11-761
Language and Statistics Project

Apoorva Bansal
Language Technologies Institute
CMU
apoorvab@andrew.cmu.edu

Shachi Paul
Language Technologies Institute
CMU
shachip@andrew.cmu.edu

Abhinav Arora
Language Technologies Institute
CMU
aa1@andrew.cmu.edu

Vishrav Chaudhary
Language Technologies Institute
CMU
vchaudha@andrew.cmu.edu

April 25, 2016

Contents

1 Introduction

In this project, our goal was to investigate, discover and exploit deficiencies in the conventional trigram language model, using statistical methods. We approach this problem by experimenting with a combination of linguistic, semantic and statistical features to a machine learning classifier to distinguish real Broadcast News articles from fake "articles" generated by a Broadcast-News-trained trigram model.
In this report, we present a machine-learning approach in the form of Support Vector Machines [1] to classify between fake and real articles. We tried using a bag-of-words model, n-gram language models, topic-modelling (Latent Dirichlet Allocation) and Stanford parser to generate features that capture the local dependencies and semantic and syntactical information of the articles in the data set. By feeding a combination of these features to our machine learning classifier (SVM), we hope to design a program that is capable of discovering the deficiencies in the conventional trigram language model and therefore discriminating between fake and real articles.

2 Feature Engineering

Our experience with language technologies reveals that feature engineering the most vital part of such a project. Therefore, the bulk of our time was spent on feature engineering to evaluate the best and the most informative features for the model. The features we used for training our classifier can be broadly divided into three categories: syntactical, semantic and statistical. In this section, we will discuss in detail about these features.

2.1 Bag of Words

We begin with probably the most basic feature, using bag-of-words features. The bag-of-words model is commonly used in methods of document classification, where the (frequency of) occurrence of each word is used as a feature for training a classifier. This feature however, wasn't able to accurately distinguish between the classes of articles. This is probably because of the distributional differences of sentences in the articles in the training set and the development set.

Traditionally, Bag of Word features think of a document as a collection of words. Since these did not work well initially, we tried a modified approach that involved *Bag of Bigrams* instead of the usual Bag of Words approach. The intuition behind this was using this feature could help us capture some context. However, this turned out to be very bad. The reason we feel that this feature did not work out for us was due to the sparsity of the bigrams due to limited training data.

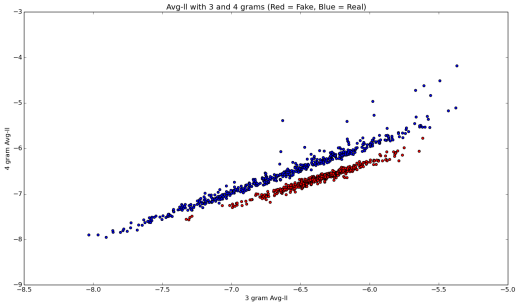
2.2 TF-IDF

We upgraded the bag-of-words feature (explained above) by penalizing the more commonly occurring terms by their inverse document frequency. This feature again wasn't able to ably differentiate the fake articles from the true ones. This is because this feature doesn't trap contextual meaning of the sentences in the articles.

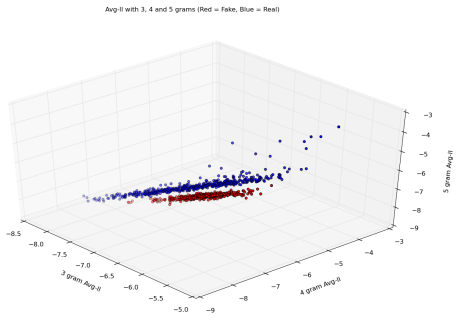
Similar to the above Bag of Words approach, we also tried the Tf-Idf approach with bi-gram tf-idf. Due to the same reasons as discussed above, this did not give good results.

2.3 Type Token Ratio

As learnt in the initial lectures in class, a type-token curve is a very good measure of language identification. We could see this clearly in our assignment 1 as well. Here the feature is represented by the ratio of the number of types to the number of tokens in the article. In order to handle for different sizes of the articles, we normalized this feature using the number of sentences in the article.
Our intuition behind this feature was that because there were distributional differences between the training and development articles, the type token curve would be a good measure to account for this. However, even this feature did not add much information to our classifier. The reason for this was because they were very few sentences in each article in the development set as compared to that of the training set.



Average Log-Likelihood with 3 and 4 grams



Average Log-Likelihood with 3, 4 and 5 grams

2.4 Average Log-Likelihood of N-gram language models

This feature refers to the N-Gram Language models trained on the provided Broadcast News 100 million words corpus. As we have seen multiple times during this course, N-Gram language models have been the best language models currently available to us. The intuition behind using this approach is that higher order N-grams will be better differentiators of the two types of articles. It is known to us that the fake articles have been generated from a smoothed Trigram model. Therefore, we believed that using higher order N-grams will help in differentiating the two types of articles.

For this purpose, we used the CMU Statistical Modelling Toolkit to train **3-gram, 4-gram, 5-gram, 6-gram and 7-gram** models on the 100 million words, Broadcast News Corpus. Using these models, we calculated the Average Log Likelihood of the articles and used them as the features for our machine learning algorithms after appropriate scaling. The reason we feel that these features are good features because, when plotting the articles on the co-ordinate axis with these features, we were able to observe hyperplanes that were clearly separating the fake and the real articles. This hypothesis is also confirmed when these features give us the best cross-validation and hold-out accuracy using Support Vector Machines with a linear kernel.

2.5 Part of Speech Tags

For this feature, we trained a bigram model on the POS tags. In order to limit the size of the POS vocabulary, we clustered the POS tags to only the first two letters of the tag. Therefore, tags such as *VBZ* became *VB*. The idea behind doing this was to reduce the vocabulary size. Since the POS tags capture the main part of speech in its first two letters, hence we believed that using the first two letters of the tag would be beneficial. We trained two separate Bigram models for each class, fake and true. Once we trained the Bigram language models, we used the scaled Perplexity of the article with respect to the fake articles language model and the good articles language model as features in our machine learning algorithm.
In our experiments, we were able to see that this was a good feature. However,it was unable to perform that well when compared with other features such as N-gram models trained on the provided 100 Million Word Corpus of Broadcast News.

2.6 Topic modeling using Latent Dirichlet Allocation (LDA)

N-gram models only capture the local relation between words in a document. We tried to overcome this shortcoming by using LDA[2] to exploit the semantic relations between different words spread throughout the article. This is based on the premise that real articles should discuss only a few number of topics. Randomly generated fake articles, on the other hand, should have a distribution which is more spread-out across all topics.
We used the gensim[3] toolkit to find the latent topics an article might belong to and extracted different features using that information. This toolkit allows LDA model estimation from a training corpus and subsequent inference of topic distribution on new and unseen articles. We used this toolkit to classify the articles into 50 different topics. Using this information, we extracted the following 2 features for every article:

- **Word Percentage per Topic**
  In this feature, we calculated the percentage of words in an article that belong to a particular topic. This feature gives us an indication of the distribution of topics across an article.

- **Topical Entropy of an Article**
  For each article, we calculated the topical entropy[4] of that article with respect to the 50 latent topics. The intuition behind this feature is that since a real article contains only a few topics, the entropy of its topical distribution should be lower than that of the fake articles.

However, using these features with other features degraded the performance of our classifier. One reason behind this could be the quality of the articles in the development set. The development set contains articles which are as short as - "<s> LET'S TAKE </s>". The topical distributions assigned to such short articles don't really give any information and thus, the features generated didn't add any value to our classifier.

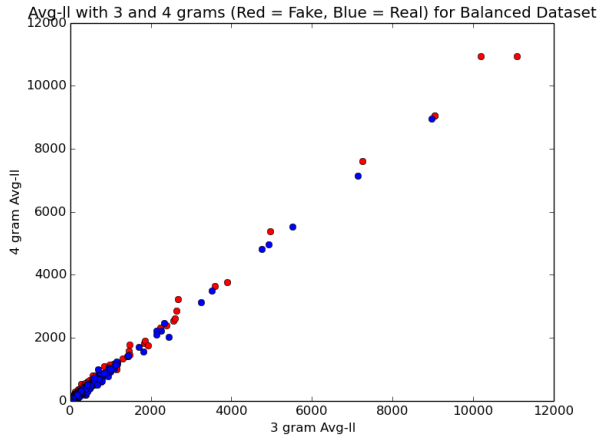
3 Classification Approach

3.1 Classifier Information

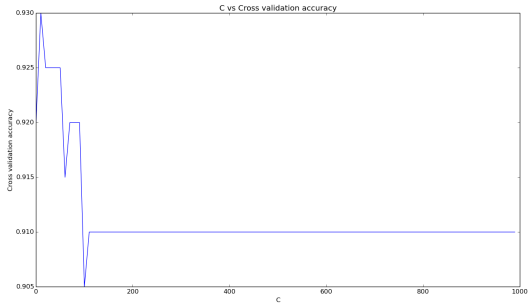
- **Support Vector Machine(SVM)[1]**
  - **Kernel**: Linear
- **Parameters**
  - **C (Penalty parameter of the error term)**: 10

Final Features

Our final feature space consists of five features which consists of the Average log-likelihood of the articles for 3, 4, 5, 6 and 7 grams. We used a linear Support Vector Machine(SVM) to predict the labels of the test data which comprises of fake and real articles. Note that we also used the same classifier for predict the soft labels i.e. the posteriors P(fake|article) and P(true|article). Also our classifier uses a linear kernel as we believe, the training dataset was linearly separable so using another kernel (e.g. RBF) would have resulted in overfitting on the test dataset. The parameter (C) was chosen by 5-cross validation over the development data. As the training dataset with the above mentioned features was linearly separable, we could not do cross validation on the training data set. So we chose to do the same on the development set. This also helped us in getting a true sense of how our model would perform on the test set as it is guaranteed that the development data set would be similar in distribution to the test data set. We also created a balanced training data set by reconstructing the distribution of article of development set. It did not yield expected results as the points in the feature space were too close to each other.



Average Log-Likelihood with 3 and 4 grams using Balanced Data-Set



Cross Validation Accuracy Curve

4 Results

Classifier Used : Linear Support Vector Machine(SVM)
Parameter Value :
C = 10
Features :
Average log likelihood using 3, 4, 5, 6 and 7 grams

Overall accuracy on Training Set : 99.5%
Overall accuracy on Development Set : 93%

5 Conclusions

1. Bag of word features don't work well with our model as they are unable to account for the contextual meaning of sentences in the data set. Topical modelling could not add any new information to the model due to the poor quality of articles in the development set.

2. The balanced dataset created did not yield the expected results as the points were very close to each other in the feature space.

3. Average log-likelihood performs better than average perplexity as a feature.

4. Linear SVM as a classifier performed better than other classifiers like Random forests etc.

5. The final weights assigned to the features demonstrated the importance of trigrams and 7 grams.

6. Even though the feature space of 3 and 4 grams was yielding good cross validation accuracy, the accuracy on development data set improved after introducing 5, 6, and 7 grams as features.

6 References

[1] Vladimir Vapnik and Corinna Cortes. Support vector networks. Machine Learning, 20:273âĂŞ297, 1995.

[2] Latent Dirichlet Allocation (Blei, David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). Lafferty, John, ed. "Latent Dirichlet Allocation". Journal of Machine Learning Research 3)

[3] Gensim (Latent Dirichlet Allocation) (https://radimrehurek.com/gensim/models/ldamodel.html)

[4] H. Misra and F. Yvon. Using lda to detect semantically incoherent documents. In In Proceedings of CoNLL, 2008.

[5] Statistical Language Modeling Toolkit (http://svr-www.eng.cam.ac.uk/ prc14/toolkit.html)