Authorship Detection with PPM

Notebook for PAN at CLEF 2013

Victoria Bobicev

Technical University of Moldova
victoria_bobicev@rol.md

Abstract. This paper reports on our work in the PAN 2013 author identification task. The task is to automatically detect the author of the given text having small training sets with known authors. The task was solved by a system that used the PPM (Prediction by Partial Matching) compression algorithm based on an n-gram statistical model.

1 Introduction

With the emergence of user-generated web content, text author profiling is being increasingly studied by the NLP community. Various works describe experiments aiming to automatically discover hidden attributes of text which reveal author's gender, age, personality and others. Authorship identification is an important problem in many areas including information retrieval and computational linguistics.

While a great number of works have presented investigations in this area there is need for a common ground to evaluate different author recognition techniques. PAN 2013 as part of the CLEF campaigns aims to provide the common conditions and data for this task.

We participated in this shared task with the PPM (Prediction by Partial Matching) compression algorithm based on a character-based n-gram statistical model.

2 Model description

Authorship identification can be viewed as a type of classification task. Such tasks are solved using learning methods. There are different types of text classification. Authorship attribution, spam filtering, dialect identification are just several of the purposes of text categorization. It is natural that for different types of categorization different methods are pertinent. The most common type is the content-based categorization which classifies texts by their topic and requires the most common classification methods based on classical set of features. More specific methods are necessary in cases when classification criterions are not so obvious, for example, in the case of author identification.

In this paper the application of the PPM (Prediction by Partial Matching) model for automatic text classification is explored. Prediction by partial matching (PPM) is an adaptive finite-context method for text compression that is a back-off smoothing technique for finite-order Markov models [1]. It obtains all information from the original data, without feature engineering, is easy to implement and relatively fast. PPM produces a language model and can be used in a probabilistic text classifier.

PPM is based on conditional probabilities of the upcoming symbol given several previous symbols [2]. The PPM technique uses character context models to build an overall probability distribution for predicting upcoming characters in the text. A blending strategy for combining context predictions is to assign a weight to each context model, and then calculate the weighted sum of the probabilities:

$$P(x) = \sum_{i=1}^{m} \lambda_i p_i(x), \quad (1)$$

where

λ_i and p_i are weights and probabilities assigned to each order *i (i=1…m)*.

For example, the probability of character '***m***' in context of the word '***algorithm***' is calculated as a sum of conditional probabilities dependent on different context lengths up to the limited maximal length:

P_{PPM}('***m***') = $\lambda_5 \cdot$ P('***m***' | '***orith***') + $\lambda_4 \cdot$ P('***m***' | '***rith***') + $\lambda_3 \cdot$ P('***m***' | '***ith***') +
+ $\lambda_2 \cdot$ P('***m***' | '***th***') + $\lambda_1 \cdot$ P('***m***' | '***h***') + + $\lambda_0 \cdot$ P('***m***') + $\lambda_{-1} \cdot$ P('esc'), (2)

where

λ_i (i = 1…5) is the normalization weight;

5 is the maximal length of the context;

P('esc') – 'escape' probability, the probability of an unknown character.

PPM is a special case of the general blending strategy. The *PPM* models use an escape mechanism to combine the predictions of all character contexts of length *m*, where *m* is the maximum model order; the order 0 model predicts symbols based on their unconditioned probabilities, the default order -1 model ensures that a finite probability (however small) is assigned to all possible symbols. The *PPM* escape mechanism is more practical to implement than weighted blending. There are several versions of the *PPM* algorithm depending on the way the escape probability is estimated. In our implementation, we used the escape method C [3], named PPMC. Treating a text as a string of characters, a character-based *PPM* avoids defining word boundaries; it deals with different types of documents in a uniform way. It can work with texts in any language and be applied to diverse types of classification; more details can be found in [4]. Our utility function for text classification was cross-entropy of the test document:

$$H_d{}^m = -\sum_{i=1}^{n} p^m(x_i) \log p^m(x_i), \quad (3)$$

where

n is the number of symbols in a text d,

$H_d{}^m$ – entropy of the text d obtained by model m,

$p^m(x_i)$ is a probability of a symbol x_i in the text *d*.

$H_d{}^m$ was estimated by the modelling part of the compression algorithm.

Usually, the cross-entropy is greater than the entropy, because the probabilities of symbols in diverse texts are different. The cross-entropy can be used as a measure for document similarity; the lower cross-entropy for two texts is, the more similar they are. Hence, if several statistical models had been created using documents that belong to different classes and cross-entropies are calculated for an unknown text on the basis of each model, the lowest value of cross-entropy will indicate the class of the unknown text. In this way cross-entropy is used for text classification.

On the training step, we created *PPM* models for each class of documents; on the testing step, we evaluated cross-entropy of previously unseen texts using models for each class. Thus, cross-entropy was used as similarity metrics, the lowest value of cross-entropy indicated the class of the unknown text.

The maximal length of a context equal to 5 in PPM model was proven to be optimal for text compression [6]. In all our experiments with character-based PPM model we used maximal length of a context equal to 5; thus our method is PPMC5.

The character-based *PPM* models were used for spam detection, source-based text classification and classification of multi-modal data streams that included texts. In [1], the character-based PPM models were used for spam detection. In [5], the PPM algorithm was applied to text categorization in two ways: on the basis of characters and on the basis of words.

3 Method description

In the previous author identification experiments the standard PPM classification methodology was applied to the large set of forum posts written by tens of forum participants. More exactly, we experimented with 30 authors, 100 posts for each authors; approximately length of posts was 150-200 words. The unknown unit was one post. The task was to classify test posts having these 30 classes – authors. The classification accuracy was surprisingly high: F-measure was around 0.8. Further experiments showed that the test text length is the most influencing factor and the maximal accuracy we reached with test texts of 300 words length. The F-measure was 0.97 and did not grow for the longer test texts.

The current task had several differences and we could not use the standard classification methodology directly. First, the task was to make a decision about only one test text whether it belonged to the same author as the several known texts which were written by one author. Thus, we actually had only one class and could not compare entropies of the test text for several classes in order to select one. Second, in this task training and test data were extremely small; in some cases test text was larger than the whole training set. It should be mentioned that the volume of test and especially training data affected the classification results for our method; it tended to attribute texts to the class with the larger training set. We applied a special normalization procedure to normalize entropies of larger and smaller training texts for better classification in the experiments with forum posts.

For the data in this experiment we needed a number of known and unknown texts; thus we divided all given known and unknown texts in small fragments and compared their entropies calculated on the basis of known texts models and on the basis of

unknown text models. If the entropies were considerably different we considered that texts were written by two different authors. The following algorithm presents our methodology in more details.

The algorithm:

- join all known texts into one;
- divide this known text in fragments with the similar length (about 350 words);
- divide unknown text in fragments with the similar length (about 350 words);
- for each known fragment in turn:
    - create PPM statistical model on the rest of known texts;
    - calculate entropy of this fragment on the basis of the created model.
- for each unknown fragment in turn:
    - create PPM statistical model on the rest of unknown fragments;
    - calculate entropy of this fragment on the basis of the created model.
- create PPM statistical model on all unknown texts;
- for each known fragment in turn:
    - calculate entropy of this fragment on the basis of the created model.
- create PPM statistical model on all known texts;
- for each unknown fragment in turn:
    - calculate entropy of this fragment on the basis of the created model.
- compare these four lists of the entropies as the statistical variables using t-test for the null hypothesis that these variables are equivalent. If the null hypothesis was accepted we considered that the unknown text was written by the same author as known texts, otherwise we decided that the known and unknown texts were written by two different authors.

References

1. Bratko, A., Filipic, B.: Spam Filtering Using Compression Models. Department of Intelligent Systems, Jozef Stefan Institute, Ljubljana, Slovenia, IJS-DP-9227 (2005)
2. Cleary, J., Witten, I.: Data Compression Using Adaptive Coding and Partial String Matching. IEEE Transactions on Communications, vol. 32, no. 4, pp. 396 – 402 (1984)
3. Bell, T.C., Witten, I.H., Cleary, J. G.: Modeling for text compression. Computing Surveys, 21(4), pp. 557-591 (1989)
4. Bobicev, V.: Comparison of Word-based and Letter-based Text Classification. Recent Advances in Natural Language Processing V, Bulgaria, pp. 76–80 (2007)
5. Bobicev, V., Sokolova, M.: An effective and robust method for short text classification. Proceedings of the 23rd national conference on Artificial intelligence - Volume 3, pp. 1444–1445 (2008)
6. Teahan, W.: Modelling English text. PhD Thesis, University of Waikato, New Zealand (1998)