Language Identification from Text Documents

Priyank Mathur
Adobe Systems
Stanford University
Stanford, California 94305
priyankm@stanford.edu

Arkajyoti Misra
Target Corporation
Stanford University
Stanford, California 94305
arkajyot@stanford.edu

Emrah Budur
Garanti Technology
Stanford University
Stanford, California 94305
emrah@stanford.edu

***Abstract*—The increase in the use of microblogging came along with the rapid growth on short linguistic data. On the other hand deep learning is considered to be the new frontier to extract meaningful information out of large amount of raw data in an automated manner [1]. In this study, we engaged these two emerging fields to come up with a robust language identifier on demand, namely *Stanford Language Identification Engine (SLIDE)*. As a result, we achieved 95.12% accuracy in *Discriminating between Similar Languages (DSL) Shared Task 2015* dataset, beating the maximum reported accuracy achieved so far [2].**

***Index Terms*—Language identification, LID, SLIDE.**

I. Introduction

Automatic language detection is the first step toward achieving a variety of tasks like detecting the source language for machine translation, improving the search relevancy by personalizing the search results according to the query language [3], providing uniform search box for a multilingual dictionary [4], tagging data stream from Twitter with appropriate language etc. While classifying languages belonging to disjoint groups is not hard, disambiguation of languages originating from the same source and dialects still pose a considerable challenge in the area of natural language processing. Regular classifiers based on word frequency only are inadequate in making a correct prediction for such similar languages and utilization of state of the art machine learning tools to capture the structure of the language has become necessary to boost the classifier performance. In this work we took advantage of recent advancement of deep neural network based models showing stellar performance in many natural language processing tasks to build a state of the art language classifier.

We benchmarked our solution with the industry leaders and achieved first rank in the DSL test dataset.

II. Previous Work

In the past, a variety of methods have been tried like Naive Bayes [5], SVM [6], n-gram[7], graph-based n-gram[8], prediction partial matching (PPM) [9], linear interpolation with post independent weight optimization and majority voting for combining multiple classifiers [10] etc. and the best accuracy achieved are still in the lower ninety percents.

The researchers have worked on various critical tasks challenging the dimensions of the topic, including but not limited to, supporting low resource languages, i.e. Nepali, Urdu, and Icelandic [11], [12] handling user-generated unstructured short texts, i.e. microblogs [11], [10] building a domain agnostic engine [11], [8]. Existing benchmarking solutions approach the LID problem in different ways where LogR [11] adopts a discriminative approaches with regularized logistic regression, TextCat and Google CLD[13] recruits N-gram-based algorithm, langid.py [5] relies on a Naive Bayes classifier with a multinomial event model.

The outstanding results, of the time, suggested by Cavnar and Trenkle became de facto standard of LID even today [7]. The significant ingredient of their method is shown to use a rank order statistic called "out of place" distance measure [14]. The problem in their approach is that they generated n-grams out of words that requires tokenization. However, many languages including Japanese and Chinese have no word boundaries. Considering that Japanese is the second most frequent language used in Twitter [12], there is a need for better approach to scale the solution to all languages. As a solution to their problem, Dunning came up with a better approach with incorporating byte level n-grams of the whole string instead of char level n-grams of the words [14].

After a rigorous literature survey, we found no prior study that applied deep learning on language identification of text. On the other hand, there are a few number of studies that applied deep learning to identify the language of speech [15], [16], [17], [18]. We believe this study will be the first in the literature if published for LID in textual data by means of deep learning.

III. Dataset description

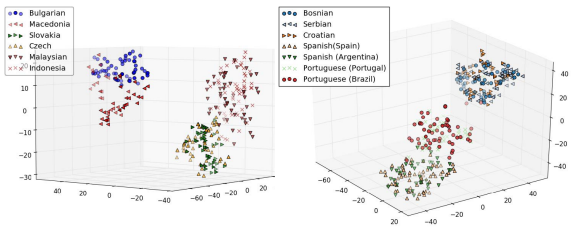
The data for this project work was obtained from "Discriminating between Similar Language (DSL) Shared Task 2015" [19]. A set of 20000 instances per language (18000 training (train.txt) and 2000 evaluation (test.txt)) was provided for 13 different world languages. The dataset also consisted of a subset (devel.txt) of the overall training data which we utilized for hyper-parameter tuning. The languages are grouped as shown in Table I. The names of the groups will be frequently referred in the subsequent sections.

Each entry in the dataset is a full sentence extracted from journalistic corpora and written in one of the languages and tagged with the language group and country of origin. A similar set of mixed language instance was also provided to add noise to the data. A separate gold test data was provided for the final evaluation (test-gold.txt).

We applied t-SNE algorithm to visualize the instances in 3D euclidean space [20], [21]. For feature extraction,

Group Name	Language Name	Language Code
South Eastern Slavic	Bulgarian	bg
	Macedonian	mk
South Western Slavic	Bosnian	bs
	Croatian	hr
	Serbian	sr
West-Slavic	Czech	cz
	Slovak	sk
Ibero-Romance (Spanish)	Peninsular Spain	es-ES
	Argentinian Spanish	es-AR
Ibero-Romance (Portuguese)	Brazilian Portuguese	pt-BR
	European Portuguese	pt-PT
Astronesian	Indonesian	id
	Malay	my

TABLE I: Benchmark results of available solutions



(a) Easily separable (b) Difficult to separate

Fig. 1: t-SNE visualization of language groups. More plots including 3D animated plot are available at : http://SeeYourLanguage.info

we vectorized each sentence over 1 to 5-grams of the tokens delimited by white space characters. Fig. 1 shows the resulting plot. As can be seen on the plot, the languages in the same group overlap a lot while the languages in different groups can be linearly separable. A 3 dimensional visualization of all the languages can be viewed at https://www.youtube.com/watch?v=mhRdfC26q78.

IV. Methods

A. Multinomial Naive Bayes

We created a baseline result by training a Multinomial Naive Bayes model because it is quick to prototype, runs fast and known to provide decent results in the field of text processing. We have done no pre-processing of the text commonly done in the field like stemming or stop word removal because we believe that could potentially remove important signatures of a particular language, particularly when the same language is spoken by two geographically disconnected group of people (e.g Portuguese spoken in Portugal and Brazil). We experimented with both word and character n-grams. The character n-grams turned out to be particularly useful when differentiating between two languages using mostly distinct character sequences in their alphabet.

The character level n-gram behaves quite differently from that of word level n-grams as shown in Fig. 2. Single characters carry little information and therefore the performance for character n-gram improves quite sharply as the number of characters is increased before saturating at about n=8. We experimented with character n-grams both restricted at word boundaries and spanning across word boundaries. The latter has a marginal performance boost at the cost of longer training time and memory pressure. The word n-gram model peaks at n=2 and drops beyond that. While higher order n-grams carry more structure of the language, they become increasingly infrequent too and therefore the models don't always get a boost from it. Both the character level and word level n-gram models show similar performance where they really excel at certain languages (Czech, Slovak) and do poorly at other (Bosnian, Croatian, Serbian).

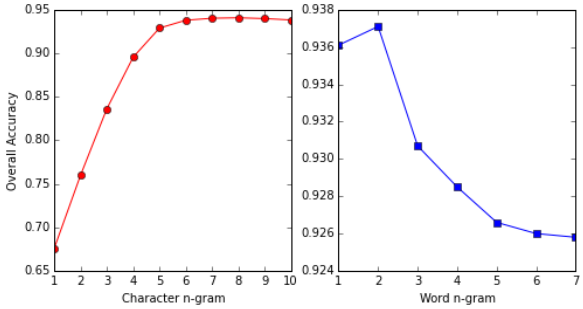


Fig. 2: Naive Bayes performance as a function of n for both word and character n-grams.

B. Logistic Regression

We next tried a regularized logistic regression and here too the character level n-gram performed a little better than the word n-grams. Fig. 3 shows that the model was able to completely fit the training set but the performance on the validation set plateaued close to 0.95. The best performance was obtained by a character 9-gram model that includes all n-grams up to n=9. These n-grams were truncated at the word boundaries, or in other words these n-grams did not capture two or more consecutive words. Relaxing this criterion significantly increases the size of the term frequency matrix and pushes the boundary of the computer memory but it does improve the performance by a fraction of a percent.

C. Recurrent Neural Network

The MNB and LR approaches work really well in distinguishing two languages that have very little in common because the set of n-grams will have very little overlap between them. This approach does not work very well when two languages are close to each other and share a lot of words between them. Therefore, it becomes necessary to capture the structure of a languages better to distinguish between similar languages. We explored Recurrent Neural Networks (RNN) for this purpose.

RNNs are a special kind of neural networks which possess an internal state by virtue of a cycle in their hidden units. As

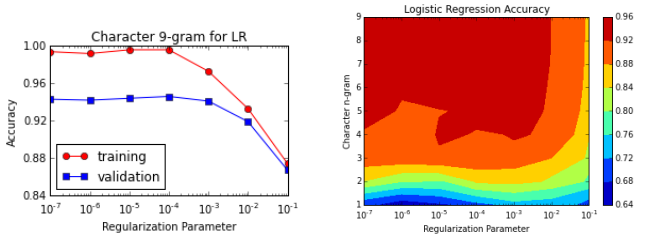


Fig. 3: The LR model was able to completely fit the training data but the accuracy on validation data peaked at about 94% overall.

such, RNNs are able to record temporal dependencies among the input sequence, as opposed to most other machine learning algorithms where the inputs are considered independent of each other. Hence, they are very well suited to natural language processing tasks and have been successfully used for applications like speech recognition, hand writing recognition etc.

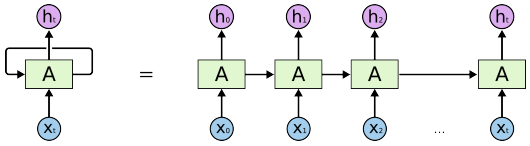


Fig. 4: Visualization of an un-rolled recurrent neural network [22]

Until recently, RNNs were considered very difficult to train because of the problem of exploding or vanishing gradients[23] which makes it very difficult for them to learn long sequences of input. Few methods like gradient clipping have been proposed to remedy this. Recent architectures like Long Short Term Memory (LSTM) [24] and Gated Recurrent Unit (GRU) [25] were also specifically designed to get around this problem. In our experiments, we used single hidden layer recurrent neural networks that used gated recurrent units.

Hyper-parameter tuning: In our single layer networks, we had three model hyper parameters to search over

1) Epochs - the number of iterations over training data. We generally try to train until the network saturates.
2) Hidden layer size - Number of hidden units in the hidden layer.
3) Dropout - Deep neural networks with large number of parameters are very powerful machines but are extremely susceptible to overfitting. Dropout provides a simple way to remedy this problem by randomly dropping hidden units as each example propagates through the network and back [26].

We used a subset of our overall training data (devel.txt) for hyper parameter selection. This subset was further divided into 75% training data and 25% validation data. In the first step of the process, we varied a single parameter while keeping the other two constant. The plots below (fig. 4) show the performance of the resultant models on the validation dataset as each parameter was changed.

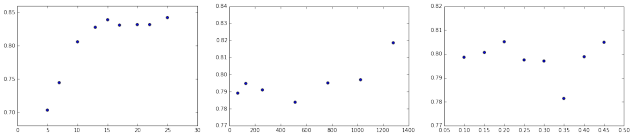


Fig. 5: Variation of the accuracy on validation dataset as we vary training epochs, number of hidden units and drop off

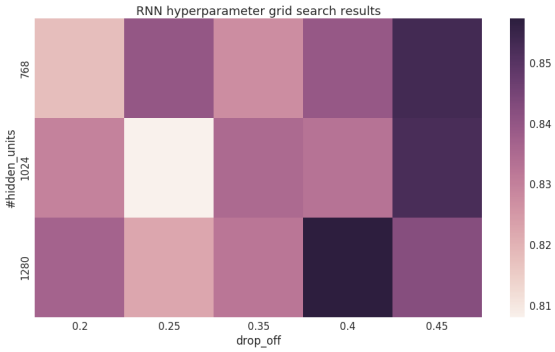


Fig. 6: Grid search over the best parameter values found in the previous step

As we can see in the plot above in Fig. 4, increasing the number of training epochs improves the model performance up to a certain stage, after which it plateaus. Hence, for the next stage of tuning, we fixed the number of training epochs to 20. Using the best values for the number of hidden units and dropout found above, we performed grid search over all combinations of these parameters. The result of the grid search is visualized in Fig. 5. The (number of hidden units, dropout) combinations (1280, 0.4) and (768, 0.45) gave us the best performance on the validation set. The final values chosen for further experimentation were 768 hidden units and 0.45 dropout so as to avoid overfitting.

Training procedure: Our final model is an ensemble of 5 RNNs, each built using a different feature set, namely, from character 2-grams to character 5-grams and word unigrams. To train our models, we divided our entire training data (train.txt) into 90% training set and 10% validation set. Once trained, we measured the performance of each model individually on the validation set and is reported in Table II.

As seen in Fig. 6, to construct the ensemble, instead of manually assigning weights to each model, we constructed a Logistic Regression model to get the final output. The features for this LR model were the outputs from the 5 RNNs created earlier and it was tuned using 5 fold cross validation over the 10% validation dataset.

For training the RNNs, we used a Python library called *passage* [27], which is built on top of Theano. Although the

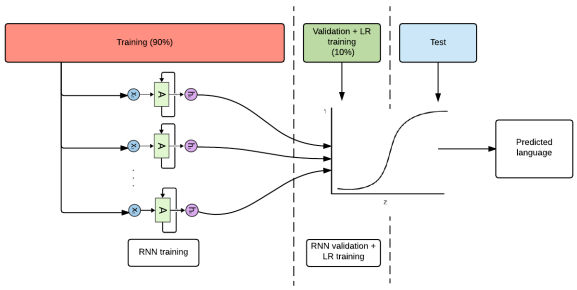


Fig. 7: Training procedure for the RNN ensemble

Model	Accuracy Validation Set	Test Set
MNB (char 9-gram)	0.9479	0.9452
LR (char 9-gram)	0.9486	0.9449
RNN (char 2-gram)	0.9200	0.9213
RNN (char 3-gram)	0.9328	0.9338
RNN (char 4-gram)	0.9377	0.9347
RNN (char 5-gram)	0.9347	0.9316
RNN (word uni-gram)	0.9351	0.9330
Ensemble of RNN model (SLIDE)	0.9533	0.9512

TABLE II: Performance comparison of various models

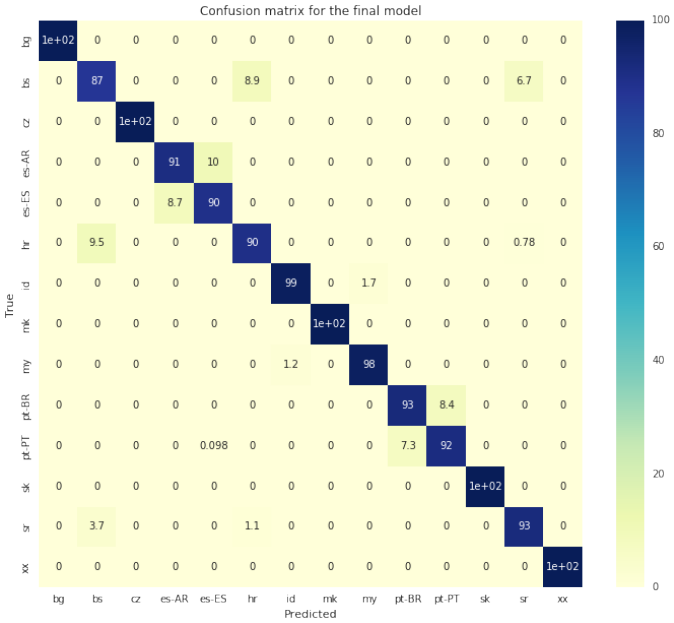


Fig. 8: Confusion matrix

library provides several tools for text pre-processing including tokenization, it lacked the ability to generate character n-gram level features. Therefore, we had to extend the library with custom character level feature generators. In addition, training neural networks on CPUs consumes a lot of time. Hence, for our experiments, we leveraged AWS GPU (g2.2xlarge) instances that provided a 10x boost in time required to train one model.

V. Results

Table II shows a comparison of the models we have experimented with. One surprising feature of the result is that individual RNN models were not able to beat the performance of the MNB and LR models, even though the latter models have minimal knowledge of a language structure. However, when we created an ensemble of RNN models, it turned out to be the best model and crossed the 95% threshold for the first time. It should be noted that for a particular n-gram model, MNB and LR models use all m-grams where $1 \leq m \leq n$. However, due to the very nature of an RNN architecture, a combination of n-grams cannot be used because that will lead to an overlapping sequence of content to be fed to the network. Since any given n-gram captures only limited information about a language, it was natural to try an ensemble of n-gram RNN models with different values of n, so that structure of the language can be captured at multiple different levels.

The boost in performance due to ensemble can also be attributed to model combination, which aims to achieve at least as good of a performance as the worst model in the ensemble. This is because individual models can make mistakes on different examples, and therefore, by using an ensemble we are able to reduce this variance. While we tried other model combination strategies like median and manual weighting, building a Logistic Regression classifier on top of RNNs really helped us find the optimal weight that should be given to each individual model. We could not include RNN models beyond character 5-gram in the ensemble because of memory limitation and including the MNB or LR model in the ensemble did not improve the performance of the model.

VI. Discussion

The final classification for each language group is captured in the confusion matrix in Fig. 8. It is quite evident from our results that the biggest challenge consistently posed to our classifiers is distinguishing the languages in South Western Slavic group (*bs, hr, cr*). The training set revealed that among all the words in *bs*, 48% are common to *hr* and 41% to *sr*. Since Fig. 3 clearly showed we didn't underfit the training set, it made sense to augment the training data in these three language categories. We incorporated a significantly larger labeled data for two of these languages and also downloaded newspaper articles in *bs*, but the classification accuracy in this language group did not improve. Looking closer to some of these external datasets revealed that none of the new words could be uniquely associated to any of the three languages and therefore, the additional data probably added more noise than signal.

To understand the failure mechanism of the classifier for the South Western Slavic language group, we fed the LR classifier, which is the best of single models in validation set according to Table II, different fractions of a document it failed to classify correctly. For example, the following document is in Bosnian(*bs*) but the classifier predicts its language as Serbian(*sr*): *Usto se osvrnuo na ekonomsku situaciju u kojoj*

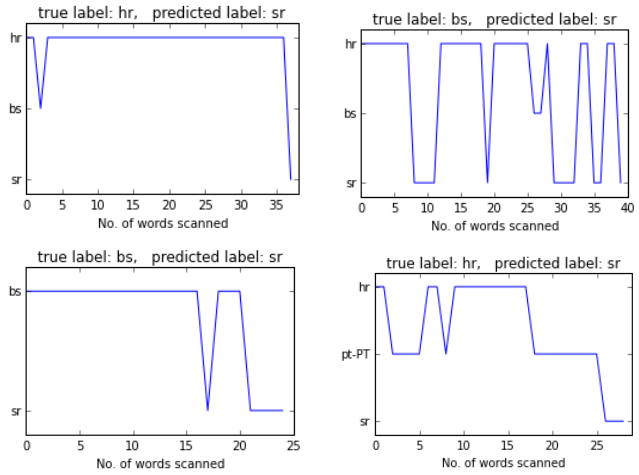


Fig. 9: Scenarios where the LR classifier incorrectly predicted the target language. (Detailed description in Section VI)

je veliki broj novinara u potrazi za poslom, na mizerne plae i guranje etike strane profesije u zapeak. So we fed the classifier with "*Usto*" and noted the prediction, then fed it with "*Usto se*" and noted the prediction, and so on until the full sentence is fed. The classifier prediction at different stages of the sentence scan is plotted in Fig. (9). The top left panel of Fig. (9) shows that the classifier for the most part thinks the document to be actually *bs*, until it saw the last word of the sentence when it switched its prediction to *sr*. We think this is due to the fact that the last word associated very uniquely to *sr* in the training corpus. The bottom left panel of Fig. (9) shows a similar scenario but in this case the classifier switched back and forth a couple of times. The 'confusion' of the classifier is very high in the top right panel of the figure because the particular sentence was made of words and phrases that are common to all three languages. We believe that the correct classification of such documents needs creation of extra features based on deeper understanding of this language group. Another possible scenario where any classifier can struggle is when the body of the text contains a quotation of a different language. The bottom right panel of Fig. (9) shows a scenario where a document in Serbian had a comment in Portuguese, though that was not the cause of the eventual classification failure. Removing quotes from a document is a potential option but it can also have adverse effect if the quote is in the same language as that of the main document.

VII. Conclusion and next steps

We have presented a deep neural network based language identification scheme that achieves near perfect accuracy in classifying dissimilar languages and about 90% accuracy on highly similar languages. Specifically, the languages in Western Slavic Slavic group posed the highest challenge. And expanding the corpus of these languages using external sources did not help much mainly because no n-grams of words that are unique to certain languages were ingested by the expanded part of the corpus. We have relied on the ensemble of RNN models to discover the structure unique to a specific language but we could not engineer any additional feature due to lack of knowledge in those specific languages. At this point, we think, further improvement can only be achieved by designing rule based features by talking to language experts or native speakers.

Acknowledgment

We would like to thank *David Jurgens* of Department of Computer Science, Stanford University for helping with the initial idea, dataset and previous research, *Junjie Qin* for his mentoring and insightful comments that polished the outcome of the study, *AWS Educate Program* for providing EC2 credits and computing resources and *Microsoft Azure for Research Program* for providing Azure credits and full featured computing resources, and lastly *Google*, *Yandex* and *Basis Tech* for providing free access to their language detection APIs for our benchmarking analysis [28] [29] [30].

Comparison with other systems

We assessed the performance of SLIDE by comparing its result with the domain leaders in an *unfair* test described below.
We queried the *test file* of dataset of DSL Shared Task 2015 and accepted the resulting predictions *even if*

- the dialect of the language is not distinguished in Ibero-Romance language group due to lack of support, i.e. Google always predicts *Portuguese* for sentences both in Brazilian Portuguese and European Portuguese.
- a certain language is not supported at all, i.e. Rosette doesn't support Bosnian.

Table III shows the resulting accuracies. Although SLIDE had lack of competitive advantage in this unfair test, it surpassed the industry leaders in terms accuracy.

Solution	Accuracy
SLIDE	95%
Google Translate API	89%
Rosette Language API	86%
langid.py	80%
Yandex Translator API	79%

TABLE III: Benchmark results in non-increasing order of accuracy

REFERENCES

[1] I Arel, D C Rose, and T P Karnowski. Deep machine learning - a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010.

[2] Marcos Zampieri, Liling Tan, and Nikola Ljube. Overview of the dsl shared task 2015. (2014), 2015.

[3] Juliane Stiller, Maria Gäde, and Vivien Petras. Ambiguity of queries and the challenges for query language detection. *CLEF 2010 Labs and Workshops Notebook Papers*, 2010.

[4] Dong Nguyen and a Seza Do. Word level language identification in online multilingual communication. 23(October):857–862, 2013.

[5] Marco Lui and Timothy Baldwin. langid. py: An off-the-shelf language identification tool. *Proceedings of the ACL 2012 System Demonstrations*, (July):25–30, 2012.

[6] Aditya Bhargava and Grzegorz Kondrak. Language identification of names with svms. *Computational Linguistics*, (June):693–696, 2010.

[7] William B Cavnar, John M Trenkle, and Ann Arbor Mi. N-gram-based text categorization. *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[8] Erik Tromp and Mykola Pechenizkiy. Graph-based n-gram language identification on short texts. *"Proceedings of the 20th annual Belgian-Dutch Conference on Machine Learning"*, pages 27–34, 2011.

[9] Victoria Bobicev. Native language identification with ppm. pages 180–187.

[10] Simon Carter, Wouter Weerkamp, and Manos Tsagkias. Microblog language identification: overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, 47(1):195–215, 2012.

[11] Shane Bergsma, Paul Mcnamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. Language identification for creating language-specific twitter collections. *"Proceedings of the 2nd Workshop on Language in Social Media at NAACL-HLT'12"*, (Lsm 2012):65–74, 2012.

[12] Semicoast. Half of messages on twitter are not in english japanese is the second most used language. *Semiocast*, page 75005, 2010.

[13] Dick Sites. Google compact language detector 2, 2013.

[14] Ted Dunning. Statistical identification of language. *Computing*, (November), 1994.

[15] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro J. Moreno. Automatic language identification using deep neural networks. *Icassp-2014*, pages 5337–5341, 2014.

[16] Grégoire Montavon. Deep learning for spoken language identification. *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, pages 1–4, 2009.

[17] Bing Jiang, Yan Song, Si Wei, Jun Hua Liu, Ian Vince McLoughlin, and Li Rong Dai. Deep bottleneck features for spoken language identification. *PLoS ONE*, 9(7):3012–3016, 2014.

[18] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J. Moreno. Automatic language identification using long short-term memory recurrent neural networks. In *Interspeech-2014*, pages 2155–2159, 2014.

[19] Lt4vardial workshop.

[20] t-sne laurens van der maaten.

[21] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 1 2014.

[22] Understanding lstm networks – colah's blog.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning*, (2):1310–1318, 2013.

[24] Sepp Hochreiter Schmidhuber and Jürgen. Long short-term memory. *Neural Computation*, 9(8):1735 – 1780, 1997.

[25] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv*, pages 1724–1734, 2014.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 1 2014.

[27] Indicodatasolutions/passage.

[28] Google translate api - programmatic translation services google cloud platform.

[29] Translator api yandex technologies.

[30] Rosette api — basis technology.