arXiv vs. snarXiv

Tyler Blanton **Sam Kowash**

Abstract

We attempt to construct classifiers that can accurately distinguish between real arXiv `hep-th` abstracts and fake abstracts generated from a context-free grammar by the program snarXiv[1]. We study the bag-of-words and n-grams language models as well as several different classifiers: naive Bayes, likelihood-ratio, and tf-idf. We find the likelihood-ratio and tf-idf methods to perform extremely well, classifying abstracts correctly with nearly 100% accuracy.

1 Introduction

2 Language models

2.1 n-grams

Assume that each abstract X has a probability $\mathbb{P}(X|Y)$ of being from source $Y \in \{-1, 1\}$, where $Y = 1$ represents the label "snarXiv" and $Y = -1$ represents the label "arXiv." Suppose $X = (w_1, w_2, \ldots, w_N)$ is an abstract containing N words $\{w_i\}_{i=1}^N$, and let w_i^{i+k} denote the word sequence $(w_i, w_{i+1}, \ldots, w_{i+k})$. The n-grams language model approximates $\mathbb{P}(X|Y)$ as

$$\mathbb{P}(X|Y) = \mathbb{P}(w_1^N|Y) = \prod_{i=1}^{N} \mathbb{P}(w_i|w_1^{i-1}, Y) \tag{1}$$

$$\approx \prod_{i=1}^{N} \mathbb{P}(w_i|w_{i-n+1}^{i-1}, Y) \quad (n\text{-grams approx.}) \tag{2}$$

In other words, in the n-grams model we assume that the probability of a given word w occurring in some abstract only depends on what the previous $n-1$ words in the abstract are, where n is some (typically small) positive integer; the $n-1$ words on which w depends are called the "context" of w.

In the simplest case $n = 1$, each word occurrence is assumed to be completely independent of any context, which is why the 1-gram language model is often referred to as the "bag-of-words" (BoW) model. Given some training set of abstracts from source Y, we define our BoW training estimate $\widehat{\mathbb{P}}(X|Y)_{\text{BoW}}$ as

$$\widehat{\mathbb{P}}(X|Y)_{\text{BoW}} = \widehat{\mathbb{P}}(w_1^N|Y)_{\text{BoW}} \equiv \prod_{i=1}^{N} \widehat{\mathbb{P}}(w_i|Y), \tag{3}$$

$$\text{where} \quad \widehat{\mathbb{P}}(w_i|Y) \equiv \frac{C_Y(w_i) + 1}{\sum_w [C_Y(w) + 1]} = \frac{C_Y(w_i) + 1}{\sum_{w \in \mathcal{V}} C_Y(w) + V}, \tag{4}$$

$C_Y(w_i)$ is the number of times that w_i appears, $\mathcal{V}$ is a vocabulary of size $V = |\mathcal{V}|$ containing all of the words that we wish to learn about, and $\sum_{w \in \mathcal{V}} C_Y(w)$ is the total number of words in the training set. The extra $+1$ on each word count $C_Y(w)$ comes from Laplace smoothing, which is a

[1] `http://davidsd.org/2010/03/the-snarxiv/`

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

precautionary measure taken to prevent zero probabilities from occurring in cases where we need to classify an abstract that contains a word that did not appear in the training set. Other technical details regarding bag-of-words and n-grams implementation (including parsing) can be found in Appendix A.

The generalization to $n > 1$ is straightforward: we define our estimate $\widehat{\mathbb{P}}(X|Y)_{n\text{-gram}}$ for a given training set from source Y to be

$$\widehat{\mathbb{P}}(X|Y)_{n\text{-gram}} = \widehat{\mathbb{P}}(w_1^N|Y)_{n\text{-gram}} \equiv \prod_{i=1}^{N} \widehat{\mathbb{P}}(w_i|w_{i-n+1}^{i-1}, Y), \tag{5}$$

$$\text{where} \quad \widehat{\mathbb{P}}(w_i|w_{i-n+1}^{i-1}, Y) \equiv \frac{C_Y(w_{i-n+1}^{i}) + 1}{C_Y(w_{i-n+1}^{i-1}) + V}, \tag{6}$$

$C_Y(w_{i-n+1}^{i})$ is the number of times that the n-gram w_{i-n+1}^{i} occurs, and $C_Y(w_{i-n+1}^{i-1})$ is the number of times that the $(n-1)$-gram w_{i-n+1}^{i-1} appears.

Note that since a typical arXiv/snarXiv abstract contains roughly 100 words, so the abstract probability $\mathbb{P}(X|Y) = \prod_{i=1}^{N} \mathbb{P}(w_i|w_1^{i-1}, Y)$ is a product of $N \approx 100$ different word probabilities, each of which is typically much less than one. As a result, the size of $\mathbb{P}(X|Y)$ is heavily dependent on the abstract length and is usually extremely small, causing our estimates to hit numerical underflow in some cases. To combat the issue of length dependence, we prefer to work with the perplexity

$$\text{PP}(X|Y) \equiv \sqrt[N]{\frac{1}{\mathbb{P}(X|Y)}} = \left[\prod_{i=1}^{N} \mathbb{P}(w_i|w_1^{i-1}, Y)\right]^{-1/N}, \tag{7}$$

which normalizes the abstract probability by taking the geometric mean of the individual word probabilities (and inverting). To prevent numerical underflow from rendering our algorithms useless, we always work with logs of probabilities in our code, which turns products into sums and keeps the numbers reasonable. Note that the log of the perplexity is a simple arithmetic mean:

$$\log \text{PP}(X|Y) = -\frac{1}{N} \log \mathbb{P}(X|Y) = -\frac{1}{N} \sum_{i=1}^{N} \log \mathbb{P}(w_i|w_1^{i-1}, Y). \tag{8}$$

In practice, we use the logs of the classification conditions given below in Section 3 to classify our test abstracts as arXiv or snarXiv; we only exponentiate back to perplexities for the purposes of figures.

2.2 tf–idf

3 Classifiers

3.1 Naive Bayes classifier

Our first attempt at creating a program that distinguishes between arXiv and snarXiv abstracts utilized a naive Bayes (NB) classifier. The theory behind this classifier is simple: if $\mathbb{P}(Y = 1|X) > \mathbb{P}(Y = -1|X)$, then we classify X as snarXiv; otherwise, we classify it as arXiv. We can use Bayes' theorem $\mathbb{P}(Y|X) = \frac{\mathbb{P}(Y)\mathbb{P}(X|Y)}{\mathbb{P}(X)}$ to rewrite this classification condition as

$$\widehat{Y}_{NB} \equiv \arg \max_{Y \in \{-1,1\}} \mathbb{P}(Y)\mathbb{P}(X|Y). \tag{9}$$

We estimate each probability in (9) by training on a large number of arXiv and snarXiv abstracts. We define our estimate[2] for $\mathbb{P}(Y)$ as the number of abstracts in training set Y over the total number of abstracts in both the arXiv and snarXiv training sets. The conditional probabilities $\mathbb{P}(X|Y)$ are more complicated, and are approximated using a language model from Section 2.

[2]The probabilities $\mathbb{P}(Y = \pm 1)$ are somewhat trivial since we generate the snarXiv abstracts ourselves and can control over how many arXiv vs. snarXiv abstracts will be in the train and test sets. Since the snarXiv was originally created to make abstracts that compete one-on-one with arXiv abstracts, we chose to set $\mathbb{P}(Y = \pm 1) = \frac{1}{2}$ for all tests reported in this paper.

3.2 Likelihood-ratio test

A slightly more sophisticated classification rule is the likelihood-ratio (LR) test. For a given abstract X, define the likelihood ratio $\Lambda(X) \equiv \frac{\mathbb{P}(X|Y=1)}{\mathbb{P}(X|Y=-1)}$. While we certainly want our classifier to correctly label all snarXiv abstracts as fakes, we also want to minimize the probability that an arXiv abstract gets incorrectly classified as fake. The Neyman-Pearson lemma states that the optimal classifier

$$\delta^*_{LR} \equiv \arg\max_{\delta} \mathbb{P}(\delta(X) = 1|Y = 1), \text{ subject to } \mathbb{P}(\delta(X) = 1|Y = -1) \leq \alpha \tag{10}$$

for any fixed false-positive tolerance α takes the form

$$\mathbb{P}\left(\delta^*_{LR}(X) = 1\right) = \begin{cases} 1, & \Lambda(X) > \eta \\ \gamma, & \Lambda(X) = \eta \\ 0, & \Lambda(X) < \eta, \end{cases} \tag{11}$$

where η and γ can be treated as hyperparameters to be tuned in cross-validation. In practice, γ is almost completely irrelevant since for any η, the odds that a language model like n-grams would yield an estimate for $\Lambda(X)$ that is *exactly* equal to η are essentially zero.[3] We arbitrarily use $\gamma = 0.5$, meaning the only hyperparameter we tune is η. Because of the issues of small $\mathbb{P}(X|Y)$ probabilities discussed in Section 2.1, it is more practical to instead compare the perplexities $\text{PP}(X|Y) = \mathbb{P}(X|Y)^{-1/|X|}$. The relevant parameter in the LR-test is thus $\eta^{1/|X|}$. We would like our hyperparameter to not depend on the specifics of the input though, so we note that the average abstract length is about 120 words and define $\eta_{\text{PP}} \equiv \eta^{1/120}$ to be our new LR-hyperparameter.

4 Results

We report results for three different algorithms: bag-of-words model with naive Bayes classifer (BoW-NB), bag-of-words model with likelihood-ratio test (BoW-LR), and bigrams=(2-grams) model with likelihood-ratio test (bi-LR). In each case, we trained on an equal number $N^Y_{\text{train}} = 1000$ of arXiv and snarXiv abstracts, and we also tested on 1000 arXiv abstracts and 1000 snarXiv abstracts.

For BoW-NB, we found the classification accuracy to be 77%, with the only misclassifications being false positives (arXiv abstracts that the classifier mistakes as snarXiv); when a snarXiv abstract is tested, it is correctly identified as snarXiv 100% of the time. This is somewhat expected for a classifier which does nothing to constrain the false-positive rate.

For BoW-LR and bi-LR, we tuned the LR-hyperparameter η_{PP} through cross-validation to maximize the classification accuracy on abstracts from both arXiv and snarXiv. We found $\eta_{\text{PP}} = 0.7$ to perform best in both models; Figure 1 shows histograms of our estimates for $\frac{\text{PP}(X|\text{snarXiv})}{\text{PP}(X|\text{arXiv})}$ for both the arXiv and snarXiv test corpora. The LR-classifier is clearly distinguishing between the two sets, giving a classification accuracy of 100%.

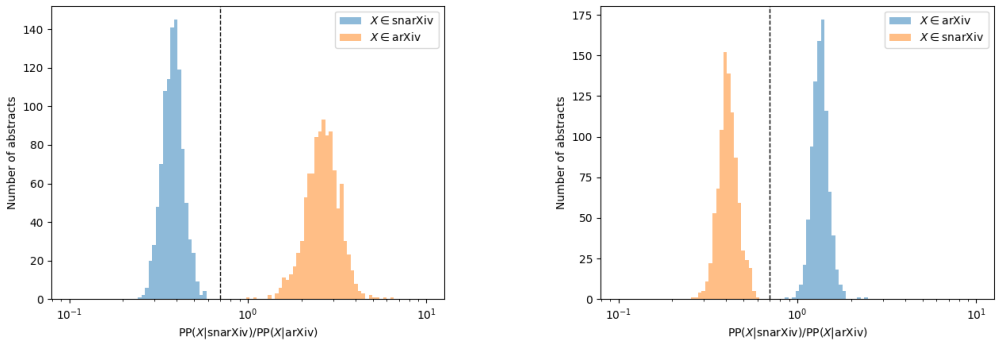
5 Conclusions

A Technical details

A.1 Parsing abstract text

Our parser attempts to translate the raw abstract text (a string) to a list of formatted words that are easily identifiable between different abstracts. We first split the text on all spaces, newlines, and hyphens, and for uniformity we make all words lowercase. Next, we look for words that end in a period or question mark and insert the special word tokens `<e>` and `<s>` afterwards to mark the end of one sentence and the start of another. Lastly, we strip out all punctuation (except for the special tokens), prepend `<s>` to the list (marking the start of the first sentence), and delete the last word if the list ends `<e> <s>` (otherwise, we append `<e>`).

[3] The borderline case of the naive Bayes classifier where $\mathbb{P}(Y = \pm 1|X) = \frac{1}{2}$ is irrelevant for the same reason.



(a) BoW-LR model (b) bi-LR model

Figure 1: Perplexity ratio histograms for both LR-models. The dotted black line corresponds to $\eta_{\text{PP}} = 0.7$.

An example of parser input/output is

Input: `"I'm taking a CSE class on machine-learning. It's a lot of work, but pretty interesting."`

Output: `[<s>, im, taking, a, cse, class, on, machine, learning, <e>, <s>, its, a, lot, of, work, but, pretty, interesting, <e>]`,

where each "word" in the output list is a string.

Our parsing scheme works well, but it leaves traces of some TeX commands from the raw abstract text; for example, it sends `\mathbb{Z}` $\rightarrow$ `mathbbz`. This would be fine if all arXiv and snarXiv abstracts have TeX commands written the same way, though it is possible in some cases for abstracts to write the same command in different ways. In practice, we find that these instances are rare enough for the concern to be negligible.

A.2 Vocabulary

We defined our vocabulary $\mathcal{V}$ as the collection of $V = 15,315$ unique words that appeared at least twice in a training corpus of 12,000 randomly chosen parsed arXiv abstracts and 12,000 (randomly generated) parsed snarXiv abstracts. We decided not to include the 13,960 words that only appeared once since they seem to be relatively uncommon (often they are the residuals of TeX commands), and including them would nearly double our vocabulary size. This pruning can be thought of as a feature-processing step.

A.3 n-grams

It is fairly common for a word to appear in the test set that is not in the vocabulary. When this happens, we change the word to a special token `<UNK>`. We then treat `<UNK>` as we would any other word when counting n-gram occurrences $C_Y(\cdot)$.

We prepend copies of the start sentence token `<s>` to each parsed abstract to ensure that the first n-gram consists of $n-1$ copies of `<s>` and one "real" word. Similarly, we append copies of `<e>` to the end of each parsed abstract.

References

Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining Text Data*. Springer, Boston, Massachusetts, 2012. URL `https://doi.org/10.1007/978-1-4614-3223-4_6`.

William Cavnar and John Trenkle. N-gram-based text categorization. 05 2001.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1): 53–65, 2018.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015. URL `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin, 1998.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel J Hsu. A spectral algorithm for learning class-based n-gram models of natural language. In *UAI*, pages 762–771. Citeseer, 2014.

Karl Stratos, Michael Collins, and Daniel Hsu. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1282–1291, 2015.

Shrawan Kumar Trivedi. A study of machine learning classifiers for spam detection. In *Computational and Business Intelligence (ISCBI), 2016 4th International Symposium on*, pages 176–180. IEEE, 2016.