STEVENDU2018's system in VarDial 2018: Discriminating between Dutch and Flemish in Subtitles

Steven Du
stevenlol@qq.com

Yuan Yuan Wang
circlea@sina.com

Abstract

This paper introduces the submitted system for team STEVENDU2018 during VarDial 2018 Discriminating between Dutch and Flemish in Subtitles(DFS). Post evaluation analyses are also presented. The results obtained indicate that it is a challenging task to discriminate between Dutch and Flemish.

1 Introduction

The DFS task is a supervised learning task to classify text into Dutch or Flemish. Dutch is the language spoken in the Netherlands and Flemish is a variant of Dutch language and also known as Belgian Dutch. There are 300000 labeled training data, 500 labeled development data, 20000 on-hold test data. DUT in training data denotes Dutch, and BEL is the label for Flemish. F1 score is the evaluation metrics.

This paper is structured as follows: first, a brief training data analysis will be given. Then systems trained during the evaluation will be introduced. Finally more systems will be explored for post evaluation analysis.

2 Data analysis

The training data set consists of 300000 labeled sentences. After being lower cased and tokenized, the average sentence's length in characters and number of words for both DUT and BEL is nearly the same. As showed in Table 1, it is a well balanced data set. It is worth to note that the two languages share 57.2% of vocabulary.

Dialect	DUT	BEL
Number of samples	150000	150000
Average sentence length in characters	187.86	187.90
Average number of words per sentence	40.36	40.35
Unique words	115560	115442
Shared words	66142	
Percentage of shared words	57.2%	57.2%

Table 1: Statistics for the training data set.

One interesting finding is that the use of punctuation is a little bit different. BEL has more commas, periods and question marks but less exclamation marks than DUT as showed in Table 2.

3 Systems trained during evaluation

There are two systems trained during evaluation: a bag-of-ngram model and dual convolutional neural network model.

Dialect	DUT	BEL
,	157725	183736
.	690629	708076
?	118236	136742
!	1450	110

Table 2: Statistics for the punctuations in training data set.

3.1 Bag-of-ngram

Conventional methods for text classification adopted the typical features such as bag-of-words, n-grams, and their TF-IDF features (Zhang et al., 2008) as input of machine learning algorithms such as support vector machine (SVM) (Joachims, 1998), logistic regression (Genkin et al., 2007), naive Bayes (NB) (Mccallum, 1998) for classification.

In this work the bag-of-ngram system and Linear SVM is used as baseline system. First the text is lower-cased and converted to n-gram tokens (n is from 1 to 3), then filtered by TF-IDF with minimal document frequency of 5. Extracted features are used to train Linear SVM classifierr. A 20 folds cross validation is performed on the training set, the average F1 score is 0.63 and 0.69 is obtained on development set.

3.2 Dual-CNN

This approach builds simple CNN model (with pre-trained embedding) for each language. The input text will pass through these CNNs separately. Outputs of two CNN networks are then concatenated together. This is followed by a fully connected layer for classification task. Detail of this network can be found at github[1]. During evaluation the proposed Dual-CNN network obtain 0.62 through cross validation and score 0.61 on the development set.

The final submitted system is only a bag-of-ngram model which has better performance than Dual-CNN.

3.3 Evaluation results

Released test score is ranging from 0.55 to 0.66 in Table 3, our bag-of-ngram, the most simple approach yield 0.623. On the other hand proposed Dual-CNN yield 0.621. The test score correlated well with local cross validation score, development set is not a good choice for model selection. The best score is only 0.66, it implies the DFS task is challenging.

Rank	Team	Run	F1 (macro)
1	taraka_rama	3	0.6600474291
2	Taurus	4	0.6455823383
3	clips	2	0.6357352338
3	LaMa	3	0.6325606971
3	XAC	3	0.6317829736
3	safina	0	0.6308914957
4	**STEVENDU2018**	2	**0.6230923676**
4	mskroon	5	0.6201248435
5	SUKI	1	0.6127429864
6	DFSlangid	3	0.5961836466
7	dkosmajac	1	0.5674320041
7	benf	2	0.5582862249

Table 3: Evaluation results

[1] https://github.com/StevenLOL/vardial2018_dfs_stevendu2018/blob/master/dul-cnn.md

4 Post evaluation systems

Since bag-of-ngram system only score 0.623 on test set, to achieve better result a series of studies had been carry out after the evaluation. These can be broadly divided into three groups: one group focus on finding the vector representation for given text data, another group focus on deep learning approaches, third group utilize existing text classification framework.

4.1 Vector representation based approach

Vector representation approach intend to convert text data in variable-length pieces of text into a length fixed low dimension vector. There are many works have been done in this direction (Kim, 2014; Wieting et al., 2015; Kusner et al., 2015; Kenter et al., 2016; Ye et al., 2017), only two basic approaches are investigated: vector representation through take mean word vector and through doc2vec from the work in distributed representation of sentences and documents (Le and Mikolov, 2014).

4.1.1 Mean word vector system

A popular idea in modern machine learning is to represent words by vectors. These vectors capture hidden information about a language, like word analogies or semantic. Common used word vectors are word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014) and fastText (Bojanowski et al., 2017). Compare to word2vec, FastText is capable to capture sub-word information, thus in this study, we use FastText to train word vectors. Skip-gram, window size of 5 and minimal word count of 5 , 5 negative samples, sub-word range is between 3 and 6 characters are the default training parameters. After training, for each sentence, the mean word vector is used as its feature, Linear Discriminant Analysis classifier[2] is selected as the learning algorithm.

Word vector dimension	40	100	250	300	400
Test F1 Score	0.5642	0.5848	0.5922	0.598	0.6024

Table 4: F1 scores for mean word vector system

Table 1 shows F1 score for the mean word vector system. With increase of the length of word vectors, the system performance better. The 400 dimensional word vector is well suited for this task.

4.1.2 Doc2vec

In this study we use the doc2vec (Le and Mikolov, 2014) from gensim[3]. The doc2vec model is trained on training set with minimal word occurrence of 5 and window size of 8. Table 5 shows the best score is 0.5308, slightly better than random guess.

Sentence vector dimension	100	200	300
Test F1 Score	0.5282	0.5246	0.5308

Table 5: F1 scores for Doc2vec

Two set of sentence vector had been used in this study, the average word vector approach is better than doc2vec. In the following experiment, 400 is used as the default size of word embedding.

4.2 Deep learning based approaches

Our proposed Dual-CNN didn't beat the conventional bag-of-ngram model. This motivated us to examine the performance of deep learning approaches. Five type of deep learning based approaches are investigated, started from the most basic architecture, they are:

[2] http://scikit-learn.org/stable/modules/lda_qda.html
[3] https://radimrehurek.com/gensim/index.html

4.2.1 MLP

The MLP system is build by an embedding layer, one flatten layer and fully connected layer. Please refer to system diagrams in github repository[4].

4.2.2 AVERAGE

The Average system is similar to MLP system but the flatten layer is replaced by an average pooling layer. It is also known as neural bag-of-word model and being surprisingly effective for many tasks (Iyyer et al., 2015).

4.2.3 GRU

The GRU system is similar to AVERAGE system but the average pooling layer is replaced by a bidirectional GRU layer.

4.2.4 CNN-LSTM

The CNN-LSTM system is build by an embedding layer followed by two convolution-max pooling layers and one bidirectional GRU layer.

The four deep approaches are indeed most fundamental networks in NLP research. Incorporating language model fine-tunning (Howard and Ruder, 2018) and attention mechanism (Vaswani et al., 2017) is the recent trends, which we leave it for further exploration.

Word Embedding	D20 Random	D400 Random	D400 pre-trained
MLP	0.6350	**0.6365**	0.6334
AVERAGE	0.6352	0.6356	**0.6402**
GRU	0.6299	0.6388	**0.6413**
CNN-LSTM	0.6352	**0.6421**	0.6399

Table 6: F1 scores for popular deep learning based approaches

Table 6 presents the result of four popular deep learning based approaches. D20 Random denotes randomized word embedding of 20 dimensions is used in the network. D400 pre-trained denotes embedding layer is pre-trained with word vector size of 400 dimensions. This result confirms the observation in 4.1.1, that 400 dimension word vectors is a good choice for this task. Three out of four systems are higher than 0.64 which are significant better than submitted baseline system.

4.2.5 CapsuleNet

Capsules with transformation matrices allowed networks to automatically learn part-whole relationships. Consequently, (Sabour et al., 2017) proposed capsule networks that replaced the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. The capsule network has shown its potential by achieving a state-of-the-art result on highly overlapping digit parts in MutiMNIST data set. The PrimaryCapsule used in that paper is a convolutional capsule layer with 32 channels of convolutional 8D capsules. We increase the number of channels from 32 to 320 in this study, the assumption is that there are more part-whole relations in language than those in MNIST digit images.

Number of Channels	32	320	320
Output dimension	1	1	2
Test F1 Score	0.5992	0.6076	**0.6206**

Table 7: CapsuleNet Classification results.

Table 7 introduces F1 score of CapsuleNet on the test data set. The results indicate that with increase of number of channels and thus the number of capsules the system performance better. When changing the binary classification problem to two class classification problem, the capsule net yield comparable

[4] https://github.com/StevenLOL/vardial2018_dfs_stevendu2018

result to the bag-of-ngram baseline. The work by (Zhao et al., 2018) also shows significant improvement when transferring single-label to multi-label text classifications.

4.3 Text Classification Framework

FastText (Joulin et al., 2016) is a library for efficient learning of word representations and sentence classification[5]. It use vectors to represent word n-grams to take into account local word order, which is important for many text classification problems. Following Table 8 shows fastText classification results. The 0.6476 is the highest score achieved.

Word n-gram	1	2	3
Test F1 Score	0.6318	**0.6476**	0.6377

Table 8: FastText Classification results. The 0.6476 is the highest score achieved.

5 Conclusion

In this paper, a wide range of systems have been evaluated for the ValDial 2018 DFS task. A bag-of-ngram system score 0.6230 and serves as the baseline. Complex systems such as Dual-CNN and CapusleNet have competitive score to baseline system. Four simple deep learning based methods outperform baseline, three of them are higher than 0.64. FastText is identified as the best single system, yielded a F1 score of 0.6476.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexander Genkin, David D Lewis, and David Madigan. 2007. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304.

J. Howard and S. Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *ArXiv e-prints*, January.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé Iii. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 1681–1691.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg. Springer Berlin Heidelberg.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Tom Kenter, Alexey Borisov, and Maarten De Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. In *Meeting of the Association for Computational Linguistics*, pages 941–951.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. *ArXiv e-prints*, August.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *International Conference on International Conference on Machine Learning*, pages 957–966.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. 4:II–1188.

A Mccallum. 1998. A comparison of event models for naive bayes text classification. *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pages 41–48.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

[5] `https://github.com/facebookresearch/fastText`

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

S. Sabour, N. Frosst, and G. E Hinton. 2017. Dynamic Routing Between Capsules. *ArXiv e-prints*, October.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. *ArXiv e-prints*, November.

Jianbo Ye, Yanran Li, Zhaohui Wu, James Z. Wang, Wenjie Li, Jia Li, Jianbo Ye, Yanran Li, Zhaohui Wu, and James Z. Wang. 2017. Determining gains acquired from word embedding quantitatively using discrete distribution clustering. In *Meeting of the Association for Computational Linguistics*, pages 1847–1856.

W. Zhang, T. Yoshida, and X. Tang. 2008. Tfidf, lsi and multi-word in information retrieval and text categorization. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 108–113, Oct.

W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. *ArXiv e-prints*, March.