Acoustic Word Embedding

Marie SARBIEWSKI
School of Electrical Engineering
and Computer Science
KTH Royal Institute of Technology
Stockholm, Sverige
`sarbi@kth.se`

Florent MONIN
School of Electrical Engineering
and Computer Science
KTH Royal Institute of Technology
Stockholm, Sverige
`florentm@kth.se`

Abstract

The recently introduced Speech2vec model [1] is an efficient method for learning word embeddings from speech, that can be seen as a speech version of Word2Vec[2]. Its design is based on a RNN Encoder-Decoder framework, and borrows the methodology of Skipgrams and Continuous Bag Of Words for training. In this project, we propose an implementation of this model for the Skipgram embeddings, but instead of using the MFCCs we decided to use MSpec. This will allow further reproducing of the results obtained in the paper, and to train on custom dataset if need comes and to further extend the model.

Key words: Speech2vec, Skipgram, RNN Decoder-Encoder

Introduction

Related work

In 2013, Mikolov *et al.* [2] developed a technique to create word embeddings often referred to as Word2Vec. These embeddings have been shown to have really interesting properties. For instance they allow to define an algebra on words. A really well-known property, is the fact that $x_{\text{king}} - x_{\text{man}} + x_{\text{woman}} = x_{\text{queen}}$, with x_{word} the embedding associated with "word" (see [3], [4] and [5]). Other embedding techniques were further developed as extensions or variants of the original Word2Vec, such as the Global Vector (GloVe) embeddings (see [6]) for instance, with Word2Vec being the standard to compare the embeddings to. In the field of Speech Technology, Chung *et al.* started working on equivalent embedding using audio files [7]. This allowed to embed audio of variable size into a vector of fixed dimension. This kind of unsupervised learning lead to vector that reflected the acoustic similarity of the samples. The next step was to train a model to produce embeddings that would take into account semantic proximity [1]. These embeddings share the same interesting properties as Word2Vec, as shown in the original paper, and have been used as a building block for more complex systems (see [8], [9]). Another variant of the Speech2vec embeddings is the Phoneme2Vec embeddings (see [10]), which is just the Speech2vec technique applied on a phoneme level, rather than a word level. Hence, this technique has be shown to be fruitful. It is sensible to expect that one could find interesting variants and extensions of this technique in the same fashion as Word2Vec.

Background

In this paper, we provide an implementation of the Speech2vec embedding model in PyTorch [11], based on the description in the original paper. This model is based on the RNN Encoder-Decoder as described in [12]. The implementation is available at [13].

As mentioned in the previous section, the base idea of Speech2vec is word embedding. The first really successful word embedding technique is Word2Vec. The idea is to learn a fixed-dimension vector for each word. To do this, there are two main models, described below. The general idea is that words that appear in a similar context have similar meanings, so their embeddings should be close. Hence, by going through a lot of text data, the model can learn these embeddings.

Skipgram is one of the two methods presented in [2]. It is an unsupervised learning technique that allows one to predict a context from a word. The context consists in the surrounding words around a word. More precisely, during the training, the context of the word is set as the target and from one word, the model should return context words. No activation function is used in the original model. The final word embedding lies in the hidden layer of the model.

The other embedding method used by [2] is called Continuous Bag Of Words (CBOW). This technique is the converse of Skipgram. Given the context, the model learns to find the word. While both of these techniques use the same idea, they show different efficiencies. CBOW is faster to learn, while Skipgram yields better results in general.

To be able to compare different embeddings, tests have been developed to quantitatively measure the efficiency of an embedding. To do so, benchmarks of word similarity have been created. Humans were asked to rate the similarity between two words, for many pairs of words. This allows to set similarities that the embedding should reflect. A set of 13 benchmarks was set up (see [14]) to evaluate embeddings on words of different types or domains.
The similarity is calculated by computing the cosine similarity between their embedding, and then reporting the Spearman's rank coefficient ρ between the ranking produced by each model against the human ranking. This coefficient indicates whether there is a monotonous correlations between the two rank variables and not the slope of said correlation. Here is its formal expression:

$$\rho = \rho_{\mathrm{rg}_X, \mathrm{rg}_Y} = \frac{\mathrm{cov}\left(\mathrm{rg}_X, \mathrm{rg}_Y\right)}{\sigma_{\mathrm{rg}_X} \sigma_{\mathrm{rg}_Y}}$$

where:

- $\mathrm{rg}_X, \mathrm{rg}_Y$ represents the rank variables,
- $\mathrm{cov}\left(\mathrm{rg}_X, \mathrm{rg}_Y\right)$ is the covariance of the rank variables,
- σ_{rg_X} and σ_{rg_Y} are the standard deviations of the rank variables.

As audio samples are rarely given word by word but more sentence by sentence, for this paper, sentences have to be split in words. To do so, forced alignment is required. Forced alignment is a technique that takes a textual transcription of an audio file and generates a time-aligned version. For instance, the Montreal Forced Aligner [15] can be used. It can align on both words and phonemes, if needed (see its homepage).

A Recurrent Neural Network (RNN) is a class of Neural Network designed to process sequences, and to take into account time dependencies. These are used in the case of audio data, as well as text processing for instance. A specific unit often used as a RNN is the Long Short Term Memory (LSTM) as introduced in [16]. Another unit that was more recently introduced by Cho *et al.* (see [17]) is the Gated Recurrent Unit (GRU). These units allow the model to remember useful information even far away in the sequence, if the information has been learned to be useful.

A RNN Encoder-Decoder is an architecture composed of two elements, the Encoder and the Decoder. The Encoder is a RNN that takes as an input a sequence $\mathbf{x}$ and, for each element in the sequence, sequentially updates a hidden vector. This vector is, at the end of the sequence, an embedding of the original sequence into a vector $\mathbf{z}$ of fixed dimension T.

The Decoder takes this vector as an input, and generates sequentially a whole new sequence $\mathbf{y}$. This new sequence has a size T' which could be different from T.

This allows a RNN Encoder-Decoder, for instance to translate sentences, where the expected output doesn't necessarily have the same size as the input. However, this feature is not used in this particular case, as all words are padded. In the case of Speech2vec, the RNN Encoder is used to generate the embedding of the word, and the Decoder is supposed to find the context from the embedding (for the Skipgram method).

Often, when dealing with audio data, it is needed to reduce the dimensionality of the input. To do this, the Mel Frequencies Cepstrum Coefficients (MFCC) are features that contain useful information on the sample (see [18] and [19]). This set of feature is computed by first applying a Fourier transform to the sample. Then, a bank of triangular filters spaced according to the Mel frequencies is applied, to simulate human-fashion hearing. The log of the power of the result is then taken. The result is treated as a signal, and the MFCCs are the amplitudes of the cosine transform of the result. This allows to obtain coefficients that are uncorrelated, to reduce the dimensionality of the feature vector.

However, when training deep learning models with no assumption that the features are uncorrelated and with sufficient computational resources, the MFCCs are not strictly necessary. Indeed, it is possible to use the Mel Spectrogram (MSpec). The MSpec are computed with the same steps than the MFCCs, without the final cosine transform. The resulting features are correlated and there are many more features than with MFCCs (usual numbers for MFCCs are 13 *vs* 40 for the MSpec. In usual machine learning algorithms, it might therefore be preferable to choose the MFCCs as they are a meaningful representation in a much smaller representation space. However, with deep learning, it is possible to feed the network high-dimensional data without any problem and with good results. Hence we decided to use them instead of the classical approach with the MFCCs.

As defined by Chung *et al.* in [1], Speech2vec is a model to compute word embeddings starting from speech audio, rather than plain text. More precisely, the model is composed of a RNN Encoder-Decoder, as shown in Figure 2, taken from the Speech2vec paper ([1]). The input data is the MFCCs of a word. The encoder (a single-layer bidirectional LSTM) transforms this word into an embedding. The embedding of the whole sequence is the last hidden state vector outputted by the LSTM. Then, the decoder constituted of a single-layer unidirectional LSTM predicts the context. Note that we only describe the Skipgram method here, but the original paper also presents a model with CBOW. However, the authors of Speech2vec showed that Skipgram performed almost always better than CBOW. Hence, we only recreated the Skipgram model. The idea behind this model is that if the embedding contains enough information about the input word, then the decoder should be able to predict information about the context. The embedding that is computed at the end is only the result of the encoder.

Furthermore each different utterance of the same word is treated as a different word, during the training. However, when the final embeddings are computed, they correspond to an average of all the embeddings of the utterances of the same underlying word.

When training the models, we used Stochastic Gradient Descent, as well the Adam optimizer algorithm, as described by Kingma *et al.* in [20]. The Adam optimizer computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Its name is derived from "Adaptive Moment Estimation". As demonstrated in the original paper [20], it achieves good results fast.

1 Method

1.1 Data

To train and test the implemented Speech2vec described in this project, the dataset `LibriSpeech ASR corpus` (available at this address) is used, as in the original paper. This dataset is a collection of 500 hours of broadband speech produced by 1,252 speakers. (see [21] for more precision about the original dataset). Contrarily to other speech datasets such as `LibriVox`, this dataset has been designed and processed to ensure a gender balance at the speaker level and in terms of the amount of data available for each gender. However, due to time and space constraints, we ended up using a reduced part of the dataset. We used a total of 10 speakers, for approximately 4h30 of speech.

1.2 Preprocessing

As the dataset previously mentioned is split by sentences and not by word, the first preproccessing step is to segment all audio files into words. To do so, we needed forced alignment data. We used already computed alignments for the LibriSpeech dataset (see [22]). These were computed using the Montreal Forced aligner (see [15]). Each audio file was then segmented according to word boundaries given by forced alignment with respect to the reference transcriptions such that each audio segment corresponds to a spoken word. This resulted in a dataset of raw audio segments.

However, contrarily to the original paper, we decided not to use the MFCCs, but the MSpec instead. So the MSpec is then computed for every word, providing a set of 2-D tensors, with one 2-D tensor for each word. These tensors represent the MSpec for each time frame of the words.

Then, to facilitate the learning for the model, we normalized the data over the whole dataset for each feature.

However, all tensors do not have the same length (as every word does not have the same length). To be able to compute 3-D tensors for a batch, we need to pad each word with zeros, so that the model can take a whole batch at once, rather than one word at a time.

The dataset was then split in a training set and a validation set. As we won't be using a traditional evaluation method for our model, but rather the benchmarks described earlier, we won't need a testing dataset. The whole dataset was split with 10% in the validation set, and 90% in the training set.

The preprocessing is summarized in the Figure 1.

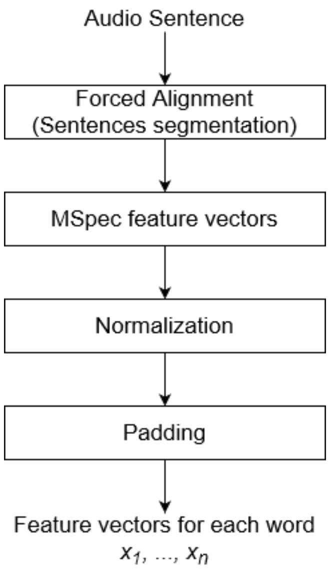


Figure 1: Preprocessing steps

1.3 Model architecture

The architecture used in this project replicates the one used in the paper of Chung *et al.* [1] and is summarized in the Figure 2. The model first implements the encoder RNN as a single-layered bidirectional LSTM and the decoder RNN is another single-layered unidirectional LSTM. It is fed the MSpec vectors for each word and outputs the context. The selected loss that we used is the **Mean-Square Error** (MSE), with the following expression:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

where:

- n is the total number of samples in the training (or validation) set,
- $(Y_i)_{i\in\{1,\cdots,n\}}$ are the real targets,
- $(\hat{Y}_i)_{i\in\{1,\cdots,n\}}$ are the targets computed by the model.

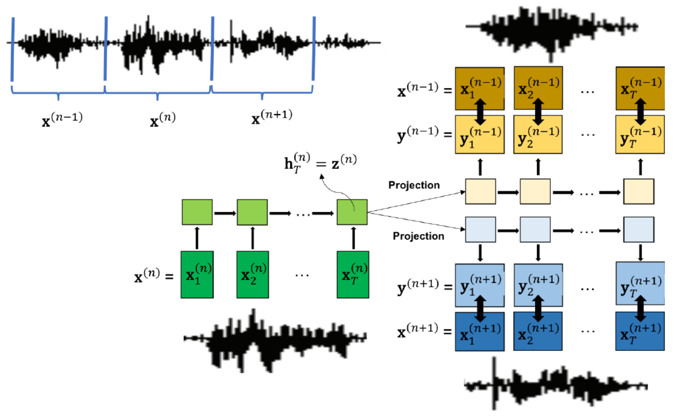


Figure 2: Scheme of the model as taken from the Speech2vec paper ([1])

2 Experiments

2.1 Training Configuration

We implemented the model Speech2vec with Pytorch [11]. We also define also the windowing length k that corresponds to the size of the right context and the left context (so the total context size is $2 \cdot k$). As mentioned in the paper, we set the window size to $k = 3$ (so a total context size of 6). We first trained the model with a stochastic Gradient Descent (SGD) without momentum, with a fixed learning rate of 5×10^{-2} and 200 epochs. We then trained another model with the Adam optimizer (see [20]), with a learning rate of 10^{-1} and 100 epochs.

The authors of [1] described a precise set of parameters. They searched amongst many possible hyperparameters, including the depth of the RNN Encoder and Decoder, the choice of the RNN cells (LSTM or GRU). As the parameters above provided the most stable and satisfying results, they were kept for the final implementation in the original Speech2vec implementation.We kept most of these parameters in our implementation. However, due to time restrictions, we reduced the number of epochs, and therefore increased the learning rate. We also tried two different models, one trained with Stochastic Gradient Descent, one with the Adam optimizer.

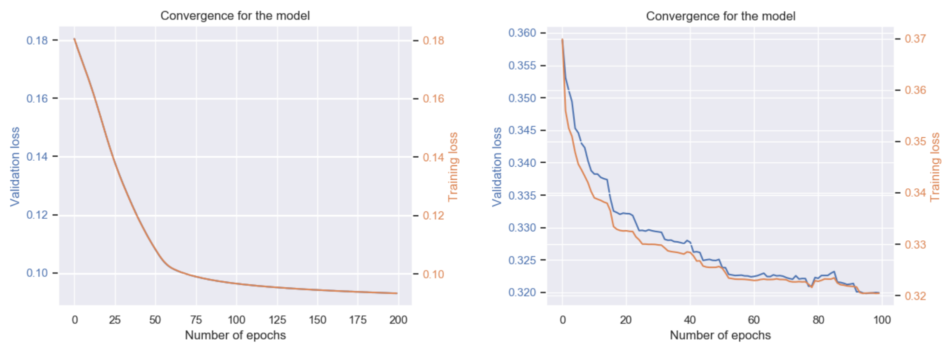
2.2 Performance measurement

In the original paper, the authors used 13 benchmarks to measure word similarity, including **WS-353** [23], **WS-353-REL** [24], **WS-353-SIM**, **MC-30** [25], **RG-65** [26], **Rare-Word** [27], **MEN** [28], **MTurk-287** [29], **MTurk-771** [30], **YP-130** [23], **SimLex-999** [31], **Verb-143** [32], and **SimVerb-3500** [33]. They consist in different pairs of English words that have been assigned similarity ratings by humans, and each of them evaluates the word embeddings in terms of different aspects. These benchmarks have been carefully to test as many different aspects of word similarity as possible.

3 Results

3.1 Model Training

To measure the convergence of the two models, we plotted the the training and validation losses. Note that on Figure 3a, the training and validation losses are confounded.



(a) Model trained with SGD (b) Model trained with Adam optimizer

Figure 3: Training and validation losses

3.2 Embeddings evaluation

Using the 13 benchmark as described in the previous sections, we obtained results with the two optimization methods we used:

- Results with the Adam Optimizer are reported in the table 1,
- Results with the Stochastic Gradient Descent Optimizer are reported in the table 2.

Dataset	Number of pairs	Not found	ρ
MC-30	30	24	0.7247
MEN	3000	2278	0.0281
MTurk-287	287	249	0.0318
MTurk-771	771	593	0.0193
RG-65	65	51	0.0945
Rare-Word	2034	1963	0.1814
SimLex-999	999	606	0.0084
SimVerb-3500	3500	2389	0.0532
Verb-143	144	63	0.3077
WS-353	353	286	0.0389
WS-353-REL	252	202	0.0977
WS-353-SIM	203	163	0.0951
YP-130	130	102	0.1178

Table 1: Results on the 13 benchmarks for the embeddings generated by the model trained with SGD

4 Discussions and Conclusions

4.1 Training

Figure 3 shows that the model were not clearly overfitting, comforting the idea that more training would have help the model to achieve better results. However, the losses did not decreased by a lot, and one might wonder about the significance of this loss if it did not improve much. It should be noted that the loss is the MSE between the prediction of the network and the target context. However, the purpose of the model is to construct embeddings. The generative power of the decoder is not the main study of this paper, hence the loss might not be the best tool to measure how well the model learned. It still helps in some way to monitor the convergence however.

While the Figure 3b shows a convergence as we might expect it from the model, the Figure 3a is quite surprising. It shows a smooth training curb matching exactly the validation curve. This could mean that the validation set contains utterances of words that are also contained in the training set.

Dataset	Number of pairs	Not found	ρ
MC-30	30	24	0.2353
MEN	3000	2278	0.4628
MTurk-287	287	249	0.0138
MTurk-771	771	593	0.0026
RG-65	65	51	0.1122
Rare-Word	2034	1963	0.2200
SimLex-999	999	606	0.1092
SimVerb-3500	3500	2389	0.1335
Verb-143	144	63	0.1784
WS-353	353	286	0.1672
WS-353-REL	252	202	0.2164
WS-353-SIM	203	163	0.2168
YP-130	130	102	0.1760

Table 2: Results on the 13 benchmarks for the embeddings generated by the model trained with Adam

As the dataset is quite small, there could be some similarities in the validation and training dataset speaker-wise as well. It could also be that this model achieved better generalizability. However as we'll see in the section 4.2 this is probably not the case.

The figures display also different scales on the y-axis. This is probably because of the high-variance in the splitting of the dataset, and the random initialization.

4.2 Results

The results on the 13 benchmarks obtained with MSpec have to be compared with the reference results obtained by the original implementation of Speech2vec with MFCCs and to Word2Vec (see table 3). As expected, given the fact that the dataset we used is quite small, many words were not found amongst the required pairs in each benchmarks.

With the Stochastic Gradient Descent optimizer, we can see good results on the MC-30 dataset compared to the original implementation of Speech2vec in [1] and shown in table 3. This benchmark focuses essentially on nouns, and has a very low number of word pairs. The results must be quite good "by chance": indeed, if all the found words are well classified, as there is not a lot of found words, this leads to a good result.

Dataset	Speech2vec	Word2Vec
MC-30	0.846	0.762
MEN	0.619	0.642
MTurk-287	0.468	0.504
MTurk-771	0.521	0.499
RG-65	0.790	0.752
Rare-Word	0.323	0.408
SimLex-999	0.292	0.280
SimVerb-3500	0.157	0.149
Verb-143	0.315	0.378
WS-353	0.508	0.452
WS-353-REL	0.346	0.308
WS-353-SIM	0.663	0.602
YP-130	0.321	0.391

Table 3: Results for the embeddings in the Speech2vec paper, compared to Word2Vec embeddings

On bigger benchmarks though, results are really bad compared to the original results, except for the Verb-143 benchmark. The verb distribution must be less spread out than the noun distributions, which means that it is more likely that one noun appears only once in a text than a verb, which could explain the results on the verb datasets.
Overall, the Stochastic Gradient Descent did not perform well in this case. This could be explained by the fact that we multiplied the learning rate by 50, due to the restricted number of epochs we could use. This probably prevented the model from finding a good local minimum in the loss function. It could also be the case that the chosen loss expression is not perfectly suitable to the task, and that there might be a better loss to use in our case (with so few iterations).

As the Adam optimizer uses an adaptive learning rate, its results are generally better than the Stochastic Gradient Descent results. However, as mentioned before, we suspect the loss not to be meaningful enough, and our model not to be trained enough, so that our results do not reach the original paper's results.
Some results are nonetheless quite close to the originals one, for instance in the case of the SimLex-999 or SimVerb-3500. Without more training, it is difficult to further analyse the results.

4.3 Conclusion

In order to obtain better results, more training time, with a smaller learning rate would have been necessary. But the most important factor in creating good embeddings would be to train on a much bigger dataset. We were limited by the time, and the disk access time of the computer on which we trained. Hence, to be able to come up with a satisfying model, one should probably change the way the data is loaded, or train with a more powerful computer.

Furthermore, the lack of resources and training time prevented from assessing the efficiency of the MSpec features compared to the MFCCs. However, with the few results that we got, we suspect that with similar training, the MSpec would perform better than the MFCCs, as they essentially contain more information.

4.4 Further discussion

In this paper, we only considered the Skipgram implementation of the Speech2vec. However, it could be possible to adapt our code to implement CBOW instead, but Chung *et al.* ([1]) showed that the Skipgram embeddings almost always outperformed the CBOW embeddings.

This work could be extended by changing the features that are fed the RNN Encoder-Decoder. We already changed the features from MFCCs to MSpec, hoping that they would contain more information than the MFCCs, and would help the model to learn better the embeddings. However, one could imagine more suitable features for this project. Indeed, the MSpec contain a lot of phonetic information that make them suitable for HMM training, but they might not be fully useful in this case. Features containing more contextual information could perhaps yield better results.

In Speech2vec, once the model is trained, only the encoder part of the RNN is kept. However, it could be interesting to study the generating properties of the Decoder part. Indeed, such a network could be used to generate a sequence of words, in the same fashion as Word2Vec. Though, this would require a way to transform MSpec to a waveform (see the WaveRNN architecture [34]).

It could also be interesting to try and train this model on a dataset with specific vocabulary, to see how specialized the model can become, for embedding as well as for generation for instance.

Here, the dataset had to be forcefully aligned. However, it might be interesting to design a model predicting the word boundaries given the MSpec, and to use such a model to predict when the word sample ends, and where the generation of the context should stop, for instance.

References

[1] Yu-An Chung and James R. Glass. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *CoRR*, abs/1803.08976, 2018.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality, 2013.

[3] Edgar Altszyler, Sidarta Ribeiro, Mariano Sigman, and Diego Fernández Slezak. The interpretation of dream meaning: Resolving ambiguity using latent semantic analysis in a small corpus of text. *Consciousness and Cognition*, 56:178–187, Nov 2017.

[4] J. Lilleberg, Y. Zhu, and Y. Zhang. Support vector machines and word2vec for text classification with semantic features, 2015.

[5] Yiben Yang, Lawrence Birnbaum, Ji-Ping Wang, and Doug Downey. Extracting commonsense properties from embeddings with limited human guidance, 2018.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[7] Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder, 2016.

[8] Eda Okur, Shachi H Kumar, Saurav Sahay, and Lama Nachman. Towards multimodal understanding of passenger-vehicle interactions in autonomous vehicles: Intent/slot recognition utilizing audio-visual data, 2019.

[9] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. Towards unsupervised speech-to-text translation, 2018.

[10] Albert Haque, Michelle Guo, Prateek Verma, and Li Fei-Fei. Audio-linguistic embeddings for spoken sentences, 2019.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[12] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[13] Marie Sarbiewski and Florent Monin. DT2119 - final project, 2020. Accessible at this address.

[14] Manaal Faruqui and Chris Dyer. Community evaluation and exchange of word vectors at wordvectors.org.

[15] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: trainable text-speech alignment using kaldi. *Proceedings of the 18th Conference of the International Speech Communication Association*, 2017. Accessible at this adress.

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

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[18] A. Waibel and K.F. Lee. *Readings in Speech Recognition*. Elsevier Science, 1990.

[19] Lindasalwa Muda, Mumtaj Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques, 2010.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

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

[22] CorentinJ. Librispeech alignments, March 2019. availale at this address.

[23] Dongqiang Yang and David Powers. Word similarity on the taxonomy of wordnet, 01 2006.

[24] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Strakova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. *Proceedings of NAACL-HLT*, pages 19–27, 01 2009.

[25] George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[26] Herbert Rubenstein and John Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8:627–633, 10 1965.

[27] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[28] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, page 136–145, USA, 2012. Association for Computational Linguistics.

[29] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, pages 337–346, 01 2011.

[30] Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 1406–1414, New York, NY, USA, 2012. Association for Computing Machinery.

[31] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456, 2014.

[32] Simon Baker, Roi Reichart, and Anna Korhonen. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289, Doha, Qatar, October 2014. Association for Computational Linguistics.

[33] Daniela Gerz, Ivan Vulic, Felix Hill, Roi Reichart, and Anna Korhonen. Simverb-3500: A large-scale evaluation set of verb similarity. *CoRR*, abs/1608.00869, 2016.

[34] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2017.

A Corrections from the report draft

Remarks	Corrections
This project is mostly the reproduction of a previous study, which provides the description of model, the theoretical support and the detailed implementation, and does not perform further exploration.	To bring some novelty to the project, we focused more on the use of **MSpec**, contrarily to the original paper where they use the MFCCs. We expect to bring some novelty with our use of different optimizers too.
Problem with citation [9].	This problem has been corrected in the citation 11 now.
Many concepts or terms are not well-explained, such as Specch2Vec, Skip2Vec (which I believe is a typo) and CBOW.	These concepts have been explained in this version of the paper and the corresponding literature has been referred to too.
Further clarification of models and terms is needed.	We clarified the model further in the section 1.3.
Necessary evaluation is missing in this report.	We evaluated our model in the section 4.2 and explained our evaluation process in the section 2.2.
Replication is sometimes good for science, but novelty and necessary alteration of the original work is also highly needed.	To bring some novelty to the project, we focused more on the use of **MSpec**, contrarily to the original paper where they use theMFCCs. We expect to bring some novelty with our use of different optimizers too.
Further description and explanation of the figures will be appreciated.	We described our figures a little further (for instance, our model is more described in the section 1.3).
More research, and lifting up more relevant work which is related to your report.	This has been done in the introduction.
You could get more background that strengthens your work.	We did this in the introduction.
It seems a bit unclear what the authors have contributed with.	Our specific contributions (using the MSpec instead of the MFCCs and using different optimizer in our model) have been specified in the abstract and in the model section.
The method could also be done more in-depth.	We described our method further in the section 1.3.
The results are clearly not done yet.	We filled the result section with our results from the two models.
The reader should not be forced to enter those links in order to find out what the interesting properties were of those embeddings.	We added the example property of defining an algebra on word to show the interest of the embeddings.
When you take images from other papers, as you do in Figure 2, then you should state the source.	We added the source of the figure in the caption of the Figure 2 (from [1]).