— KWARC Blue Note* —
Paragraph Discrimination using Bag of Words Model

Jan Frederik Schaefer

July 18, 2014

Abstract

A significant subset of the *arXiv* documents has marked up paragraphs, stating whether they are theorems, axioms, proofs etc. We want to use these as a training set to be able to discriminate the remaining paragraphs. In this sketch, I want to describe an approach which is similar to some spam detection techniques.

1 The math

The idea of this approach is based on some concepts from `http://en.wikipedia.org/wiki/Bayesian_spam_filtering`. We have a (small) set of paragraph types $T = \{\text{axiom}, \text{theorem}, \text{proof}, \ldots\}$. We represent paragraphs as a set of the words W it contains, ignoring their order or their number of occurences. Now we are looking for the probability $P(T_i|W)$ that a paragraph with words W is of type T_i. Using Bayes' theorem, we get

$$P(T_i|W) = \frac{P(W|T_i) \cdot P(T_i)}{P(W)}$$

Apparently, we would get a bias towards the more common types T_i. Probably, we should remove the bias by using a uniform distribution for $P(T_i)$. Some authors didn't use the mark up consistently, and marked up e.g. only theorems, so we might a wrong bias. Clearly, we have

$$P(W) = \sum_i P(W|T_i) \cdot P(T_i)$$

Our goal is to determine $P(T_i|\mathbb{W})$, where $\mathbb{W}$ is the set of words contained in a paragraph. There is a way to combine the probabilities of individual types of words [1]:

$$P(T_i|\mathbb{W}) = \frac{\prod_{w\in\mathbb{W}} P(T_i|w)}{\left(\prod_{w\in\mathbb{W}} P(T_i|w)\right) \; + \; \prod_{w\in\mathbb{W}} (1 - P(T_i|w))}$$

*Inspired by the "blue book" in Alan Bundy's group at the University of Edinburgh, KWARC blue notes, are documents used for fixing and discussing ϵ-baked ideas in projects by the KWARC group (see `http://kwarc.info`). Unless specified otherwise, they are for project-internal discussions only. Please only distribute outside the KWARC group after consultation with the author.

[1] see `http://en.wikipedia.org/wiki/Bayesian_spam_filtering#Combining_individual_probabilities`

1.1 Optimizing computation

We have

$$P(T_i|\mathbb{W}) = \frac{\prod_{w\in\mathbb{W}} P(T_i|w)}{\left(\prod_{w\in\mathbb{W}} P(T_i|w)\right) + \prod_{w\in\mathbb{W}} (1 - P(T_i|w))}$$

$$\Leftrightarrow \qquad \frac{1}{P(T_i|\mathbb{W})} - 1 = \frac{\prod_{w\in\mathbb{W}} (1 - P(T_i|w))}{\prod_{w\in\mathbb{W}} P(T_i|w)}$$

Additionally, we can use logarithms to turn the multiplication into addition:

$$\log\left(\frac{1}{P(T_i|\mathbb{W})} - 1\right) = \left(\sum_{w\in\mathbb{W}} \log\left(1 - P(T_i|w)\right)\right) - \left(\sum_{w\in\mathbb{W}} \log P(T_i|w)\right)$$

2 Things we should consider

2.1 Normalization

Since we have a limited set of marked up paragraphs, we might get better results if we stem words.

2.2 Using only helpful words

Some words might be more useful to consider than other words. E.g. the word *the* wouldn't help us discriminating paragraphs, whereas *follows* might be very helpful. So ignoring the words which do not really hint at any type might yield clearer results.

2.3 Skipping rare words

Rare words can have devastating effects (strong noise, even division by zero problems)

2.4 Using actual frequencies

The math would get more complicated, but it might be very helpful to use the actual number of occurences of a word in a paragraph, not just the binary *is contained* vs. *is not contained*.

2.5 Using ngrams

So far, we completely ignored the order of words. It might be worth to consider using bigrams instead of single words. For larger ngrams the training set probably isn't large enough.