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. [1, 4]. [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.

Leo Boytsov	Wrote code to produce features based on n-gram models (including models based on POS-tag sequences), 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

Fake documents were generated using a tri-gram model, i.e., they include three consecutive words. Hence, we expected that features computed over longer text spans would be useful in distinguishing between fake and real documents. In particular, we employed perplexities computed using higher order n-gram models ($n > 3$).

[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]. We took into account every pair of words in a sentences (stopwords excluded) that were separated by at least 2 words.
- A number of words (normalized to the total word number) that were mentioned in more than one document (stopwords were excluded).
- Syntactical parsing scores (log-likelihood) of the best parsing tree (for each sentence) generated by the Stanford Parser [2].

All these additional features were inferior to perplexities computed using the n-gram models.

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 text-generating function from the CMU Sphinx to the CMU-Cambridge toolkit.

We also changed an approach to generate real documents. 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 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.

[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 used a soft-margin SVM with a linear kernel. More specifically, we employed the package LIBLINEAR, which is very efficient and reliable.[5] To produce posterior probabilities we trained 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). Additionally, we tested the SVM with the RBF kernel and a continuous Naive Bayes (with the normal kernel), but none of them worked well (RBF overfit).

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.

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 could produce better results when they are trained on longer documents. These documents cannot be excessively long, because long spans of text in the Broadcast Corpus may contain multiple documents with very different properties.

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.

[5] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

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 documents, respectively.

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.

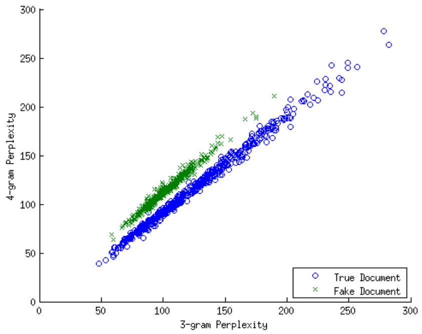


Figure 1: Perplexities for 3- and 4-gram models computed for the original **training** set.

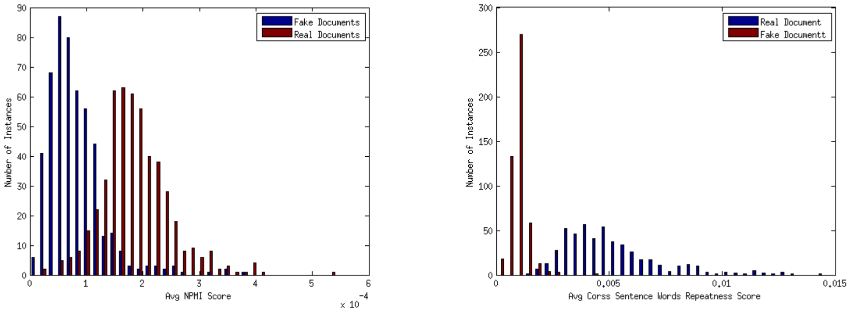
As we can see from Figure 1, in the training set, these two models alone separate the fake and real documents almost perfectly. Note, though, that for a development set, the "clouds" of real and fake documents have a considerable overlap. As explained in Section 2.2, this is due to the fact that the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

There apparently exists 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 n-gram model is sufficient to detect fake sentences. Yet, this is possible with two models.

Other types of features were less useful in separating fake and real documents. For example, the point-wise mutual information and the average proportion of words that repeat across sentences are generally different for real and fake documents, but there is also a large overlap (see Figure 2). 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.

Our classifier performs better on long sentences (see Table 3). For document containing only one sentence, we can detect fake sentences with ≈ 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%. For short documents, there are fewer co-occurring words and there is little or no information in regard to words repeating across document sentences. Syntactical parsing scores were also useless in detecting short long sentences (and were useless for longer ones as well).

Unlike the classification task, it is less clear whether the complete set of features is better or worse than the short set of features for the purpose of producing posterior probabilities. 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 on the original DEV set, but it performs worse on 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



(a) Average NPMI

(b) Average Proportion of Words that Repeat Across Sentences

Figure 2: Distribution of the Average Normalized Pointwise Mutual Information (NPMI) and of the average proportion of words that repeat across sentences.

# 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

Table 2 includes data only for the LASSO regularization, because the Tikhonov regularization was somewhat inferior (and is not presented here).

4 Conclusions

We found that simple features: perplexities computed using 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 between real from fake sentences. Yet, detection is possible and accurate if two models are used (especially the 3-gram and the 4-gram model). The n-gram models did not work very well for short sentences, but neither did other types of features. Additional training data (generated by our team) allowed us to achieve better results on the DEV set. Yet, the improvement was marginal and might have been due to random effects.

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.