Classification of Fake and Real Articles Based on Support Vector Machines

Language and Statistics
Spring 2013

Justin Chiu, Ajda Gokcen, Wenyi Wang, Xiaohua Yan

1 Introduction

Fake or real? That is the question, even in the context of languages. In this course project, we are given the task of distinguishing real Broadcast News articles from fake "articles" generated by a trigram model trained from the 100 million word corpus of Broadcast News articles from 1992–1996. This task is clearly not difficult for humans, while machines are not as smart as us to tell whether the articles make sense at higher levels such as semantics. In this report we presented a machine-oriented solution based on Support Vector Machines [1] to classify between fake and real articles. Features deployed in our task covered various aspects of languages. For example, we tried n-gram language models and parsers to generate features that capture the local dependencies and syntactical information of the texts in the data sets. The topic model based on Latent Dirichlet Allocation (LDA) [2] was used in an attempt to distinguish the articles at the semantic level. With all these features and a good classifier (namely 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 Balancing the training set

At the very start of our project, we noticed that the distributions of number of sentences in each article for the training set and the development set are very different, while experience told us that most machine learning algorithms ask for the training and testing samples to come from same or similar distributions. In order to accomodate for the difference, we utilized a simple algorithm proposed by [3] to balance the training set, so that the distribution of numbers of sentences in each article is identical to that of the development set.

The algorithm works iteratively, which adds a new article to the new training set from a pool of partial articles until there are no more candidate articles to add. As a result, we obtained a new training set comprising 7,239 articles, which is the basis of all our later experiments.

Actually, we made a comparison on both the original and balanced training set with respect to the classification accuracy on the development set using the features proposed in the next section. And results showed that the accuracy indeed improved on the balanced training set. Therefore we believe the balancing step is helpful for our task.

3 Feature extraction

In general, the feature we extracted can be put into three categories: statistical features, syntactical features and semantic features. In this section, we will give a detailed discussion about the procedures of feature extraction.

3.1 Statistical features

3.1.1 Type-token ratios

Recall that in Assignment 1 of this class, we observed that different writing styles may have distinct type-token ratio curves. And in this project, the tri-gram model can be well regarded as a special "writing style". So we calculated the ratio between the types and tokens for each article, as a feature to capture this kind of long-range dependency among words. This idea can also be justified by observing that a word that appeared once in an article is more likely to appear again in the same article. Note that because the ratio will get lower as the article gets longer, the type-token ratios are normalized by the length of the corresponding article. And the length of an article is also incorporated as a basic feature in the final feature set.

3.1.2 Lexical entropy

It has been suggested that the amount of structure diversity in an author's writing can be measured by lexical entropy [4,5]. The lexical entropy H of an article can be defined as follows:

$$H = -\sum_{i} \frac{iV_i}{N} \log \frac{iV_i}{N}$$

where N is the total number of tokens in the sample and V_i is the number of types that appear exactly i times. Besides, many researches in the literature have attempted to measure the richness of an author's vocabulary, for which we computed the standard type-token ratio. Besides, many of them have also postulated that an author's writing style is reflected in *hapax legomena* and *hapax dislegomena* [5]. We computed both measures (i.e. the ratios of V_1 and V_2 to V where V is the total number of types) for all the articles in the data sets as two important features at the lexical level.

3.1.3 Error features

There is a noticeable error in the fake article that some words may appear repeatedly in a row while this would hardly happen in authentic articles. In addition, several grammatical errors in the fake articles. Such features involve distances between wh-words (e.g. *who* and *what*) and distances between stop words [3]. We used a stop word list of more than 400 words consisting mostly of function words and closed-class words. All the error features addressed here were first normalized by the total number of occurrences (be it duplicated words, distances between wh-words, or distances between stop words). And the final values were normalized by the number of sentences in the article.

3.1.4 Longer n-grams

Since the fake articles are generated according to the tri-gram model. One intuition is that longer n-grams (that is, 4-gram and longer) would be effective in distinguishing between real and fake articles. Therefore, we also investigated more complex language models from 4-gram to 7-gram trained on the 100 million token corpus. And we used the cross entropy of each model on both the training and development set as 4 statistical features.

3.2 Syntactical features

When humans make the judgment about whether an article is fake or real, one important criterion is whether the article has syntactical errors. This led us to look for syntactical features of the articles. Specifically, we used the Berkeley parser [6] to turn the original sentences into syntactical trees so we could use any useful grammatical features.

3.2.1 Extracting useful syntactical information

In order to figure out what information was useful, we kept track of several statistics over the training set. First, we split the training data into real and fake sets. We then calculated distributions of overarching structures for the sentences: "S," "SBARQ," "X," and so on. We also kept track of the relative amount of sentences lacking important structures and parts of speech.

Some structures occurred with consistently different frequencies in the real and fake data. The most consistent features were the fraction of sentences labeled "S," "X," "NP," and "INTJ." The fractions of sentences without verbs and noun phrases were consistent within each set, as well.

3.2.2 Combining syntactical features

The six features deemed most useful were then assigned parameters based on observed differences between the real and fake text. If the observed value was greater than the parameter, then the article would most likely be real or fake, depending on the values observed in training. For instance, the parameter for the fraction of lines labeled as 'S' was .65; above this, the article is most likely real. Below it, the article is most likely fake. We assigned a probability of the article being real or fake given each parameter, with more balanced probabilities for observed values closer to the parameter.

Once we calculated the real and fake probabilities given each parameter, we combined them into two final real and fake probabilities via interpolation, with each weighted according to importance and consistency. We ended up with eight feature values: the six syntactical observations and the two final probabilities.

Unfortunately, the first six features were not very good with shorter articles of one or two sentences. Thus, we added article length as a ninth feature so that SVM would be able to pick up on the usefulness of the features for longer articles.

3.2.3 Scope and limits

The results of these features alone were respectable, with roughly 85% accuracy for real data. For fake data, on the other hand, it wasn't much better than 50%. When working with general data, this came out to around 65-70%, which could have proven useful when combined with other features.

The Berkeley parser, however, took an extremely long time to generate syntax trees for a sizable set, such as the balanced set of over seven thousand. So, due to time and processing constraints, the syntactical features did not make it into our final model. We saw that they were useful, but simply not practical.

3.3 Semantic features

It is common knowledge that n-gram models do not capture long-distance dependencies between words in the article. In order to exploit this fact, we designed several semantic features to capture long-distance dependencies between words.

3.3.1 Topic modeling

Latent Dirichlet Allocation (LDA) [2] is a well-designed generative probabilistic topic model for collections of discrete data such as text corpora, the basic idea of which is that documents (articles) are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

Using the MALLET toolkit [7] which uses LDA, we tried to identify the latent topics among the articles as an important feature that characterize the semantic information. The assumption behind our method of extracting topic modeling features is that the topics of words in the article are likely to be more focused than that of fake articles. For example, below is a list of key words of one topic.

- DEATH CRIME PRISON LAW CASE VICTIMS RAPE MAN VICTIM SYSTEM JAIL LIFE MURDER CRIMINAL JUDGE CRIMES VIOLENT CASES JUSTICE

Specifically, we used the toolkit to identify 100 topics and the 20 most frequently occurring words for each of these topics. Then, we calculated two kind of features based on what were proposed in [3]:

- *topicFraction(t)*: For each topic, we calculated the fraction of words in the article that belong to that topic.
- *topicCover(t)*: For each topic t, we calculated the fraction of the 20 most common words for that topic that are covered by the article.

This led to 200 features for each article in both the training and development data sets.

Although the authors of [3] already stated that these features yield relatively poor classification accuracy on the development set, it still surprised us when the features we obtained produced an accuracy of merely around 60%. At first we thought this was because the topics as well as the related common words were not representative enough to distinguish between fake and real articles. So we turned to the large 100 million token corpus in order to generate an more informative topic-word lists. In particular, since the large corpus is not organized in the article level, we randomly split the corpus into articles according to the length distributions of the training and development data. With the new topic-word lists, the same procedure of extracting features was performed. However, this still did not yield better results. We believe this amounts to the fact that some words in the topic-word lists are somewhat misleading, and the large dimensionality undermines the performance of the features when they are combined with other stronger features.

3.3.2 Triggers network

Mutual information is a common evaluation for the inter-word relations. This feature basically rely on the observation that a certain word occurred in the article may “trigger" the occurrences of some other words in the same article. For example, the probability of observing the word “baseball" increases when the word “sport" is also present in the article.

The differences of such conditional probabilities can be captured by defining the article as a *triggers network*, which is an idea inspired by [3]. To be specific, a triggers network is an undirected weighted graph. The nodes in the graph are non-function words in the article, and for words w_i and w_j, there is an edge if $L \leq |w_i - w_j| \leq U$ where $|w_i - w_j|$ denotes the distance between w_i and w_j and L and U stand for the lower and upper bounds of distances to be considered in order to reduce computational complexity. The weights between the words represent how semantically linked two words are, which can be obtained by calculating the point-wise mutual information based on the joint distributions of words on articles calculated from the large Broadcast news corpus. More specifically, the point-wise mutual information of w_i, w_j can be written as [8]

$$f^{pmi}(w_i, w_j) = \log_2 \frac{f^b(w_i, w_j) \times m}{f^t(w_i) f^t(w_j)}$$

where $f^t(w_i)$ tells us how many times the type w_i appeared in the entire corpus, and $f^b(w_i, w_j)$ tells us how many times the word w_i appeared with word w_j in the context windows described above. Here m stands for the total number of tokens in the corpus.

4 Results

The final feature set we used include statistical features and triggers network features, which overall have 24 dimensions. We chose to use the LIBSVM [9] package for building the model as well as outputting the probability estimates. The resultant model used a RBF kernel which parameters $C = 32$ and $\gamma = 0.125$. The parameters were chosen by cross validation. And the classification accuracy and average log perplexity of each individual type of feature along with the overall feature set on the **development set** is reported in the following table. Also, the overall accuracy and average log-perplexity on the training set is 92.71% and -0.1279 respectively.

Feature Set	Accuracy	Avg log-perplexity
Statistical features	86.5%	-0.25962
Semantic features	87%	-0.28401
Overall	95%	-0.13039

5 Comments and suggestions

As can be observed, the classification results produced by our program is quite satisfying. Working on this project is really a memorable experience. Though the task seemed simple at the first glance, we encountered more and more problems as we push for better results. And we are glad that we finally achieved decent results that put us in the candidate place for the chocolate truffles. And most importantly, we learned a lot while working as a team and we are now more confident dealing with languages and their statistics!

One interesting finding is that the final model made most of its errors on short articles, which is not surprising since short articles may not contain enough information for computers to classify. This led us to think of ways to improve the classification power of our program on short articles. Possible solutions may include using different models for articles of different lengths. Furthermore, we could make a more careful design of features that may be exclusively used on short articles.

Author contributions: we designed the feature extraction procedures all together. J.C. was responsible for *triggers network*. A.G. did the syntactical features. W. W. was responsible for statistical features. And X.Y. was responsible for *topic modeling*. The authors also wrote the scripts of their own parts. J.C., W.W and X.Y. wrote the final scripts that merged the results. All the authors contributed to the write-up of the report.

References

[1] Vladimir Vapnik and Corinna Cortes. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] Eric Davis, Jason Adams, and Shay Cohen. Classifying articles as fake or real. *Language and Statistics course project*, 2007.

[4] David I. Holmes. Authorship attribution. *Computers and the Humanities*, 28(2):87–106, 1994.

[5] Neil Graham, Graeme Hirst, and Bhaskara Marthi. Segmenting documents by stylistic character. *Nat. Lang. Eng.*, 11(4):397–415, December 2005.

[6] A natural language parser from UC Berkeley. `http://code.google.com/p/berkeleyparser`.

[7] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. `http://mallet.cs.umass.edu`, 2002.

[8] A. Islam and D. Inkpen. Second order co-occurrence pmi for determining the semantic similarity of words. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*, pages 1033–1038, 2006.

[9] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.