language Toolkit) platform were used.

5 Discussion

For all the classification,the counts of the words common to the training data and the sentiment lexicon have been kept as features and the accuracy of 25% for the test data set,32% for the eval data set,is

Features	**Accuracy(%)**
Sentiment words	32
Sentiment words + Word Bigrams	38
Sentiment words + Word Bigrams + Word Trigrams	47.0588
Sentiment words + POS Bigrams	32.3529
Sentiment words + POS Bigrams + POS Trigrams	29.411
Sentiment words + Bigrams for Non-terminals + Trigrams for Non-terminals	-

Table 2: Accuracy rates for separate feature combinations in the Eval data set

used as the baseline to check the validity of adding other features for classification.

5.1 For the test data:

Features with words and word n-grams only From the reviews test data,we had expected a linear increase in the percentage of accuracy from the counts of all sentiment words to the total counts of sentiment words,bigrams for non-terminals and trigrams.

From Table 1,the values indicate that while we see an increase in the percentage of accuracy from adding word bigrams to the sentiment word as features,adding trigrams gives a slightly higher accuracy.This is as expected as counts of unique trigrams are likely to make the data features more distinct than with only considering bigrams and word counts as features.

Features with words and POS n-grams only Here we see that the addition of POS trigrams to the word and POS bigrams count does not bring any increase in accuracy of the prediction results.The most likeliest explanation is that compared to word trigrams,POS trigrams for non terminals are less distinct for a fixed set of POS tags.This indicates the likelihood that all data instances have counts of POS tags of the same order and so these trigrams do not add any significant information for the classifier to distinguish data instances.

5.2 For the eval data set :

Features with words and word n-grams only For this set,we had a similar linear increase in the percentage of accuracy from the counts of all sentiment words to the total counts of sentiment words,bigrams for non-terminals and trigrams,as we did in the test set.

The accuracy values indicate that an increase in percentage terms from adding word bigrams to the sentiment words as features.Adding word trigrams gives a slightly higher accuracy than just the word bigrams.Again as unique trigrams are likely to make the data features more distinct than with only considering bigrams,this increase validates our hypothesis.

Features with words and POS n-grams only Here we see that the addition of POS bigrams to word counts brings a very slight increase in accuracy where the addition of POS trigrams to the bigrams and words actually decreases the value.

The possible explanation for the lower accuracy result is that adding the extra set of POS trigrams as features actually leads to the eval data set overfitting the training model.This leads to most or all the predictions being of the same class labels as the training data and loss in accuracy when used for predicting eval data instances.Since we create the eval data set from random sampling of the larger given eval data,this needs to be further checked by recreating different eval data sets and averaging the accuracy percentages over all the runs.

5.3 Features with words and Non-terminal POS n-grams only

For both the eval and the test data set,we could not correctly estimate correct counts for the parsed bigrams and trigrams tags of non-terminal nodes.The feature set that we could come up with was extremely sparse for the SVM to train a classifier.Hence results are kept marked blanked until in the near future,we can correct our algorithm and come up with usable feature counts for training and testing.

6 Conclusion and Future Work

From the features used,we can conclude that the use of sentiment words can only act as a baseline from which higher order n-grams of both words and POS tags are required to give an acceptable lower level of about 40-50 % in accuracy values for prediction.For both the test and eval data,trigrams when used as features in conjunction with bigrams,are the best for prediction.Since the accuracy values are most verifiable for word bigrams and word trigrams,the n-grams of words as well as sentiment word counts seems to be the best feature.

In the near future,we would like to complete the data extraction for non-terminal POS tags and add it to our results so that we can see which type of POS tags,those from terminals and/or those from non-terminals,are likely to help in classification.With this the addition of specific non-terminals such as modals and trying more complex combination word n-grams and POS tag n-grams are goals that we want to include as to find out the correct prediction of the rating of a recipe from its reviews.

Acknowledgements

We are indebted to Prof. Kubler for giving us the opportunity to apply our classroom NLP knowledge in this project.Her support and enthusiasm has immensely helped in the execution of the project and her patience in always being open to questions and suggestions has given us a lot of confidence in tackling challenges.

We would like to thank Levi King,from the IU Linguistics department, for his help in setting up scripts on the IU Mason server for tagging and parsing data.

References

Ning Yu,Desislava Zhekova,Can Liu and Sandra Kubler 2013. Do Good Recipes Need Butter?Prediction User Ratings of Online Recipes In *Cooking with Computers Workshop,IJCAI 2013*

Chih-Wei Hsu,Chih-Chung Chang and Chih-Jen Lin 2010 A Practical Guide to Support Vector Classification, Department of Computer Science National Taiwan University, Taipei 106, Taiwan

Slav Petrov Leon Barrett Romain Thibaux Dan Klein 2006 Learning Accurate, Compact, and Interpretable Tree Annotation In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL

Dan Klein and Christopher D Manning. 2003 The Stanford Parser *Accurate Unlexicalized Parsing* In Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430.

Chris Manning and Hinrich Schutze 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. Cambridge, MA: May 1999