Theatrical and Cinematic Textual Speaker Segmentation with RNNs

Aidan Clark, Alon Daks, and Daniel Nguyen
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
{aidanbclark, alon.daks, dnguyen44}@berkeley.edu

Abstract

This paper formulates a new problem known as Textual Speaker Segmentation (TSS) and proposes a framework for solving it through the use of recurrent neural networks. Textual speaker segmentation is the problem of identifying speaker transitions when given continuous text representing output from multiple persons speaking. In particular, we examine the efficacy of segmenting theatrical and cinematic dialogue into speaker components. We establish simple baselines on this problem, and additionally provide first benchmarks of deep networks on this problem. We show experimentally that using recurrent neural networks substantially outperforms baseline methods drawn from traditional machine learning.

1 Problem Statement

Given a sequence of sentences, where the speaker changes between some, but not all, sentence divisions, we wish to develop a model that can accurately segment the larger sequence of sentences into contiguous sub-sequences of text all belonging to a specific speaker.

For example, suppose a program is given the following text, a continuous steam of words representing speech from two characters:

You want answers? I think I'm entitled to them. You want answers? I want the truth! You can't handle the truth! Son, we live in a world that has walls. And those walls have to be guarded by men with guns. Who's gonna do it? You?"

In that paragraph there are eight sentence transitions, and out of those eight transitions four of them represent a transition from one speaker talking to another, and four represent two sentences both spoken by the same speaker. We define the problem of sentence-level Textual Speaker Segmentation (TSS) to be the following: given such an input of text, label all sentence transitions as either representing a change of speaker, or alternatively as a continuation of one speaker. Our model will not be responsible for learning to identify or classify the speaker for any sentence, just to identify where transitions occur. For example, the sentences "I think I'm entitled to them." and "I want the truth!" were spoken by the same person, but identifying this is not part of our problem.

Specifically, we define our problem to be the following: given s_1, s_2, two sentences which are in sequence, determine the probability that s_1 and s_2 were spoken by the same speaker. We note that there is an additional formulation of the problem which is defined at the word-level, and is formulated as such: given $\{s_1\}$, a sequence of sentences, determine $\{a_i\}, a_i \in [0, 1] \forall i$, a sequence of numbers where a_i indicates the probability of s_i being spoken by a different speaker than s_{i-1}.

In this paper we will give results only on the first problem formulation, but we will discuss the latter in the conclusion and note that, while the latter problem has more direct applicability, we believe results and methodologies developed on the former problem will be transferable to the latter one. We consider only the first problem for several reasons. First, it is more compatible with traditional machine learning methods which struggle with time-dependent information. Second, the lack of baseline results on the former problem make it hard to judge the second, as the latter is considered a more difficult problem to learn.

2 Background

Many personal assistant applications (such as Siri or Cortana) utilize a break in sound to determine

that the conclusion of a question has been reached. This runs into obvious limitations, both when used in a noisy location, and also in situations where the speaker continues to talk, even if the recipient of the speech is obviously not the personal assistant.

Traditional speaker segmentation work is focused on solving a problem of overlapping speakers, and work published recently (such as Ma et al. 2015) focuses on building deep architectures to isolate speakers from ambient sound. An HMM-based model proposed in (Woubie et al., 2016) is the most recent and state-of-the-art. All approaches rely on auditory signal data as the primary information.

Our problem, while also one of speaker segmentation, takes a different approach than other work in the field. We assume the speech has already been synthesized into text, and instead of separating one speaker from several speakers talking at the same time, we wish to segment one speaker from other speakers speaking directly before and after. The crucial distinction is that our work focuses on segmentation that can be done at the language level, while previous work focuses on segmentation done at the auditory level. This makes our approach distinct and complementary to that explored previously.

Having such a trained network would provide a heuristic to a personal assistant to allow it to determine the conclusion of a question independently from continued noise or speech. Additionally, many state-of-the-art neural networks (such as Sequence-to-Sequence models) rely on the input of a special end-of-sequence token to signify that the encoder phase is complete, and to tell the network to begin decoding. Having a network like this would allow a Sequence-to-Sequence model to automatically identify when it should stop listening and start speaking, without required a manually inputting end-of-sequence sign.

Additionally, we believe that a solution to TSS could also provide insights into a different and related problem known as Unsupervised Authorial Decomposition (UAD). UAD, first proposed by (Aldebei et al., 2016), is the process of taking a document written by multiple authors, and partitioning sequences of sentences written by distinct authors. It is quite similar to our TSS setup, but instead of predicting speaker changes, UAD requires predicting author changes. UAD is an active area of research, and novel techniques have recently been proposed using a very similar model to those used for auditory speaker segmentation (Aldebei et al., 2016). While acknowledging the distinction between author and speaker, we believe these two problems can benefit from one another.

3 Approach

3.1 Dataset

Since no dataset had been curated for a problem like ours, we spent time collecting and curating a robust corpus suitable for benchmarking and evaluating our work. We produced two suitable corpora. While any textual dataset with speaker information would be sufficient, we highlight plays and screenplays as two cases where speaker information is easily accessible, and those changes in speaker are commonplace.

The first dataset, inspired by the work of (Karpathy, 2015), is the concatenated plays of Shakespeare, 33 in total. These plays were scraped from their raw form from Project Gutenberg, and then parsed using regular expressions and basic python into a CSV file.

In that file, each row in the CSV corresponds to a particular line from a play, along with the associated character that spoke said line. From the full text of plays, we generate 111,359 sentences pairs (all pairs of contiguous sentences), each of which are labelled either 1 or 0: indicating that they represent two sentences that were (resp. weren't) spoken by different speakers.

The second dataset is adopted from the Cornell Movie-Dialogs Corpus. This dataset consists of 220,579 conversational exchanges between 10,292 pairs of movie characters. It involves 9,035 characters from 617 movies, yielding a total of 304,713 utterances. After similar preprocessing we produced approximately 570,000 contiguous sentences pairs, each labelled either 1 or 0 like the Shakespeare set.

3.2 Baseline Model

To remind the reader, our problem takes the following form: Given a set of sentence pairs s_1, s_2 that are known to be sequential, predict whether or not s_1 and s_2 originate from the same speaker. This is tantamount to predicting the probability of this transition occurring, and this is the method of labelling we use.

For all baseline models be experimented with, we needed to represent each pair of sentences as

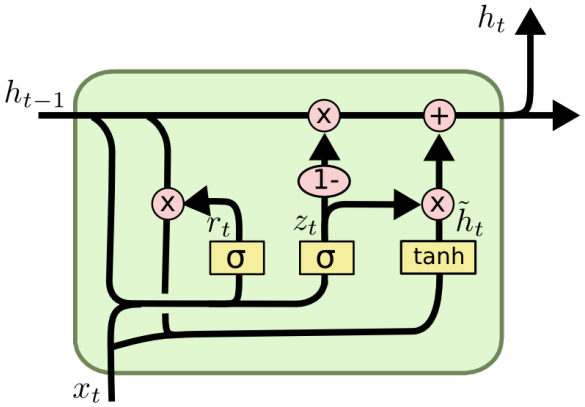


Figure 1: A nice visualization of an GRU cell due to (Olah, 2015). The top horizontal line represents s_t, whereas the bottom one represents h_t. The output of the yellow boxes (in left-to-right order) are f_t, i_t, c_t and o_t. More detail can be found on his blog.

a single data point. In all cases, we represented each sentence as the sum of one-hot word vectors of each word in the sentence, but subsequently discovered that the results of our model were highly dependent on our choice of function for combining the two sentence vectors into one.

After initially attempted to use summation and concatenation as our methods of vector combination, both of which failed to achieve accuracies higher than randomly guessing, We adopted the subtraction method: taking difference of the two vectors $x_{12} = x_{s_1} - x_{s_2}$ as the input to our classifier. This formulation has the interpretation that the classifier is given the difference in word usage between two sentences, and attempts to classify transitions based on this knowledge. Overall, we found that this choice of combination resulted in approximately a 2% accuracy increase over other methods.

This performs poorly. We used a Logistic Regression classifier as our first baseline: which resulted in 76.6% accuracy on the Shakespeare dataset. Initially exciting, this accuracy turned out to be disingenuous. Because most speakers continue speaking for most than a single sentence, our dataset was extremely skewed, with approximately 73.1% of sentence pairs having the negative label. When running our classifier specifically on validation sets of only positive and only negative samples, we noting that the classifier would correctly classify 96% of negative samples as negative, and only 24% of positive samples as positive: almost exactly reflective of the distribution in the data.

Clearly, our classifier was learning to mimic the skew, not to meaningfully classify based on features. To combat this, we took 30,000 negative samples and 30.000 positive samples, randomly permuted them and trained our Logistic Regressor on this new dataset. This generated our first substantial prediction, with the resulting classifier having an overall 63.5% validation set accuracy, with 68.0% and 70.3% training accuracy on only the positive and negative samples respectively.

Naive variations on this method achieved little improvement. We compared the accuracy of the Logistic Regressor with a Linear Classifier, as well as with a Random Forest with 50 trees. These alterations made a negligible change in performance.

3.3 Recurrent Model

We propose to solve this problem using a recurrent neural network (RNN). We believe that the structure of this problem is extremely suited to such a network, which would note have to simplify over timesteps and be able to directly utilize sequential information (which is lost in the baseline). Additionally, the increased expressive power of a RNN will allow this model to learn more meaningful results.

Gated Recurrent Units (GRUs), introduced by (Cho et al., 2014) are very similar to Long Short-Term Memory units first introduced by (Gers and Schmidhuber, 2000) and improved by (Hochreiter and Schmidhuber, 1997) , but have simplified operation, including the removal of a separate hidden state. The update rules for a GRU unit is given below.

$$
\begin{aligned}
z_t &= \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z) \\
r_t &= \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r) \\
\hat{h_t} &= \phi(W_{\hat{h}h}(r_t \bullet h_{t-1}) + W_{\hat{h}x}x_t + b_c) \\
h_t &= (1 - z_t) \bullet h_{t-1} + z_t \bullet \hat{h_t}
\end{aligned}
$$

Here, ϕ represents the hyperbolic tangent and W with subscript represents the weight matrices of the recurrent unit. h_{t-1} represents the output at the previous timestep, and x_t the input at the current timestep. b with a subscript represents the bias vectors. h_t represents the hidden state of the

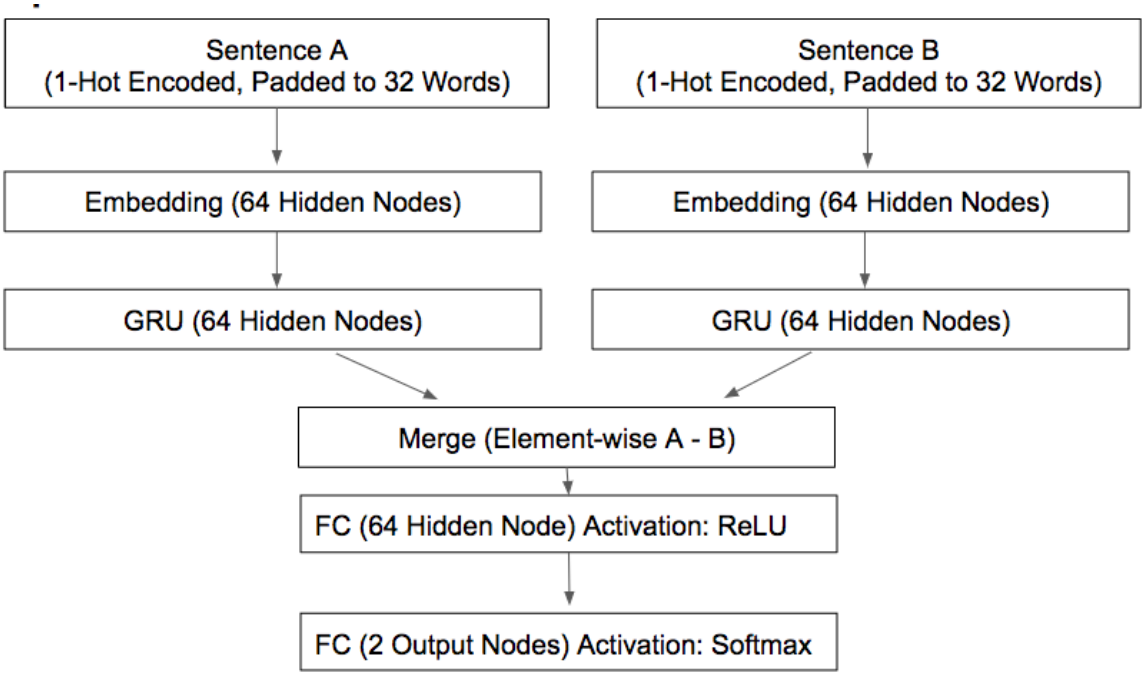


Figure 2: An overview of our recurrent model. Each tail was independently fed a sentence, and was recurrent in the context of that sentence. At each time step, each tail produced an output which was then merge with the other sentence's timestep via subtraction, then fed into two fully connected layers to produce a final prediction.

GRU unit at time t, and σ represents the sigmoid function[1]. The $\bullet$ represents the hadmard product: an element-wise multiplication.

Our model was a two-tailed three layer neural network with a single recurrent layer on at each time. Each tail of the network would be trained to produce a good representation of the underlying sentence, and then later layers would be trained to segment it. An overview of this model is given in 2.

4 Tools

Baseline models were implemented using an ensemble of tools from the Numpy (Van Der Walt et al., 2011), Scipy (Jones et al., 2001) and Scikit Learn (Pedregosa et al., 2011) libraries. Minor components from the Gensim Topic Modeling library and (Řehřek and Sojka, 2011) from NLTK (Bird, 2006) were used for data parsing. Both NLTK and Gensim are Python libraries for textual parsing and natural language processing, and all steps proceeded smoothly.

We implemented our recurrent model using the Keras (Chollet, 2015) Python library with a TensorFlow (Abadi et al., 2015) backend. This was chosen both for ease of development and deployment, and also because of the preexisting experience our team had with these tools.

[1]In practice we use an numerically efficient piecewise linear approximation of the sigmoid function.

Corpus	Baseline	Final Model	Improvement
Shakespeare	76.6%	**97.5%**	+20.9%
Movies	52.2%	**71.4%**	+19.2%

Table 1: Results on test sets for Shakespeare and Movies Corpora, comparing baseline Logistic Regression with our final recurrent model.

5 Results

All of our models were deployed on Keras with a TensorFlow backend as described above. We did our training on a Macbook Pro with no GPU, as the moderate size of our network forced less of a necessity on GPU training. We trained using the ADAM gradient optimizer (Kingma and Ba, 2014), which our team has empirically observed to outperform related methods like RMSProp. Note that a large number of Epochs (roughly 1000) were needed before accuracies substantially improved

to where they are reported being.

To test our models, we split both Shakespeare and Movies datasets into training and testing partitions according to an 80% / 20% respective division. Plots on the top of the next page show training accuracies and losses as a function of number of trained epochs for the final RNN model on both the Shakespeare and Movies corpora. Table 1 summarizes test accuracies of our final model and compares those results to accuracies achieved by our baseline. As outlined in the table, using a reccurent neural network greatly outperforms the baseline model. Our RNN-based framework improves on simple logistic regression by around 20% accuracy. These results are sensible because sentence structure is critically important towards segmenting sentences. Since classical machine learning methods are limited at their point of featurization (bag-of-words) in that any sensible form of featurizing a sentence will lose interesting structure, recurrent networks are ultimately necessary since they are capable of learning relevant features.

5.1 Shakespeare

As noted above, the baseline model achieves 76.6% test accuracy on the Shakespeare corpus. This is an exceptionally poor result since the ground-truth distribution is about 73.1% class 0 (no speaker change) and 27.9% class 1 (speaker change). Upon further exploration, we see that the model is essentially learning to always guess class 0, indeed hardly "learning" at all. Our recurrent model achieves 97.5% accuracy on the Shakespeare test set, substantially outperforming the baseline, and more critically learning meaning future structure towards distinguishing class 0 sentence pairs from class 1 pairs. Finally, since the Shakespeare corpus was written entirely by a single author, our results are controlled for any influence that distinct author might have in dictating the speech of disparate characters.

5.2 Movies

The results on the Movies corpus are consistent with those we have achieved on Shakespeare. The baseline test accuracy of 52.2% matches the underlying ground-truth distribution, while the final test-accuracy of 71.4% is a substantial improvement that indicates meaningful learning had occurred. Both baseline and final accuracies on the Movies corpus are lower than accuracies achieved on the Shakespeare corpus due to the underlying differences in the ground-truth class distributions. Finally, results on the Movies corpus should be regarded as being more robust than the Shakespeare results since a) many more distinct scenes and character exchanges were present and b) many different authors contributed to the dataset.

6 Conclusion

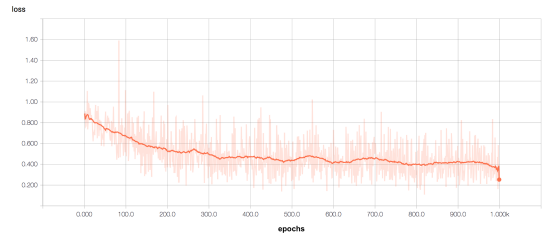
Our experiments with the baseline model showed that simple classifiers struggled to predict speaker changes with any accuracy, reporting just an incremental improvement over random guessing once the skew of the data is taken into consideration. We also showed that forcibly removing this skew during training does actually result in a slight improvement in the baseline model, but just negligibly.

Switching our model from a logistic classifier to a three layer neural network with a recurrent component, we showed that a deep network could reliably learn to differentiable between sentences spoken by the same speaker in succession and sentences spoken one-after-another by two different speakers. We showed that this network, once trained, could increase our prediction accuracy by between 19.2% and 20.9% over the baseline, representing an understanding of meaningful markers of differentiation in speakers.

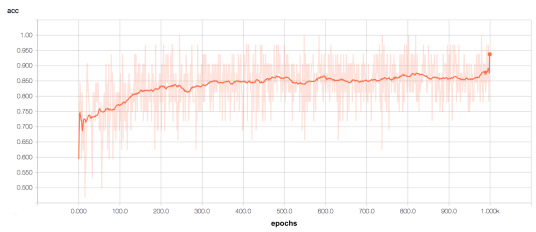
7 Future Work

There are several directions in which future work could proceed. Due to time constraints and technical difficulties, the two tails of our model were trained independently from one another. While there is strong reason to believe that the weights of each model would, given sufficient training time, converge to equivalent weights, in many cases our model would require each tail of the network to learn the same connections independently. An obvious direction of improvement would be to implement weight sharing between the two tails of the network: both in the embedding and recurrent layers.

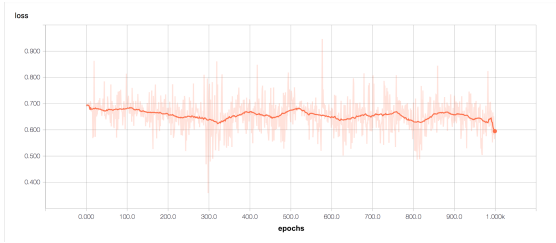
Additionally, our model was just one of several different neural network types we considered to tackle this problem. Alternatives that were proposed include one-dimensional convolutional networks, deeper stacked LSTM or GRU layers, and Siamese networks, whose design inspired the two-headed nature of our model. Al-



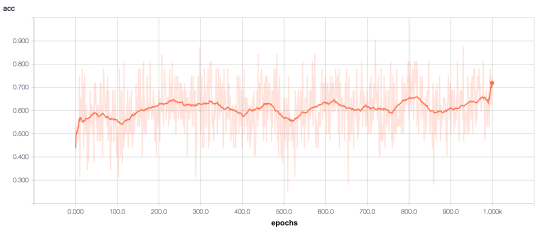
(a) The loss curve for our recurrent model on the Shakespeare dataset.



(b) The validation accuracy for our recurrent model on the Shakespeare dataset.



(a) The loss curve for our recurrent model on the Movie Caption dataset.



(b) The validation accuracy for our recurrent model on the Movie Caption dataset.

though time restrictions and the efficacy of our chosen GRU model kept us from experimenting with other model types, future work could easily expand on our work by comparing our performance with the performance gained from any of these alternate model times.

Finally, our original goal was two examine two different problem statements. The first, the problem we ended up tackling, was to determine speaker changes on a sentence-by-sentence level. The second was more granular, and the goal of the second problem formulation was to be able to give word-level predictions of speaker change for every word when linearly proceeding through text.

The first problem ended up being harder than expected, and we were forced to spend the majority of the problem establishing baselines for that problem. Now that those baselines have been set, further work could expand our results by comparing the accuracies we got with accuracies gotten on the word-level problem, either using our network or any equivalent network.

8 Open Source

Code for our project is accessible at: https://github.com/Cogitans/cs294

9 Team Contributions

Aidan Clark (1/3 effort), Alon Daks (1/3 effort) and Daniel Nguyen (1/3 effort) each contributed to the project evenly.

Aidan initially found and parsed the Shakespeare dataset, as well as developing and running the baseline model used for the first half of the project. Additionally, he wrote the framework in which the recurrent model was later implemented. He also wrote the mid-semester checkpoint paper, and contributed equally with Alon and Daniel to the slides throughout the semester and to the final writeup.

Alon did much of the data cleaning and preprocessing including: identifying and parsing the Movies corpus and writing drivers between data CSV files and Keras models. Additionally, he designed the final RNN model and produced the initial implementation of it in Keras. He designed and completed the class poster. Finally, he contributed equally with Aidan and Daniel to the slides throughout the semester.

Daniel focused on processing the data. He worked on initial designs for the neural models and developed the generators to input the data into the Keras models. Additionally, he worked on training the final version of the models and collecting results data. Finally, he contributed equally with Aidan and Alon to the slides throughout the semester.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin,

Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Khaled Aldebei, Xiangjian He, Wenjing Jia, and Jie Yang. 2016. Unsupervised multi-author document decomposition based on hidden markov model. In *The 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet. 2015. Keras. https://github.com/fchollet/keras.

Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

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

Eric Jones, Travis Oliphant, P Peterson, et al. 2001. Open source scientific tools for python.

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christopher Olah. 2015. Understanding lstm networks. Accessed: 2016-11-28.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

R Řehřek and P Sojka. 2011. Gensim–python framework for vector space mo delling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*.

Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Abraham Woubie, Jordi Luque, and Javier Hernando. 2016. Short-and long-term speech features for hybrid hmm-i-vector based speaker diarization system. Odyssey.