Proceedings of the Workshop on Processing Language Variation: Digital Armenian (DigitAm 2022) @LREC2022, pages 8–12
Marseille, 20 June 2022

Dialects Identification of Armenian Language

Karen Avetisyan
Russian-Armenian University
Yerevan, Armenia
avetisyan.karen@student.rau.am

Abstract

The Armenian language has many dialects that differ from each other syntactically, morphologically, and phonetically. In this work, we implement and evaluate models that determine the dialect of a given passage of text. The proposed models are evaluated for the three major variations of the Armenian language: Eastern, Western, and Classical. Previously, there were no instruments of dialect identification in the Armenian language. The paper presents three approaches: a statistical which relies on a stop words dictionary, a modified statistical one with a dictionary of most frequently encountered words, and the third one that is based on Facebook's fastText language identification neural network model. Two types of neural network models were trained, one with the usage of pre-trained word embeddings and the other without. Approaches were tested on sentence-level and document-level data. The results show that the neural network-based method works sufficiently better than the statistical ones, achieving almost 98% accuracy at the sentence level and nearly 100% at the document level.

Keywords: Dialect identification, Western Armenian, Eastern Armenian, Classical Armenian

1. Introduction

The Armenian language has many actively used dialects. They differ from each other syntactically, morphologically, and phonetically. Considering the variation in the existing literature and the current usage of various variants of the language, dialect identification in texts is a relevant and open problem for Armenian dialects. Thus, this paper tries to solve that problem for three major variations of the Armenian language: Eastern, Western, and Classical.

Dialect identification is similar to language identification, but has several important differences that make the task more challenging. Contrary to different languages, dialects share the same script and have highly overlapping vocabularies. Despite the subtle differences between the tasks, to solve the dialect identification task in this work we study the performance of established lexicon- and artificial neural network-based language identification approaches.

Lexicon-based methods use a list of stop words for each language to detect their texts. A similar approach was shown in Truică et al. (2015), where the stop words and diacritics formed the lexicon. The Armenian language has no diacritics and stop words can be very similar among dialects, therefore this method may not always be suitable for the chosen task. For that reason, it was modified to use the list of the most frequent words of each of the considered dialects.

The artificial neural network-based approach learns numerical representations of words and uses them as input features for classification. One of the most popular implementations of this method relies on Facebook's fastText library[1] for text classification and representation (Joulin et al., 2016) as it has shown high results on language identification tasks. Here, the model was trained both with the usage of the pre-trained fastText word embeddings and without it. To train the model, data from Western[2] and Eastern[3] Armenian Wikipedia was collected, as well as the data from Digilib[4] for Classical Armenian.

All three methods were tested on sentence-level and document-level testing datasets that were also collected from Wikipedia-s and Digilib texts. In addition to this, the dependency of text segment size to dialect identification accuracy is shown.

2. Methods

To solve the task of Armenian dialect identification, three methods were used.

(i) Stop Words: As a baseline solution for the problem, a stop-word-based algorithm was selected. Let W_d be the stop words vocabulary of the dialect $d (d \in \{Western, Eastern, Classical\})$. W_e is the set of words contained in the text E. For each text E, the dialect is predicted according to this statement:

$$label(E) = argmax_d\ (|W_d \cap W_e|)$$

If there are two or three maximal values, the label is chosen randomly according to the values.

(ii) Lexicon-Based: The first method was modified by making the W_d not only stop words vocabulary. Here, 2 different W_d vocabularies were tested. The process of both W_d dictionary-formation is described in Chapter 3.

(iii) Neural Network-Based: For the other method, Facebook's fastText language identification model was utilized. Here, the words are being presented as a set of n-grams. Each n-gram has its representation vector that is also trainable. These representations are then being averaged and given to a linear classifier. In the end, *softmax* is used as an activation function.

The model was trained to predict 3 dialects: Eastern Armenian, Western Armenian, and Classical Armenian.

[1] https://fasttext.cc/blog/2017/10/02/blog-post.html
[2] https://hyw.wikipedia.org/
[3] https://hy.wikipedia.org/
[4] https://digilib.aua.am/en

3. Data

To collect the training data for Eastern (hye) and Western (hyw) Armenian, respective Wikipedia dumps[5][6] were used. Whereas, the texts from Digilib were utilized to get the data for Classical Armenian.

The stop word dictionary for Western and Classical Armenian was collected manually utilizing the list of most frequent words that was in turn collected from the above-described resources. Eastern Armenian stop words were taken from here[7].

For the lexicon-based method, three dictionaries (one for each dialect) were formed. The formation was processed, in two different ways, using the corresponding data for each of the dialects separately. Removing the words that contain non-Armenian letters as well as punctuation symbols, the word frequency was counted. Assuming that $\boldsymbol{V_{A,k}}$ stands for the set of top k most frequent words in dialect A, the final dictionary for the dialect A, in two different ways **a)** and **b)**, will look as follows:

$$\text{a)} \quad D_A = V_{A,k} \setminus (V_{B,k} \cap V_{C,k}),$$
$$\text{b)} \quad D_A = V_{A,k} \setminus (V_{B,k} \cup V_{C,k}),$$

where $\boldsymbol{V_{B,k}}$ and $\boldsymbol{V_{C,k}}$ are the sets of top k frequent words in dialects B and C, respectively.

As for the data to train the fastText language identification model, sentences were randomly extracted from the considered datasets. It was decided not to filter the extracted sentences according to their length, taking into account the fact that fastText trains its own models using sentences with different lengths. For each of the dialects, the training set contains an equal number of sentences.

To test the methods, two types of test data were created. The first one is a set of sentences randomly extracted from the Wikipedia dumps and Digilib. For each dialect, this set contains 500 sentences. The average length of the sentences is equal to nearly 18 words or ≈ 130 characters.

The second test set consists of whole texts, a hundred documents for each dialect, randomly extracted from the same sources. The average length of the document is equal to ≈ 600 words or ≈ 4150 characters. For Classical Armenian, only the first 50 sentences of each document were extracted to balance the average length of documents for each of the dialects.

4. Experiments

In this chapter, the process of hyperparameter tuning, the results on tuned hyperparameters, and some other additional statistics are shown.

The best results that each of the described methods achieve, and their corresponding time consumption, are shown in Table 1 and Table 2 separately for sentence and document level test sets.

According to the results shown in Table 1 and Table 2, the neural network-based method achieves sufficiently better results than the ones based on vocabulary.

Further, in subchapters 4.1 and 4.2 more detailed results for all the conducted experiments are described.

Methods	Accuracy	Time
Stop-Words	0.51	0.02s
Lexicon-Based	0.67	1.71s
Neural-Network	0.98	0.14s

Table 1: The best results and time consumption of each method on the **sentence-level test set**. (Processing time of 1500 sentence examples)

Methods	Accuracy	Time
Stop-Words	0.55	0.04s
Lexicon-Based	0.67	0.16s
Neural Network	1.00	0.76s

Table 2: The best results and time consumption of each method on the **document-level test set**. (Processing time of 300 document examples)

4.1 Lexicon-based method

For the lexicon-based method, we tuned k, the number of the most frequent words used to create the final dictionaries. For each value of k, and for both variations of dictionary-creation, the accuracy score was calculated. The results of these experiments for sentence-level and document-level test sets are shown in Figure 1 and Figure 2.

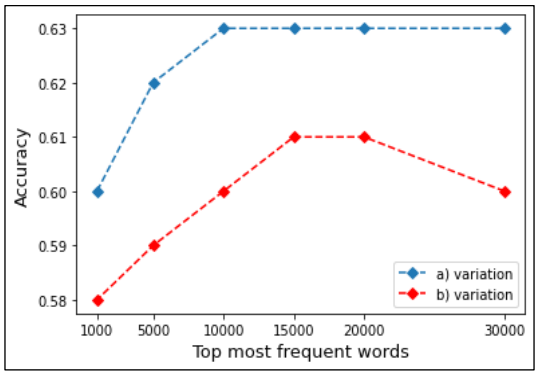

Figure 1: The comparison of a) and b) dictionary versions in terms of accuracy score shown on the **sentence-level** test set depending on the number of most frequent words taken.

According to the results (Figure 1 and Figure 2), it is noticeable that the **b)** version of dictionary-creation overall works better on both of the test sets.

4.2 Neural network-based method

For this method, we trained the fastText model on 3 different size datasets. These datasets consisted of 1000, 2000, and 5000 sentences per each label.

[5] https://dumps.wikimedia.org/hywwiki/

[6] https://dumps.wikimedia.org/hywiki/

[7] https://github.com/stopwords-iso/stopwords-hy

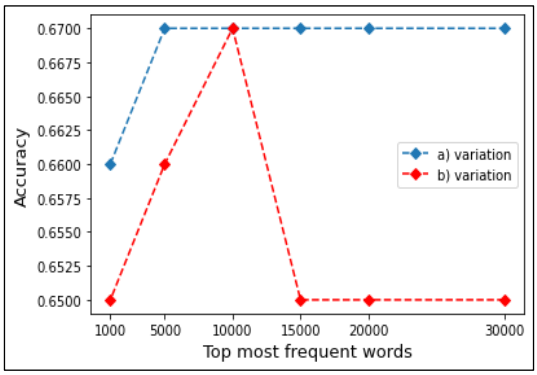

Figure 2: The comparison of a) and b) dictionary versions in terms of accuracy score shown on the **document-level** test set depending on the number of most frequent words taken.

4.2.1 Hyperparameters

For more efficient usage of the method, we had to tune some basic hyperparameters like *minn* and *maxn,* which denote the minimal and maximal length of character n-grams. Taking into account the fact that the average length of the words used in the training set is nearly 6 characters, the minimal and maximal lengths of character n-grams were tuned within these limits.
In addition, the process of training was held with and without pre-trained word vectors. While using the pre-trained word vectors, the *dim* parameter, which stands for the size of word vectors, was equal to 300. As pre-trained word vectors, fastText's default vectors for the Armenian language were used. When the training process was held without pre-trained vectors, the *dim* parameter was set to 16 as it is suggested in the fastTexts language identification tutorial[8].
Hyperparameter tuning was performed on the sentence-level test set. The results both with and without the usage of pre-trained word vectors are shown in Table 3 and Table 4. The presented results are the average of 5 separate runs with different random seeds.
As we can see from Table 3 and Table 4, the results are much more stable with the usage of pre-trained vectors, while the n-gram minimum and maximum sizes change. The best results were also achieved with the usage of pre-trained vectors with the training set size of 5000 sentences per label.

4.2.2 Results

Based on the hyperparameter tuning results, the models that achieved the best results were taken. For these models, their confusion matrixes are presented in Figure 3. As we can see from these matrixes, the models are mainly confused in predicting Classical Armenian sentences as Western Armenian ones, and Western Armenian sentences as Eastern Armenian ones.

Sentences per label	1000						2000						5000					
maxn / minn	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	93,3	95,7	96	**96,1**	96	95,9	94,1	96,2	96,9	**97,1**	97	97	95,2	96,7	96,7	96,9	97,2	97,1
2		95,3	95,4	95,7	95,7	95,8		96,3	97	97	97	97		96,3	96,5	97	97,1	97,3
3			95,5	95,5	95,5	95,5			96,7	96,9	96,9	96,9			97	97,5	**97,6**	97,5
4				95,2	94,9	94,7				96,9	96,9	96,9				96,9	97,2	97,3
5					94,4	94,2					96,7	96,7					96,7	96,9
6						94,1						96,6						96,6

Table 3: *minn* and *maxn* hyperparameters tuning, on sentence-level test set, for different size training data **with** pre-trained word vectors (dim=300).

Sentences per label	1000						2000						5000					
maxn / minn	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	84,4	74,9	71,4	76,2	81,6	68,9	90,1	94,2	93,4	91,7	89,7	83,2	92,7	96	96,1	96,1	96	95,9
2		**88,7**	68,6	73,4	79,7	77,8		**94,8**	94,1	92,9	91,1	84,5		96,1	96,7	96,5	96,4	96,3
3			85,2	74	78,7	78,9			94,6	93,6	90,9	81,2			**97**	96,9	96,4	96,1
4				79,1	76,8	72,9				94,3	92,2	85,5				96,5	96,4	96
5					71,7	61,3					92,8	89					96,2	96,1
6						68,1						91,9						95,9

Table 4: *minn* and *maxn* hyperparameters tuning, on sentence-level test set, for different size training data **without** pre-trained word vectors (dim=16).

[8] https://fasttext.cc/blog/2017/10/02/blog-post.html

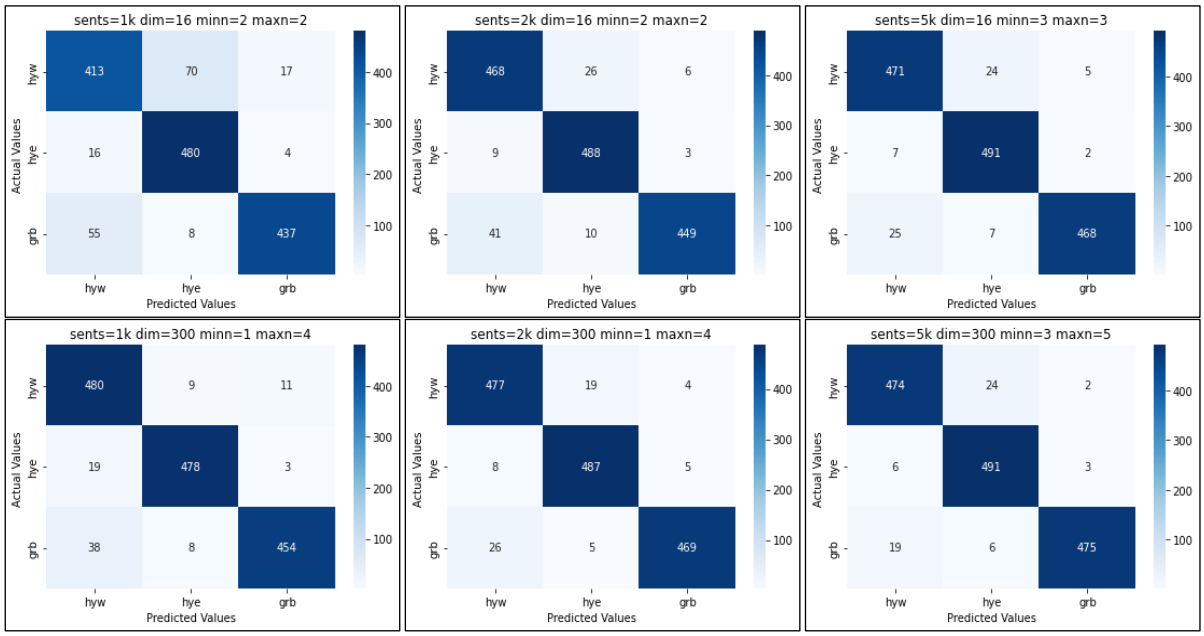

Figure 3: Confusion matrixes of models with best hyperparameters on the sentence-level test set.

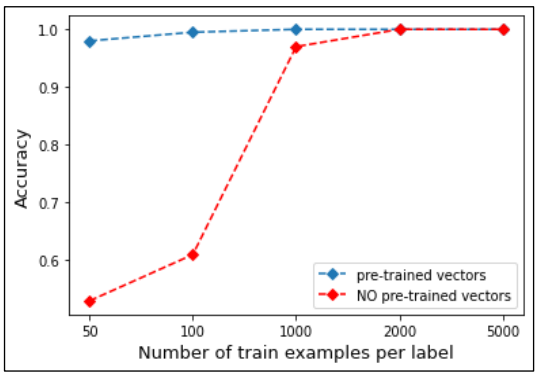

Figure 4: A comparison of models that were trained with and without pre-trained vectors on the **document-level test set**, while changing the number of training examples.

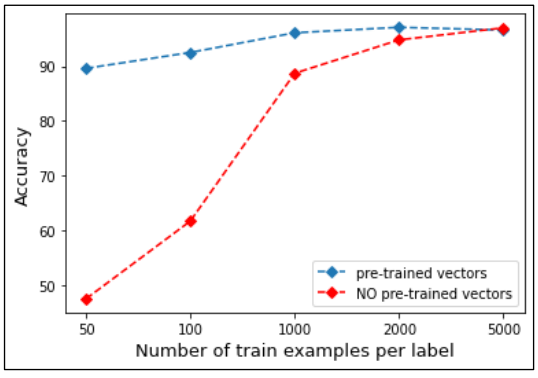

Figure 5: A comparison of models that were trained with and without pre-trained vectors on the **sentence-level test set**, while changing the number of training examples.

The best models for each number of train examples and according to the usage of pre-trained vectors were also tested on the sentence and document level test sets. These results are shown in Figure 4 and Figure 5**.** Here the presented results are also the average of 5 seeds.

From Figure 4 and Figure 5 we can conclude that the models for which pre-trained vectors were used achieve the same results with a smaller amount of data used for their training. Also, we can see that the model that does not use pre-trained vectors and for the training of which 5000 sentences per label were used, achieves nearly the same results as the model that uses the vectors.

Further, to minimize the time consumption on the document-level dialect identification task, additional experiments were held using only the first n symbols of each test example. The value of n was changed from 10 to 200 symbols. Time consumption for an experiment, where n was equal to 200 symbols, was decreased by nearly 20 times for each of the considered models. The achieved accuracy scores for these experiments are shown in Figure 6. The final scores were also calculated by averaging the results of 5 seeds.

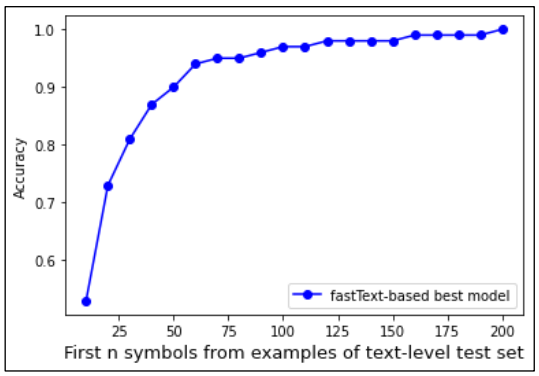

Figure 6: Accuracy score according to the change of a number of first symbols that are given to the neural network-based best model.

5. Conclusion

In this work, we evaluated three different methods of Armenian dialect identification. The neural network-based method performed best, achieving 98% accuracy at sentence level and 100% accuracy at document level. Utilizing pre-trained word vectors to train the neural network allowed us to achieve decent results for this task, using only a small number of training examples. This feature could be helpful for the identification process of less popular dialects. Additionally, it was shown that using only the first 200 characters of the document would be sufficient for accurate dialect identification, which in practice will help to significantly reduce the computation time when processing long documents.

6. Bibliographical References

Ali, A., Dehak, N., Cardinal, P., Khurana, S., Yella, S., Glass, J., Bell, P. and Renals, S. (2016). Automatic Dialect Detection in Arabic Broadcast Speech. *Interspeech 2016.*

Balaji, N.N.A. and Bharathi B. (2020). Semi-supervised Fine-grained Approach for Arabic dialect detection task. *In Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 257–261, Barcelona, Spain.

Belinkov, Y. & Glass, J. (2016). A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects. *VarDial@COLING.*

Biadsy, F., Hirschberg, J. and Habash, N. (2009). Spoken Arabic Dialect Identification Using Phonotactic Modeling. *Proceedings of EACL 2009 Workshop on Computational Approaches to Semitic Languages.*

Darwish, K., Sajjad, H., & Mubarak, H. (2014). Verifiably effective arabic dialect identification. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* pages 1465-1468.

Franco-Penya, H.-H. and Sanchez, L.M. (2016). Tuning Bayes Baseline for Dialect Detection. *In Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3),* pages 227–234, Osaka, Japan.

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.*

Malmasi, S., Refaee, E. and Dras, M. (2015). Arabic Dialect Identification Using a Parallel Multidialectal Corpus. *PACLING 2015.*

Talafha, B., Ali, M., Za'ter, M. E., Seelawi, H., Tuffaha, I., Samir, M., Farhan W. & Al-Natsheh, H. T. (2020). Multi-dialect arabic bert for country-level dialect identification.

Truică, C.-O., Velcin, J. and Boicea, A. (2015). Automatic Language Identification for Romance Languages Using Stop Words and Diacritics. 17th *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing.*