Monolingual Written Natural Language Identification with Naïve Bayes and LSTMs

Jacob Lundberg
Linköping University
LiU-id: jaclu010
Email: jaclu010@student.liu.se

***Abstract*—This project aims to correctly classify the language of written text paragraphs. Given a paragraph, one out of 235 languages can be classified. Chosen model solutions to the classification problem is an Naïve Bayes model and an LSTM. They reach a classification accuracy of 94% and 78% respectively. This is a lot better that the authors proposed method, achieving an accuracy of 88%.**

I. Introduction

The art of identifying the language in a text jas been around a long time. It is something we humans do on a daily basis. The agent in question needs to understand what language is being read in order to interpret it correctly. It is an important part of the pipeline of understanding text. In theory, any classifier should be susceptible for evaluation and many has been tested (references), but there are a number of specific classifiers that has gained traction within this field. (references)

For this project two methods are being evaluated on the Wi-Li dataset [1]. The methods chosen are a Naïve Bayes unigram classifier and an LSTM neural network. The dataset in question contains phrases from Wikipedia on 235 defferent languages with a thousand (1000) phrases from each language.

As described by Thoma [1], a Multi Layer Perceteron with character based tf-idf features is implemented and achieves an overall accuracy of 88%. Also, a number of existing classifiers are tested against said dataset but only reach a prediction accuracy of 22-36%. This is mainly due to that these classifiers only are implemented for a subset of languages existing in this dataset. The lack of existing solutions to both suggested classifiers by Thoma and other state-of-the-art solutions (such as LSTM) makes an interesting project.

The idea of this project is to use a Naïve Bayes model as a baseline and then see if improvement can be made with an advanced model such as an LSTM.

The proposed advanced model consits of sequences of character n-grams, thus an LSTM is seen as apropriate.

II. Theory

This chapter will briefly describe the necessary theory, i.e. the theory behind the models used.

A. tf-idf

tf-idf is a common feature selection technique in Natural Language Processing tasks. It is a short for *term frequency - inverse document frequency* and it is a model to evaluate how relevant a word is given a set of documents.

Given a collection of N documents, the frequency f of word i in document j helps define the *term frequency* tf as

$$tf_{i,j} = \frac{f_{i,j}}{\max_k f_{k,j}},$$

where $\max_k f_{k,j}$ denotes the word in the document with the highest frequency. This means that word j is normalized on the maximum occurance of any word in the document.

Then, suppose word i occures in n_i out of the N documents. Then, the *inverse document frequency* is defined as

$$idf_i = \log_2(\frac{N}{n_i}).$$

Finally, the two numbers are multiplied,

$$tf\text{-}idf = tf_{i,j} \times idf_i$$

and the word with the highest score usually carry most information about a document. [2]

B. Multinomial Naïve Bayes

A Naïve Bayes model is a derivation of a graph structure organized in such a way that there is assumed independence between all covariates.

Given a class C_k out of k possible classes and a word $x_i \in \boldsymbol{X}$, where $\boldsymbol{X}$ is the corpus, the probability of a certain class is given by

$$p(C_k, \boldsymbol{X}) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

This is derived from Baye's theorem and the independece assumption between all $x_i \in \boldsymbol{X}$. [3]

Given this formula, in the classification problem, one would have to compute the probability for each class and pick the class with the highest probability as a classification [4].

C. fastText

Neural networks cannot on their own understand text in string format. The strings need to be transformed into numbers or vectors. *fastText* is one such model and it is a part of a theory called *word embeddings*.

fastText produces embedding vectors by looking at each n-gram of a word with a size of 3-6 characters. Then, a vector representation is assigned to each n-gram. Finally, the sum of the n-gram vectors represent the final embedding for the word. [5]

A advantage of fastText compared with Google's *word2vec* [6] is that fastText is able to produce embeddings on words not yet seen in the data while training. This is limted to that at least some n-gram of a word were present during training. [5]

D. LSTM

III. Data

This chapter references Thoma [1]. As said, the dataset consists of 235 languages with text extracted from Wikipedia. Each language consists of 1000 paragraphs of text, making it a total of 235 000 paragraphs. The paragraphs are generated from random Wikipedia pages that consists of at least 140 characters in Unicode format. The dataset is already split into train and test data with 117 500 paragraphs each. Each of these are combined with labels of the language in pair with each paragraph. The data set is available online[1].

A. Balancing

Even though each language contains a balanced amount of paragraphs, it does not mean that the paragraphs themselves are balanced. As an example, the shortest paragraph is 140 Unicode points long while the longest is 195 402 Unicode points long. The average length is 371 Unicode points long.

B. Issues in the dataset

Thoma presents 5 distinct error sources within the data. The most disturbing are presented below.

1) References: Sometimes, books or other autors are mentioned in the article in question and is often written in the original language which often is english. A concrete example is the second paragraph from the training set:

Sebes, Joseph; Pereira Thomas (1961) (på eng). The Jesuits and the Sino-Russian treaty of Nerchinsk (1689): the diary of Thomas Pereira. Bibliotheca Instituti historici S. I., 99-0105377-3 ; 18. Rome. Libris 677492

The paragraph is from the Swedish Wikipedia, because of the string *(på eng)*, which makes this paragraph hard to classify correct.

2) Translation: Some Wikipedia pages have english translations. A notable example is from the Tonganese Wikipedia [2].

3) Quotes: Many articles contain quotes, which are often written in the original language. This error is closely related to Section III-B1.

[1]https://doi.org/10.5281/zenodo.841984

[2]https://to.wikipedia.org/wiki/Sione_Tupou_Mateialona/en

IV. Method

This chapter describes the steps taken to produce results. All implementation is done in Jupter notebooks[3], with Python 3[4] as programming language. The resulting implementation is available online[5].

A. Data collection

The dataset consists of four *.txt*-files. This is the train an test data along with their label. On top of this a sheet with more information regarding language shortenings was shipped with the data.

B. Pre-processing

The data consists of long paragraphs of text. As the data is not pre-processed in anyway before, there still exists punctuation marks, spaces and other delimiters that may have an impact on the final performace. Therefore a regular expression is applied to each paragraph with the intent to remove all punctuation marks and to some extent correctly split words.

This was harder than expected since Chinese for example uses an other Unicode character for a comma i comparison to english. This and other examples based on primarily [7], [8], [9] are added to the regular expression.

1) LSTM pre-processing: This pre-processing was enough for the Naïve Bayes model. For the LSTM model, a fastText word embedding was trained on the data. The fastText model used a vector size of 50, a window of 5 words, a minimum word frequency of 20 and, a 10 iteration training procedure. This was created with the *Gensim*[6] implementation in Python.

C. Multinomial Naïve Bayes

This model was implemented with *sk-learn*'s[7] implementation of the Naïve Bayes model. First, a *count vectorizer* is applied on the training data, which simply counts occurances of each word creating a 2D matrix with documents and counts.

Two models was created. One which only uses the previously mentioned count matrix (further referenced as *NB-cv*) and the second which converts the count matrix to *tf-idf* scores before training (further referenced as *NB-tf-idf*).

The test set is the transformed into a count matrix which is used by bothe models to make predictions.

D. LSTM

E. Evaluation

V. Results

This section presents the results obtained. They are shown as percentages in I. A detailed confusion matrix for the NB-cv model can be found in Appendix A, figure 1.

[3]https://jupyter.org/

[4]https://python.org/

[5]https://github.com/jaclu010/text-mining

[6]https://radimrehurek.com/gensim/models/fasttext.html

[7]https://scikit-learn.org/

TABLE I
RESULTS FOR THE CLASSIFIERS, NUMBERS IN PERCENT %

%	NB-cv	NB-tf-idf		
Accuracy	93.87	92.99		
Macro precision	**95.89**	**95.25**		
Micro precision	93.87	92.99		
Recall	93.87	92.99		
F1-score	94.39	93.52		

VI. DISCUSSION

* A lot of english words in some paragraphs, a lot of paragraphs should be classified as english

* Size of ngram, multiple sorts of ngrams

* Not enough data for the word vector

* Chinese and other sign based written languages may not contain relevancy between adjacent character the same way as western languages use.

* Downside of tf-idf, Very common words are useful to identify a certain language, opposite of what the model is doing.

VII. CONCLUSION

A simple model as Naïve Bayes works extremely well, given its simple characteristics. LSTM does not procure the necessary results to motivate further devlopment.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] M. Thoma, "The wili benchmark dataset for written language identification," 2018.

[2] A. Rajaraman and J. D. Ullman, *Data Mining*. Cambridge University Press, 2011, p. 1–17.

[3] S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010. [Online]. Available: http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0136042597,00.html

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[7] Wikipedia. Chinese punctuation. Read: 2018-12-29. [Online]. Available: https://en.wikipedia.org/wiki/Chinese_punctuation

[8] ——. Lao language. Read: 2018-12-29. [Online]. Available: https://en.wikipedia.org/wiki/Lao_language

[9] ——. Tibetan alphabet. Read: 2018-12-29. [Online]. Available: https://en.wikipedia.org/wiki/Tibetan_alphabet

APPENDIX

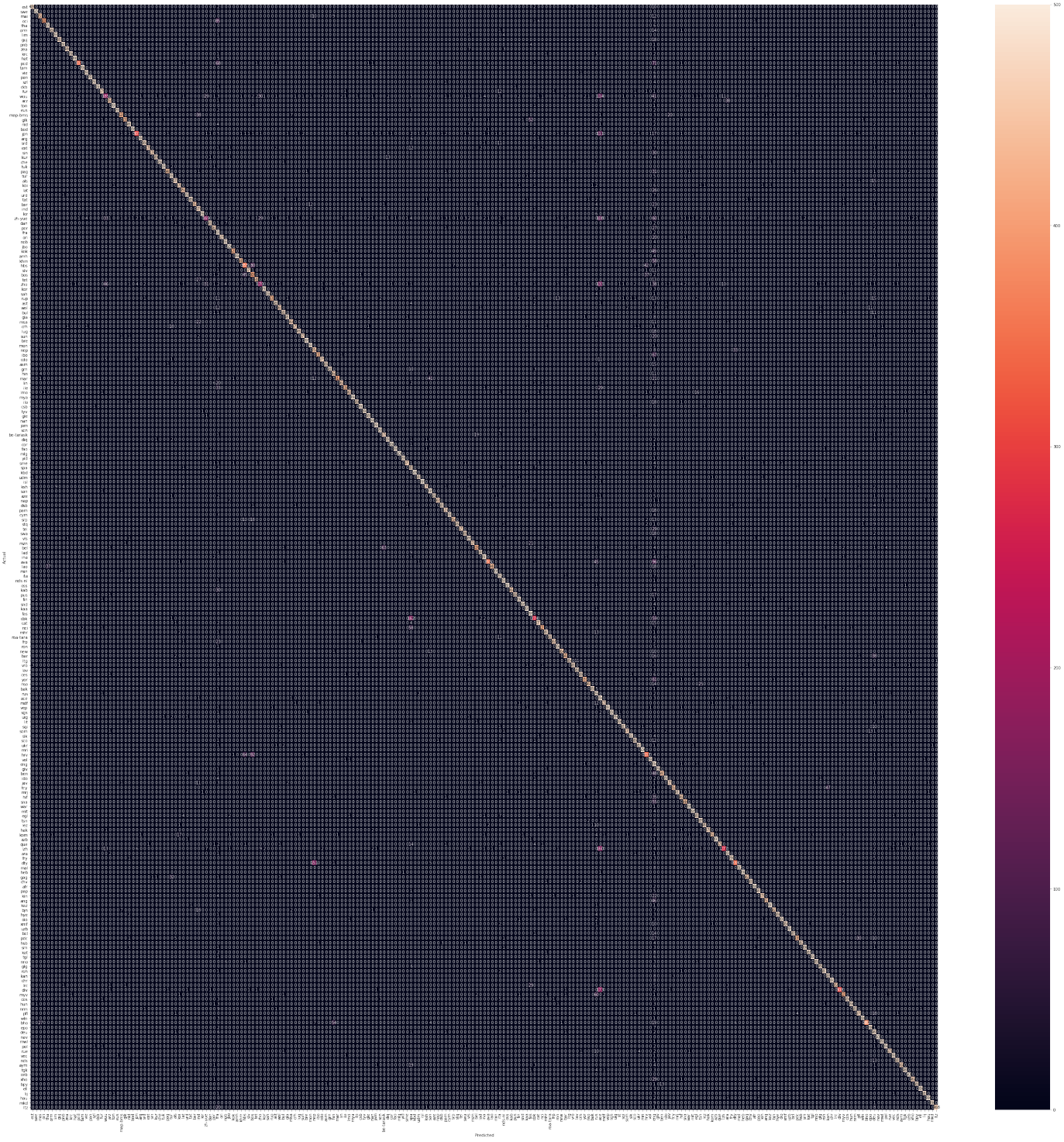

Fig. 1. Confusion matrix for the NB-cv model