DT2119 Project Report

Contextual word embedding using speech

Anirudh Seth (aniseth@kth.se) Reza Dadfar (dadfar@kth.se)
Prashant Maheshwari (pramah@kth.se)

Spring 2020

Abstract

Acoustic Word Embedding (AWE) is a fixed-dimensional representation of a variable-length audio signal in an embedding space. The motivation behind the approach is to capture the rich information from speech and to eliminate the errors caused by the process of transcribing. AWE have started gaining popularity because they are easy to train, scale well to large amounts of training data, and do not require a lexicon. In this project, we investigate recent publications, literature and implement a LSTM Encoder-Decoder framework that borrows the methodology of skipgram and continuous bag-of-words to learn contextual word embeddings directly from speech data. The word embeddings are then evaluated on widely used word similarity benchmarks and compared with the embeddings learned by Word2vec and fastText. Experimental results from our model show that it is in fact possible that the embeddings learned from speech outperform those produced by its textual counterparts.

1 Introduction

Natural language processing using a computer requires a fixed sized representation for variable length data like words, sentences, paragraphs, and documents. These embeddings are almost ubiquitous in all recent NLP research. Widely used techniques such as Word2vec [1], fastText [2] and GloVe [3] transform words from text into fixed dimensional vectors. These vectors are obtained by unsupervised learning from co-occurring information in the text, and contain semantic information about the word which are useful for several NLP tasks like machine translation [4], abstract summarization [5] and many more.

Speech is another form of language but so far it is rarely used as a source for learning semantics compared to text. Some of the prior work have used supervised [6], [7], [8], and unsupervised/semi-supervised [9], [10], [11] learning methods to find out the acoustic word embeddings. In [6] and [8], the word embeddings are a byproduct of speech recognition task. The drawbacks of these methods are the same as that of speech recognition namely it requires a volume of annotated data and bulky language models. The work in [10] and [11] have explored the usage of Convolutional Neural Network to learn the speech embedding using hinge loss and multi-modal similarity loss respectively.

In this project, we study, review and propose a model which takes inspiration from Speech2Vec [9]. The model is capable of representing raw audio segments from a speech corpus as a fixed dimensional vector. These embeddings can take the advantage of additional information in speech that is not present in text (such as emotions, variability of speakers, etc) and mitigate the recognition errors caused by the process of transcribing to text. The design of the proposed model uses a Recurrent Neural Network [12] based Encoder-Decoder framework and uses the methodology of skipgram and continuous bag of words (CBOW) from the widely used Word2vec [1] model. We compare our results with fastText [2] and Word2vec on three word similarity datasets - WS-353 [13], WS-353 Rel [13], and Verb-143 [14].

2 Dataset

For this project, we worked with LibriSpeech dataset [15]. The dataset contains 1000 hours of English speech data sampled at 16 kHz which is based on audiobooks from the LibriVox project. Due to limited computational resources and time, we used a subset of the data for our implementation i.e. ”dev-clean”.

2.1 Forced Alignments

The LibriSpeech dataset contains sentence level audio waveform from different speakers. To learn word level embeddings, the waveforms of these sentences need to be segmented according to the word boundaries. Forced alignment is a technique that uses a pronunciation dictionary and an orthographic transcription of audio data to generate time aligned data. The authors in Speech2Vec [9] did not explicitly mention one, thus we used the results from Montreal Forced Aligner [16], a widely recognized framework for speech-text alignment, for our dataset. This aligner goes through from four primary stages of training.

1. All the phones are modelled similarly irrespective of the phonological context.
2. At this stage, context on either side of phone is also utilised by using triphone model.
3. LDA and MLLT [17] is used to learn the transformation of features that make each phone maximally unique.
4. Information from different speakers is incorporated to enhance the triphone model followed by MFCC transformation for each speaker.

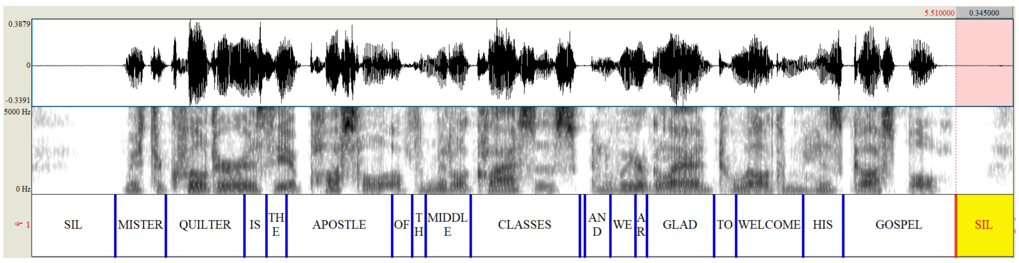


Figure 1: Original waveform is on the top and the segmented forced alignments for word is at the bottom.

Figure 1 shows a TextGrid file for a sample sentence from the 'dev-clean' dataset. The vertical blue lines indicate the alignment of the words in the sentence learned by the algorithm. After this stage, we build a large corpus of segmented waveforms $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(|V|)}\right\}$, where $|V|$ denotes the number of words. All utterances of silence were removed at this stage. The audio segments were converted into the same length, T by zero padding. The resulting segmented waveforms of words were then used to compute the 13 dimensional Mel Frequency Cepstral Coefficients (MFCCs) with a 32ms window and 10ms step size to create a dictionary for our experiments.

3 Methodology

The goal of the experiment is to learn a fixed-length embedding from an audio segment corresponding to a word that is represented in the form of acoustic features like Mel-Frequency Cepstral Coefficients (MFCC). The word utterances can have a variable length sequence and we can also have multiple vectors representing the same word (simply because of variability of speakers). We implement a deep neural network from scratch and train it using two variants

- Skipgram, and
- Continuous bag-of-words (CBOW).

The process is similar to the novel Word2vec approach described in [1].

3.1 RNN Encoder-Decoder

The RNN encoder-decoder framework forms the backbone of the model for our implementation. An Encoder-Decoder network can be constructed with a Recurrent neural network as shown in Figure 2. As it is observed in the figure, a sequence of inputs, $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ is fed into the encoder network sequentially and a corresponding sequence of hidden state, h_t, is obtained. Once the last symbol x^T is processed, the resulting hidden state h^T is considered as the representation of the entire input sequence in the embedding space. This vector can be subsequently used as an input to a decoder network where a sequence of output, $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T'})$ is created sequentially. The process of mapping a sequence to another sequence has been successfully applied to video captioning [18], machine translation [19], and natural language processing [20]. Long Short Term Memory network – 'LSTM' – is a specific type of RNN, that is capable of learning long-term dependencies from the data. For the encoder, we implement a Bidirectional LSTM [12] where it trains two LSTM's, one on the input sequence and the next on the reversed input sequence. This can result in faster learning and can also capture the additional contextual information from the audio data. The decoder is a unidirectional LSTM.

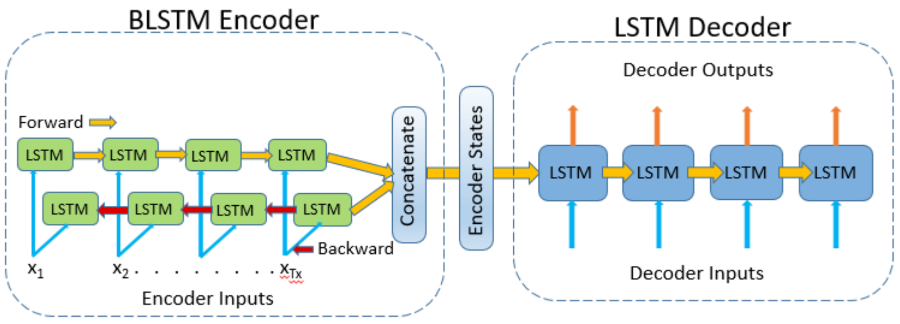


Figure 2: Framework for RNN Encoder- Decoder. Image Credits [21]

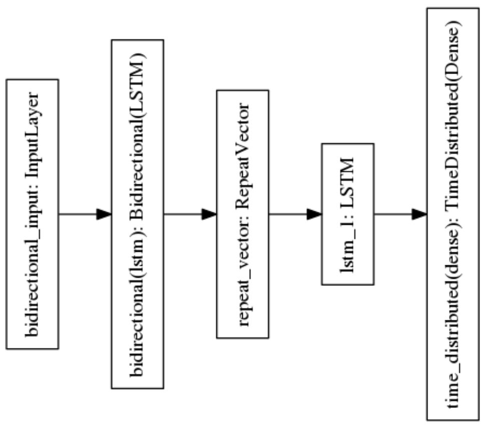

bidirectional_input: InputLayer
bidirectional(lstm): Bidirectional(LSTM)
repeat_vector: RepeatVector
lstm_1: LSTM
time_distributed(dense): TimeDistributed(Dense)

Figure 3: Model Architecture implemented for the Experiment.

3.1.1 Skipgram

As proposed by the authors in Word2vec [1], in the skipgram architecture, the model uses the current word to predict the surrounding contextual words within the specified window size. In the context of audio signals, based on a given audio segment, $x^{(n)}$ from our corpus (which represents utterance of a spoken word), the model is trained to predict the surrounding audio segments $\{\mathbf{x}^{(n-k)}, \ldots, \mathbf{x}^{(n-1)}, \mathbf{x}^{(n+1)}, \ldots, \mathbf{x}^{(n+k)}\}$ where the range **k** is the window size (depicted in green arrows in Figure 4). During the training, the encoder, takes $x^{(n)}$ as input, encodes it into a fixed dimensional vector $z^{(n)}$ and feeds it to the decoder. The decoder subsequently maps $z^{(n)}$ to the output sequences, $\mathbf{y}^{(i)}, i \in \{n-k, \ldots, n-1, n+1, \ldots, n+k\}$ (contextual audio segments). The motivation is to encode the semantic information of the current audio segment, $x^{(n)}$ into a corresponding fixed dimensional vector representation, $z^{(n)}$.

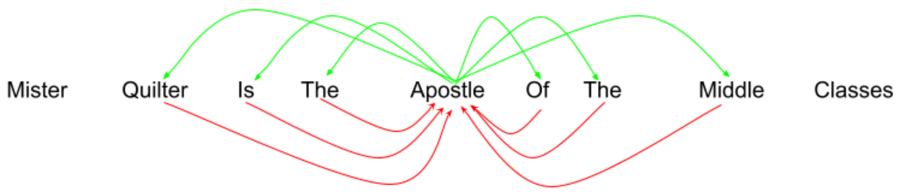


Figure 4: CBOW and skipgram word association. Red arrows are for CBOW (prediction of the center word) and green arrow indicates skipgram (prediction of the context words)

3.1.2 CBOW

Contrary to the previous approach, in continuous bag-of-words architecture, the model predicts the current word from the surrounding context words within a window size. During training, we use the nearby audio segments, $\mathbf{x}^{(i)}, i \in \{n-k,\ldots,n-1,n+1,\ldots,n+k\}$ as the input to the encoder and obtain the encoded embedding, $z^{(n)}$. The encoded embedding is then passed to the decoder to generate the audio segment, $x^{(n)}$ (please see the red arrows in Figure 4). In this process, the post training $z^{(n)}$ is considered as the fixed dimensional vector representation of $x^{(n)}$.

3.2 Experimental Setup

The Encoder is a single-layered bi-directional LSTM, and the Decoder is a single-layered unidirectional LSTM. The model was implemented using Tensorflow on Google Cloud Platform hosting a virtual machine with Linux Debian 9 OS and one Nvidia Tesla T4 GPU. The model is summarized in Figure 2. We performed two experiments for several embedding dimensions using the following methodologies:

- Skipgram,
- CBOW.

The experiments were performed on the same model architecture as seen in Figure 3 with the parameters specified in Table 1.

Parameter	**Value**
Optimizer	ADAM
ETA	0.001
Loss	Mean Square Error
Epochs	20
Batch Size	1000
Activation	Relu

Table 1: Parameters used for training the model

To train our network, with the proposed methodologies (Skipgram and CBOW), we generate pairs of context and target words from the pre-processed dataset described in Section 2.1. Window size of 3 was used to generate 2,94,054 pairs of words from 2709 sentences in dev-clean dataset for the training (as it is described in Speech2Vec [9]).

3.3 Metrics

To evaluate the word-embeddings generated from our models, we use an intrinsic approach. This approach tests for semantic or syntactic relationships between words and includes tasks like word similarity [9], word relatedness, word analogy, etc. Each model was trained with varied size of the hidden layer (the dimensionality of the learned embedding) and as proposed in [9], multiple embeddings of the same word are averaged to get a single word embedding. The vector representations learned by the model are evaluated on three different benchmarks which are widely used to measure the word similarity: WS-353 [13], WS-353-REL [13], and Verb-143 [14]. These datasets contain pair of English words and a similarity rating assigned by humans. For each pair of words in the datasets, we first measure the cosine similarity (formula in Appendix 7.2) using the word embeddings produced by our model. The similarity rankings from the model and the ratings from humans are then used to calculate the Spearman's rank correlation coefficient, ρ (formula in Appendix 7.1). The $\rho = 1$ shows that there is a flawless correlation between the results of our model prediction and the humans rating in the dataset while zero shows no correlation. The results are also compared with two state of the art word embedding methods i.e. Word2Vec [1] and fastText [2] implemented using the Gensim [22] library in python. These models were trained with the default parameter settings but with the same window size (**k**=3) on the text transcriptions of Librispeech dataset.

4 Analysis and results

4.1 Comparison with word2Vec and fastText

The performance of each model on the word similarity benchmarks, summarized in Section 3.3, is presented in Table 2. The proposed models were trained for embedding size of 10 and 30 respectively.

For most of the embedding sizes, the results are dominated by Word2vec for both skipgram and CBOW. However, our model outperforms fastText in 8 cases (highlighted in bold), and Word2vec in 2 cases (highlighted in green). This was quite surprising given the fact that we trained our model for only 20 epochs due to the computational limitations. It is also observed that the CBOW performs better in comparison with skipgram when working with a small vocabulary. This result corroborates with the findings in the original Word2vec paper [1].

	Speech Word2Vec		**Word2Vec**		**FastText**	
Embedding Size	10	30	10	30	10	30
WS-353	0.164	0.135	0.900	0.665	0.595	0.912
Verb-143	**0.146**	**0.141**	0.951	0. 988	0.013	0.002
WS-353-REL	0.108	0.219	0.090	0.593	0.283	0.531

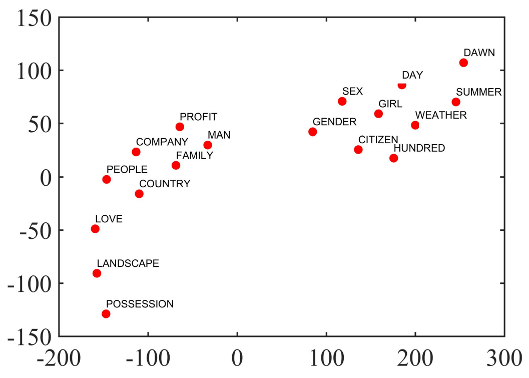
	Speech Word2Vec		**Word2Vec**		**FastText**	
Embedding Size	10	30	10	30	10	30
WS-353	**0.137**	**0.175**	0.593	0.718	0.077	0.036
Verb-143	**0.346**	**0.154**	0.668	0.867	0.049	0.026
WS-353-REL	**0.2411**	**0.217**	0.105	0.226	0.075	0.073

Table 2: The Spearman's rank correlation coefficient ρ between the rankings produced by each model. The bold text indicates the cases where Speech2Vec outperforms the fastText and the green color where it outperforms Word2vec. CBOW (Right), Skipgram(Left).

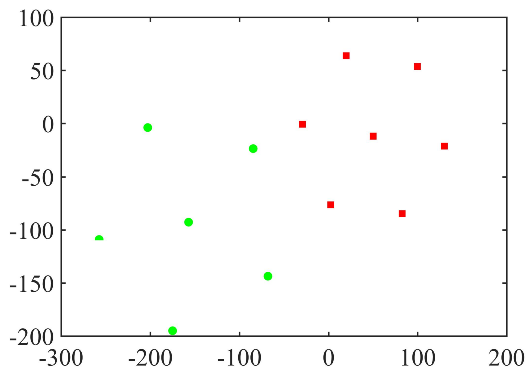
The t-SNE projections for some word embeddings learned by the model can be seen in Figure 5(a). Some interesting conclusions can be drawn as the following:

- words like weather, day, summer, dawn are quite close in the embedding space,
- related words like people, company, country, man, family, profit are clustered together, and
- words closer to love are country and landscape. This could relate to the context they were used in our small vocabulary.

In Figure 5(b), multiple embedding vectors of the same word are shown. It is clearly observed that the learned embeddings for the same word are clustered together in the embedding space.



(a) Random words from WS-353 with embedding size of 30 were chosen.



(b) Multiple utterances of the same word. Red is for 'Cat' and Green is for 'Paper'

Figure 5: The t-SNE projection of the word embeddings learned by CBOW using speech data.

4.2 Effect of embedding size

We trained our models with embedding sizes ranging from 10 up to 100 to better understand the dependency of the size of the embedding layer on the performance of the model. This can help understand the right size that is capable of capturing sufficient semantic information from the audio segments. The results are shown in Table 3.

	Embedding Size				
Word Similarity	10	30	50	70	100
WS-353	0.137	0.175	0.087	0.135	0.240
WS-353-REL	0.241	0.217	0.104	0.148	0.112
RelVerb-143	0.346	0.154	0.214	0.179	0.283

	Embedding Size				
Word Similarity	10	30	50	70	100
WS-353	0.164	0.135	0.161	0.112	0.168
WS-353-REL	0.108	0.219	0.186	0.132	0.185
Verb-143	0.146	0.141	0.098	0.106	0.076

Table 3: The relationship between the embedding size and the performance. CBOW (left), Skip-gram(right).

From the results, it is found that increasing the size of the embedding doesn't always guarantee better performance for either CBOW or skipgram. This can be seen with a dip in performance when going from size 30 to 50 for CBOW and going from size 50 to 70 for skipgram. As mentioned before, to evaluate the performance three different word-similarity datasets are used. Each dataset captures different aspect of words. For example, Verb-143 focuses more on verbs, WS-353 on similarity and WS-353-Rel on the relatedness of commonly used words. A linear combination of the performances using all these word similarity indices can be used as a criteria to identify the optimal size of the embedding depending on the application of the model. In this experiment, a uniformly weighted average is used and according to that the best overall performance is obtained by size 10 (0.241) for CBOW and 30 (0.165) for skipgram.

4.3 Variability for Embedding of same utterances

Due to the variability of speech production, there are multiple utterances of the same word. As proposed in [9], in this project the average of all the embeddings of the same word is used to get one final embedding. To better justify this assumption, the variability of the utterances of the same word is studied. This is done by dividing such words into three different categories depending on their frequencies in the vocabulary. For a word with **N** vector representations $\left\{\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3, ..., \mathbf{w}^\mathbf{N}\right\}$, the mean of the standard deviation is first computed as $m_w = \frac{1}{s}\sum_{i=1}^{s} \text{std}\left(\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3, ..., \mathbf{w}^\mathbf{N}\right)$, where s is the embedding size. Then, it is averaged for every word that belongs to the same sub-group. The sub-groups are depicted in the x-axis of the Figure 6.

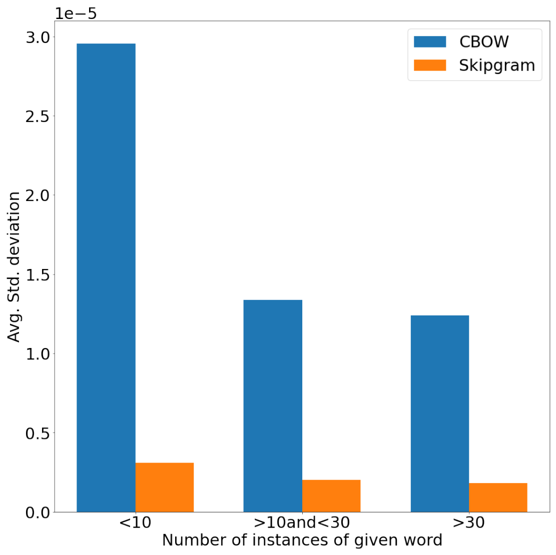


Figure 6: Variance in vector representations for word utterances with respect to the times they appear in the corpus

From Figure 6, it is clearly observed that the variability decreases as the number of utterances, **N** increases. In addition, the variability in CBOW is higher than the variability in skipgram. This could be attributed to the difference in the approaches and methodologies behind the two cases. Nevertheless, the absolute variability in each case is still in the range of 10^{-5}. This low value confirms the claim in [9], that ”the stability of the model increases with increasing the vocabulary size (in turn increasing N)”.

5 Discussion

The proposed architecture to learn embedding from audio borrows the methodology from word2vec by using the co-occurrence information of data. Unlike word2vec, where every instance of the same word is represented by a single embedding vector, in our model every instance of the same word is and can be represented by different embedding vectors. This is because every instance of a spoken word is different. The inherent variability of speech data due to different speakers, multiple occurrences of the same words/sentences, different microphones, etc, adds to our challenge. As proposed in [9], all vectors representing the same spoken word are averaged to get the final word embedding. To ensure the validity of this proposal, we visualize multiple utterances of the same word and discover that they do in fact lie close to each other. A detailed analysis of variability presented in Section 4.3 also justifies the assumption.

Our model achieved low scores in some cases when compared to word2vec and fastText. We believe that the underlying reason is the inherent variability in speech production which sets it apart from textual data. To us, what is impressive is that with a model trained on only a fraction of the complete corpus for 20 epochs, we were able to produce few test scores that not only came close but also outperformed these established methods. Probably, the performance of our model is not directly comparable to the original Speech2vec [9] model as it is trained on the complete 960 hours Librispeech dataset for over 500 epochs.

Using word similarity index to evaluate the quality and performance of embeddings is not the most optimal way and might result in biased or even incorrect inferences. However, the main goal of our project was to show the capability of learning fixed dimensional word embedding directly from speech.

6 Conclusion and Future Work

We show that it is possible to learn contextual acoustic word embeddings directly from speech data by investigating recently proposed literature. We followed ongoing research in the field and compared our implementation with state of art methods like word2vec and fastText. In addition, we applied our models on established word similarity benchmarks. Our results show that contextual word embeddings learned from speech can also be used to for NLP tasks like automatic video captioning, machine translation, etc, despite the additional complexities they carry.

Unlike text, where the corpus can easily be segmented into distinct words, speech has a time dependent continuous form, making the task of getting word boundaries quite challenging. Forced alignment using a transcription to learn these boundaries makes our model semi supervised.

Future work on the unsupervised speech segmentation techniques are a clear continuation of our work. Moreover, convolutional neural networks have proven to be efficient in extracting features for automatic speech recognition and speaker identification. Following that thread, the study can be extended to include features extracted from spectrogram via convolutional neural networks instead of hand engineered MFCC features.

References

[1] Tomas Mikolov, Kai Chen, S. Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

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

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[4] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[5] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*, 2020.

[6] Shruti Palaskar, Vikas Raunak, and Florian Metze. Learned in speech recognition: Contextual acoustic word embeddings. In *International Conference on Acoustics, Speech and Signal (ICASSP)*, pages 6530–6534, 2019.

[7] Shruti Palaskar and Florian Metze. Acoustic-to-word recognition with sequence-to-sequence models. In *Spoken Language Technology Workshop (SLT)*, pages 397–404. IEEE, 2018.

[8] Samy Bengio and Georg Heigold. Word embeddings for speech recognition. In *Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*, 2014.

[9] Yu-An Chung and James Glass. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *arXiv preprint arXiv:1803.08976*, 2018.

[10] Herman Kamper, Weiran Wang, and Karen Livescu. Deep convolutional acoustic word embeddings using word-pair side information. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4950–4954. IEEE, 2016.

[11] David Harwath and James Glass. Deep multimodal semantic embeddings for speech and images. In *Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 237–244. IEEE, 2015.

[12] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 2020.

[13] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. 2009.

[14] Simon Baker, Roi Reichart, and Anna Korhonen. An unsupervised model for instance level subcategorization acquisition. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289, 2014.

[15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[16] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, pages 498–502, 2017.

[17] George Saon, Mukund Padmanabhan, Ramesh Gopinath, and Scott Chen. Maximum likelihood discriminant feature spaces. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2. IEEE, 2000.

[18] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, page 4534–4542, 2015.

[19] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

[20] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[21] M Battula. *Time series forecasting with deep stacked unidirectional and bidirectional LSTMs*. Towardsdatascience, 6 2020. https://towardsdatascience.com/time-series-forecasting-with-deep-stacked-unidirectional-and-bidirectional-lstms-de7c099bd918.

[22] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC Workshop on New Challenges for NLP Frameworks*, pages 45–50. Citeseer, 2010.

7 Appendix

7.1 Spearman's rank correlation coefficient

Spearman's rank correlation coefficient, ρ to evaluate the model.

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n^3 - n}$$

where d_i is the distance between the scores for each pairs and n in the total number of pairs in the benchmark.

7.2 Cosine Similarity

$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

7.3 Peer Review Response

7.3.1 Relevance to the learning outcomes.

- implement methods for training and evaluation of speech recognition systems-This is fulfilled,train and evaluate a speech recognizer, using software tools-This is something you do, however you do not specify what software tools you use more than python. Maybe you could add whatever you use (like Keras or similar) somewhere. Compare different methods for feature extraction and training- You compared your method with others, and on top of that compared different embedding sizes to see what impact it had so I would say this outcome is fulfilled. Document and discuss specific aspects related to recognition of speech and of speakers-This would be your results and discussion. I think your discussion could be elaborated, for example by discussing your work compared to earlier work.
  I think you have done a project that seems relevant to the course, but I think you could add some things which is why I gave you a 5 for this. **5/6**
- The project is very relevant to the learning outcomes. The group used a method for word embedding using speech and compared it to others using text. This project also deals with a specific aspect of speech recognition. In that matter, they measure all learning outcomes of the course. **6/6**
- **Response**:
  Being able to explain the steps we have taken and why is an important factor showing that we have reached the intended learning outcomes of the course.

  You do not specify what software tools you use more than python. Section 3.2 details the parameter specifications, including information about hardware, software, tools and libraries.
  I think your discussion could be elaborated, for example by discussing your work compared to earlier work. In Section 6, we account for our decision to compare our results with methods like FastText and Word2Vec models trained on the same vocabulary and not with the results from [9] which was trained on the entire dataset. This is justified to ensure a fair and comparative evaluation of our work with the current established methods.

7.3.2 Literature study

- The literature seems very complete. They cloud cite a bit more sources in the methodology paragraph. Furthermore, they might want to detail just a little more about more basic concepts, it might allow the reader to go through the literature only for further details. I don't see other literature the group might have missed. Last, try to order the numbers of references. **6/6**
- In the introduction there is hardly any mention of earlier relevant studies, the sources you cite now are more "explaining" sources. The introduction feels a bit poor now, so maybe

you can write more about the techniques you compare with in the result and how that research went. Bring up things you want later to discuss in the discussion. The second part of the report that contains a lot of references is the method, which is great to have a lot of references to support your choices of method.
The actual citing seems to be done correctly, however there seems to be some doubts in your reference list ([15] and [16]) and [12] is just a link which I guess it shouldn't be. Figure 2 is clearly taken from the internet (I googled lstm encoder and found it) so you should definitely add a reference for it. **3/6**

- **Response:**

  They cloud cite a bit more sources in the methodology paragraph.They might want to detail just a little more about more basic concepts.Last, try to order the numbers of references. The methodology section and references in the entire report have been updated in a way that ensures the reader has relevant information about the dependent studies in a proper and structured manner.
  The introduction feels a bit poor now, so maybe you can write more about the techniques you compare with in the result and how that research went. The introduction section has been adapted as recommended. We further elaborate on the prior work/relevant studies and techniques used for comparison for clarity.
  The actual citing seems to be done correctly, however there seems to be some doubts in your reference list... Reference 15 and 16 are now fixed. Reference 12 cites the image mentioned by the reviewer. This is now converted to the same uniform format and added to the image caption.

7.3.3 Novelty / Originality

- Their work and model seems to be inspired by their first reference as they use the same data and model but they bring novelty as they compare it to methods that use text.**4/6**
- The technique you use in your project is based on something already existing, so your experiment does not show any new ground-breaking techniques or methods to do what you do. However, I think what you did using speech instead of words is what is different with your project compared to word2vec and fastText. I could have interpreted it wrong, if so, it's not clear enough what makes your project (or method) unique. I think you should write somewhere (preferably the introduction) very clearly stating how your project contributes to the field and what makes it unique (even though it might be something super-small), in that way it is clear that your work is original and have a purpose.**4/6**
- **Response:**
  The topic for our project is a part of ongoing research with limited resources. As mentioned already, the work is based on the previous work in Speech2Vec [9]. We do take some inspiration from the mentioned paper, but due to the absence of detailed information and any open source implementation, we have had to incorporate modifications in the architecture which makes our model architecture unique. We clearly highlight these adaptations with proper justifications in the report be it the neural network architecture, the comparison with earlier work or the difference in performance of our model with Speech2Vec. Although our work is based on previous work, we realize the importance of the replication and reproducibility in scientific research. Building on previous knowledge is what makes science 'self-correcting'.

7.3.4 Correctness

- I do not see any mistakes on the results or the method of their report.**6/6**
- I can't find any obvious mistake.**6/6**
- **Response:**
  We have tried to be quite descriptive and transparent with our report and choice of methodologies. The correctness of our implementation can also be validated by some of the results presented in Section 4 which are in alignment with the results from previous work in the field.

7.3.5 Clarity of presentation

- The report is globally very clear. The method is well explained and the results are detailed enough. However, I do not think the table of context is very relevant here for an 8 page paper. It will also shorten their paper by one page. Second, I found the Table2 not very clear and simple to understand, the goal is to compare the results between the 3 methods and as it is here, it is not very convenient to look for 2 values from 2 different methods. **5/6**
- I think your report is not easy to follow at the moment. First of all, there are a lot of small mistakes such as some parentheses have too much space between them and some have none, some [X] references lack spaces before they come, etc. It needs thorough proofreading. I also think the language is partly lacking, you use the word ”we” a bit too much for this kind of text. Especially in section 4. If you switch out the “we” and fix all small errors the overall quality will increase a lot.

  Another thing I came across is that your sections are not according to the instructions. For example, you do not have an “Experiment” section where you could talk about your experimental set-up. You say what parameters you use in your RNN but you don’t say how those parameters were chosen. Did you experiment to get them? This is where that should be. You also have Result and Analysis in one, which I find a bit confusing. At the moment you comment the results and come to conclusion already in section 4, reflections on why the results look like it should be in the discussion. Maybe it’s okay to have it in section 4 due to it being named ”Analysis”, however the instructions don’t say anything about analysis so I would change that if I was in your group.**3/6**
- **Response:**

  I do not think the table of context is very relevant here for an 8 page paper. Table of contents has been removed as advised (also now in line with the recommended structure).
  I found the Table2 not very clear and simple to understand. Table 2 is splitted into two sub-tables, one for each method. Some values are highlighted and bolded to increase the simplicity.
  There are a lot of small mistakes such as some parentheses have too much space between them and some have none, some [X] references lack spaces before they come, etc. Corrections in spacing and language is done.
  You do not have an “Experiment” section where you could talk about your experimental set-up: Section 3.2 is the ”Experiment” section with the detailed parameter specification and the model architecture. The reviewer might have missed it.
  You say what parameters you use in your RNN but you don’t say how those parameters were chosen. Did you experiment to get them? The choice of the hyper parameters for the model (shown in Table 1) was the default settings which gave satisfactory results. We did not perform an exhaustive search for these parameters but agree that these can be further tuned to get the optimal performance. This is also addressed in Section 6. We do however perform a qualitative and quantitative evaluation on the size of embedding which is the primary focus of our work. This is covered in detail in Section 4.2.
  I also think the language is partly lacking, you use the word ”we” a bit too much for this kind of text. Especially in section 4. Grammatical mistakes in the report have been fixed throughout the report. The language is updated to be in line with the a technical project report.
  You also have Result and Analysis in one, which I find a bit confusing. At the moment you comment the results and come to conclusion already in section 4, reflections on why the results look like it should be in the discussion. In Results and Analysis, we present the statistical results and our narration of the findings. For clarity and brevity of the analysis, we combine them into one section. The discussion however is an evaluation of how the results relate to the literature and prior work and is therefore covered in a separate section.