Optimizing Naive Bayes for Arabic Dialect Identification

Jauhiainen, Tommi

2022

Jauhiainen , T , Jauhiainen , H & Linden , K 2022 , ' Optimizing Naive Bayes for Arabic Dialect Identification ' , Arabic Natural Language Processing Workshop , Abu Dhabi , 08/12/2022 - 08/12/2022 .

http://hdl.handle.net/10138/352786

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI
HUMANISTINEN TIEDEKUNTA
HUMANISTISKA FAKULTETEN
FACULTY OF ARTS

OPTIMIZING NAIVE BAYES FOR ARABIC DIALECT IDENTIFICATION

Tommi Jauhiainen
Department of Digital Humanities, UH

Heidi Jauhiainen
Department of Digital Humanities, UH

Krister Lindén
Department of Digital Humanities, UH

FIN-CLARIN

INTRODUCTION

This poster describes the language identification system used by the SUKI team in the 2022 Nuanced Arabic Dialect Identification (NADI) shared task. The third NADI shared task featured 18 country-level dialects of Arabic. We used a Naive Bayes-based language identifier with character n-grams. We implemented a new version, which automatically optimizes its parameters. With the macro F1 score of 0.1963 on test set A and 0.1058 on test set B, we achieved the 18th position out of the 19 competing teams.

SYSTEM

The system uses a Naive Bayes-based method using the observed relative frequencies of multiple-size character n-grams as probabilities. It adds together logarithms of the relative frequencies of character n-gram combinations *f* in the training data *C* for language *g* as defined in the Equations below.

$$R(g,M) = -lg_{10}\prod_{i=1}^{\ell_{M^F}} v_{C_g}(f_i) = \sum_{i=1}^{\ell_{M^F}} -lg_{10}(v_{C_g}(f_i)) \tag{1}$$

$$v_{C_g}(f) = \begin{cases} \frac{c(C_g,f)}{\ell_{C_g^F}}, \textit{ if } c(C_g,f) > 0 \\ \frac{1}{\ell_{C_g^F}} pm, \textit{ otherwise} \end{cases} \tag{2}$$

The exact range of the used character n-grams is optimized using the development data.

AUTOMATIC OPTIMIZER

On this occasion, we implemented an automatic optimizer to streamline experimentation. The automatic optimizer is first given initial character n-gram and penalty modifier ranges which it then uses to populate a todo-table as in the example below.

n-gram range	penalty modifier
1 – 4	1.3
2 – 4	1.3
1 – 5	1.5
1 – 5	1.8

Table 1: original todo-table

The parameters in the todo-table are evaluated, and the results are stored in a master results list. An additional top ten list of macro F1 scores is created with the parameters used to obtain them. The parameter instances used in the top ten list are checked, and nearby parameter combinations are added to a new todo-table if they are not found in the master results list. In the case of n-gram ranges, the optimizer tries one higher and one lower for both the minimum and maximum n-gram sizes. For the penalty modifier, it adds and subtracts 0.5 from the current one if there are no other penalty modifiers for the respective n-gram range in the master results list. Suppose a "neighboring" penalty modifier exists in the results list. In that case, the halfway between the penalty modifiers is tried if the distance between modifiers is larger than 0.1. Table 2 is an example of a new todo-table generated from the one in Table 1.

n-gram range	penalty modifier
1 – 3	1.3
1 – 5	1.3
1 – 4	1.8
1 – 4	0.8
2 – 5	1.3
3 – 4	1.3
2 – 4	0.8
2 – 4	1.8
1 – 6	1.5
1 – 4	1.5
2 – 5	1.5
1 – 5	1.0
1 – 5	1.65
1 – 6	1.8
2 – 5	1.8
1 – 5	2.3

Table 2: new todo-table

The cycle of evaluating the todo-table, making a top ten list, and creating a new todo-table is continued as long as the top ten list changes between cycles. We have published the code of the version used in the NADI shared task on GitHub at https://github.com/tosaja/TunPRF-NADI.

EXPERIMENTS

Additionally, we were trying to develop a way to use unlabeled data to improve the identifier results. Our experiments to utilize unlabeled data did not improve the identification results on the development set. In the end, we did not use unlabeled data in the one run we submitted. Also, as the language identification accuracy was already relatively low, language model adaptation did not prove advantageous with the development data. Thus we submitted our only run using the non-adaptive NB identifier.

SUBMISSIONS

We submitted only one run on each test set using the non-adaptive version of the language identifier. First, we treated the training and the test data with the Farasa segmenter. We then ran them through the Naive Bayes language identifier using character n-grams from one to four with a penalty modifier of 1.375. With the macro F1 score of 0.1963 on test set A and 0.1058 on test set B, our submissions reached the 19/19 and 15/19 positions for the respective test sets. The final ranking for the whole shared task combined the results of the two test sets. We were ranked 18th out of the 19 participating teams, which shows that our results could have been more competitive against most other submitted results.

CONCLUSION

We successfully implemented a new version of the NB identifier, which automatically optimizes its parameters, thus leaving more time to explore ideas to improve the identification accuracy. We reached the 19th and 15th places in the shared task.

ACKNOWLEDGMENTS

The research presented in this article has been partly supported by The Finnish Research Impact Foundation Tandem Industry Academia funding, the Academy of Finland, and the University of Helsinki in cooperation with Lingsoft.

CONTACT INFORMATION

For each of the authors:
firstname.lastname@helsinki.fi