11-761 Language and Statistics - Spring 2013
Course Project

Leonid (Leo) Boytsov, Zhou Yu, Di Wang, Hector Liu

Abstract

We took a machine learning approach and trained two discriminative classifiers: a soft-margin SVM and a regularized logistic regression. The SVM is used for classification, while the logistic regression is used to produce posterior probabilities. To reduce the risk of overfitting, we generated additional training and testing data.

1 Introduction

Our group decided to focus on getting good performance with respect to the “hard” metric. We employed discriminative classifiers, because they are often superior to generative ones on the classification task. [4, 1]. [1] To reduce the risk of overfitting, we generate additional training data and testing data. The details are given in subsequent sections. The individual contributions of team members are summarized in Table1.

I have simplified here.

Please, revise.

Leo Boytsov	Wrote code to produce features based on n-gram models (including models based on POS-tag sequence), generated additional training/testing data, and fine-tuned machine learning models;
Zhou Yu	Implemented and designed code for classification and computation of posterior probabilities;
Di Wang	Implemented and evaluated additional features based on word co-occurrences and syntactical parsing, and carried out error analysis;
Hector Liu	Helped Leo to generate POS-tag based n-gram models, wrote a reliable RunMe script that feeds language-model features to machine learning code and extracts results.
Everybody	Participated in the design process and described his/her work in the report and the presentation.

Table 1: Individual contributions of team members.

2 Training the Models

2.1 Choosing Features

Because fake text was generated using a tri-gram model, we expected that higher order n-gram models ($n > 3$) would have different perplexities on fake and real data. Thus, perplexity values can be used as document features.

[1] See also `http://xiaodong-yu.blogspot.com/2009/10/good-summary-on-generative-vs.html`.

The n-gram models were created with the help of the CMU-Cambridge toolkit[2]. The Broadcast News corpus was used for training. We employed six n-gram models ($2 \leq n \leq 7$), which were smoothed using the Good-Touring method. For $n > 2$ the models included only n-grams that occurred more than once (the larger is n, the larger is the cutoff value).

In addition, we tried several other feature types:

- n-gram models for sequences of POS tags. To this end, we applied the Stanford POS tagger to the Broadcast News corpus.
- language models based on word co-occurrences inside and across sentences. Specifically, we computed the average Normalized Pointwise Mutual Information (NPMI) [3] for every pair of words that within a sentence that neither stopwords nor width in the distance of 3-gram. Also, we computed the proportion of same word (non-stopwords) that mentioned across sentences.
- syntactical parsing scores (log-likelihood) of best parsing tree for each sentence generated by the Stanford Parser [2].
- the total number of words in a document and the number of words `<UNK>`.

All these additional features were inferior to perplexities computed from the n-gram models. In particular, the accuracy of a classifier built on top of the POS-tag-based language models was less than 70%. Combining POS-tag-based n-gram models with regular n-gram models did not lead to better classification accuracy.

2.2 Generating Additional Training/Testing Data

Because the CMU-Cambridge toolkit cannot generate data from a tri-gram model, we tried to employ the NLTK tookit (using Broadcast News corpus as a training set). Unigrams not included in the tri-gram model (provided by instructors) were replaced by the word `<UNK>`. Real sentences were randomly sampled from the Broadcast News corpus. For both real and fake data, document lengths were chosen randomly to mimic distribution in the training data. Unfortunately, this data proved to be completely useless. The feature values for the generated data and the training set provided by instructors were substantially different.

Then, we found that the CMU Sphinx toolkit[3] included a version of the CMU-Cambridge toolkit capable of generating fake texts. However, we could not make the Sphinx toolkit read the provided binary language model (the `binlm`-file).[4] Nor did the Sphinx toolkit support generation from the text version of the language model (the `arpa`-file). Hence, we back-ported the generating function to the CMU-Cambridge toolkit.

We also took a different approach to generating real data. It appears to us that the Broadcast News corpus contains complete documents, but their boundaries are not marked. To generate a document containing a desired number of words, we, thus, start from the beginning of a random sentence and select a contiguous sequence of words. Thus, generated documents contain long pieces of coherent text (with similar statistical properties). For the reasons explained in section 2.3, on average, a generated document is **2000 words longer** than a document from the provided training set.

How does "similar statistical properties" sound?

I don't like forward references, but otherwise, it is hard to

[2] `http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html`
[3] `http://cmusphinx.sourceforge.net/`
[4] It complained that the model was corrupt.

Original DEV set							
SVM performance (hard metric)				Logistic regression (LASSO) performance (soft metric/$2^{\text{soft metric}}$)			
Original training set		Generated training set		Original training set		Generated training set	
short features	complete features	short features	complete features	short features	complete features	short features	complete features
0.88	0.90	0.92	0.895	-0.51/0.7	-0.46/0.73	-0.41/0.75	-0.4/0.76
Generated DEV set							
SVM performance (hard metric)				Logistic regression (LASSO) performance (soft metric/$2^{\text{soft metric}}$)			
Original training set		Generated training set		Original training set		Generated training set	
short features	complete features	short features	complete features	short features	complete features	short features	complete features
0.93	0.92	0.93	0.9	-0.32/0.8	-0.36/0.78	-0.33/0.8	-0.46/0.73

Table 2: Performance data for various data and feature sets.

2.3 Choosing and Training the Models

For the purpose of classification we use a soft-margin SVM with a linear kernel. liblinear **??** package is adopted in our experiments for its fast speed and reliable performance. To produce posterior probability we train another discriminative model: a regularized logistic regression. Two regularization methods were tested: the LASSO and the Tikhonov regularization. The soft metric cannot be computed, if any probability is zero, hence, posterior probabilities were smoothed. Parameters were chosen empirically (with performance evaluated through 10-fold cross-validation).

Zhou, please, review and expand if necessary.

add some changes

Additionally, we tested the SVM with the RBF kernel and a continuous Naive Bayes (with the normal kernel). The RBF-based SVM was overfitting (despite we tried various margin coefficients and kernel widths). The Naive Bayes also performed badly (for unknown reason).

The observed log-perplexity is an average value taken over log-perplexities of document n-grams. We adopt a point of view that the log-perplexity is sampled from a random variable (or a parametric family of random variables). Thus, the average document log-perplexity is an MLE estimate of this random variable mean. We argue that performance of machine learning models depend on how well they learn parameters of this distribution (or distribution families), in particular their mean values.

Please, proofread this paragraph carefully!

The shorter is a document, the higher is the variance of the mean estimates and, consequently, the harder it is for the model to "guess" true mean values. Hence, if a document length (or a number of words) is not included as a feature, models should produce better results when they are trained on longer documents. In particular, when we train and test on the same DEV set, we get much worse accuracy, then when we train on the training set and test on the DEV set! This is somewhat paradoxical, because the DEV set contains mostly short documents (around 100-200 words), while the training set contains few or no documents shorter than 100 words. Thus, one may consider this training set as being not representative.

3 Experiments

In addition, to the training and DEV sets provided by instructors, we used data generated by our team (see Section 2.2). Our synthetic training and development sets include 10,000 and 2,000

# of sentences.	1	2	3	4	5	7	10	15	20
share in DEV set	20%	10%	10%	10%	10%	10%	10%	10%	10%
Original DEV	0.75	0.9	0.9	0.95	0.95	1	1	1	1
Generated DEV	0.73	0.86	0.86	0.91	0.95	0.96	0.99	1	1

Table 3: Correctness of classification depending on document length

documents, respectively. Our experiments indicate that n-gram based features are superior to other features we tried. In particular, building n-gram models over sequences of POS tags was not helpful.

We employed two sets of n-gram based features. The complete set includes perplexities for six n-gram models, but the short set employs only the 3-gram and the 4-gram models. When, we use only the provided training data, it does not hurt to include all features into the SVM model (see Table 2). As we get more training data, it may be better to use only the 3-gram and the 4-gram models.

As we can see from Figure 1, in the training set, these two models alone separate the fake and real documents almost perfectly. Note, though, that for a development set, the "clouds" of real and fake documents have a considerable overlap. As explained in Section 2.2, this is due to the fact that the training set contains only few short documents and, consequently, perplexity values have little variance due to random effects.

Di, did I get this correctly on the overlap for the DEV data?

We also believe that there may exist a **latent variable** that determines the complexity of the document lexicon. If the document contains a lot of rare words, both the 3-gram and the 4-gram models would be surprised to see it. In contrast, documents with a lot of common words have low perplexity values for all n-gram models. The value of document perplexity with respect to any n-gram model varies considerably and none of the model is sufficient to detect fake sentences. Yet, this is possible with two models.

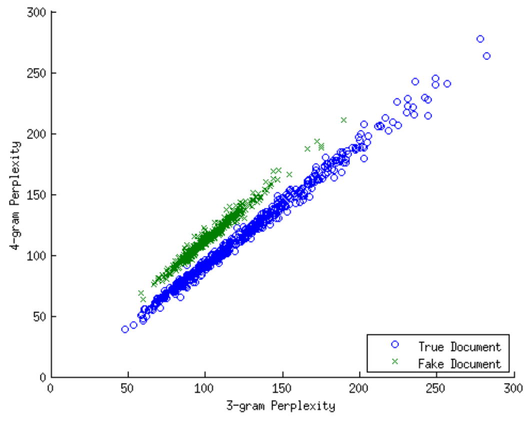
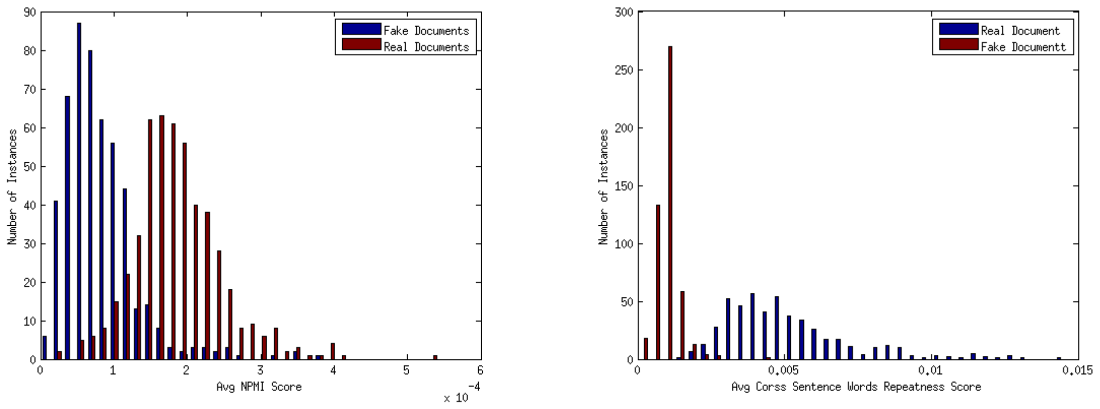


Figure 1: Perplexities for 3- and 4-gram models computed for the **training** set.

Our classifier performs better on long sentences (see Table 3). For document containing only one sentence, we can detect fake sentences with $\approx 75\%$ accuracy for both the original and generated DEV sets. Exactly half of all documents are fake. Hence, we are doing much better than random guessing (by flipping a fair coin), which would get correctly only 50% of documents. For longer documents, the accuracy is mostly higher than 95%. Yet, because documents having 1-2 sentences comprise 30% of all DEV documents, it is hard to get the overall accuracy higher than 90%.

Guys, do you agree with this estimate?

Unlike the classification task, it is less clear whether the complete set of features is better or worse than the short set of features. According to Table 2, the complete set of features performs better with a small training set. For the generated training set, the complete set of features is better only for the original DEV set, but the complete set performs worse for the generated DEV set. Nevertheless, we chose the complete set, because the generated set was created by us and might have been less representative of the unseen test data. Note that Table 2 includes data only



(a) Avergage NPMI (b) Avergage Corss-Sentences Words Repeatness

Figure 2: Histogram of the Avergage Normalized Pointwise Mutual Information (NPMI) and Avergage Corss-Sentences Words Repeatness score computed over the training set.

for the LASSO regularization, because the Tikhonov regularization was somewhat inferior (and is not presented here).

4 Conclusions

We took a machine learning approach and trained two discriminative classifiers. The soft-margin SVM turned to be the best method for the classification task, while the logistic regression with the LASSO regularization was the best method to estimate posterior probabilities. We also found that simple features: perplexities computed from n-gram models were sufficient to achieve good performance. All other approaches failed to produce better results. Interestingly enough, no single n-gram model is sufficient to distinguish real from fake sentences. Yet, it is possible with perplexities computed only for the 3-gram and the 4-gram model. In that, the training data is (almost) linearly separable. Finally, we note that additional training data (generated by our team) allowed us to achieve better results on the DEV set, but it is not clear, of course, how well we perform on unseen data.

References

[1] Christopher M Bishop, Julia Lasserre, et al. Generative or discriminative? getting the best of both worlds. *Bayesian Statistics*, 8:3–23, 2007.

[2] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *IN PROCEEDINGS OF THE 41ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, pages 423–430, 2003.

[3] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[4] R. Rosenfeld. CMU course 11-761 language and statistics, 2013.