Speech2Vec: A Comparison of Recurrent Neural Network Memory Cell Variants for Sequence-to-Sequence Learning of Word Embeddings from Noisy Speech

Group 06
Zakaria Nassreddine, Elin Saks, Ilias Talidi & Marcus Jacobsson

Abstract

In this paper, we reimplement the work of Chung and Glass [1] that introduced a deep neural network architecture to learn acoustic word embeddings from speech. This work extends Word2Vec, introduced by Mikolov et al. [2], to audio segments rather than the underlying spoken words. The model architecture being based on a Recurrent Neural Network Encoder-Decoder framework, we conducted a comparative study on two memory cell variants: Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU), trained on a corpus of a smaller size and consisting of noisier data. The authors' results favour LSTMs over GRUs on a variety of evaluation metrics when trained on a large corpus of clean data. We produced new experimental results on the two variants' noise robustness, when the corpus is of significantly smaller size and poorer quality, to serve as reference model selection guidelines for Spoken Language Understanding practitioners covering different use cases of corpus size and quality. Our new results suggest that GRUs are a very solid candidate in this use case of data scarcity and offer comparable performance at a much lower computational cost.

DT2119 Speech & Speaker Recognition
KTH Royal Institute of Technology
July 27, 2022

I. Introduction

Representation Learning is one of the pillars on which some of the recent success in Natural Language Processing has relied, as a lot of the downstream tasks in this discipline benefited greatly from compact and semantically rich pre-trained feature vectors for words. These vector representations, also known as embeddings, are built in an unsupervised way that exploits co-occurence information in text, building on the intuition that a word's meaning can be inferred from its context, provided that a large and representative enough corpus is seen by the system. Based on this premise, several architectures have been proposed, like Word2Vec [2] and GloVe [3], but they only take text data as training corpora. When only this kind of systems is available, training word embeddings on one's speech dataset requires an intermediate step of speech-to-text processing by an ASR system to build the corresponding textual data, which both induces potential ASR errors and strips the speech from all the extra information that it can carry beyond text, conveyed by speech phenomena such as prosody. As remedy to both of these issues, Chung and Glass [1] proposed an architecture to learn word embeddings directly from speech, built on an RNN Encoder-Decoder framework that can be trained in similar fashion to Word2Vec [2] using skip-gram or CBoW training methodologies, see section III.

The original paper's authors retained LSTMs as the RNN variant to use as building blocks for the framework, as they yielded better performance than GRUs. However, it is important to note that the used data set consisted of 500 hours of the clean version of the LibriSpeech corpus [3], and such data may not always be of abundance in domain specific settings, both with regard to size and quality. When building Minimum Viable Products in industrial settings or experimenting on specific data for which extensive, high quality collection hasn't been performed or cannot be afforded, one has to deal with corpora of much smaller size and poorer quality, in which case the question of appropriate model selection given the available data arises. In this work, we aim to compare both memory cell variants (LSTM vs. GRU) on our 25 hour corpus of noisy data, and compare our experimental results to the ones presented in [1]. This is motivated by results in literature that suggest that GRUs can outperform LSTMs when the data is scarce. Our aim is that this comparison may serve as guidelines to support the choice of RNN memory cell depending on the data at hand.

II. Background

A. Memory cells

Recurrent Neural Networks are a type of Artificial Neural Networks that are used to to deal with sequential data of arbitrary length. They have the particularity of using information from previous output combined with the current input at a given time to produce the new output. But when standard RNNs are exposed to long sequences, they tend to forget information from earlier patterns. Under the hood, this a result of the problem known as vanishing gradients [4] during error backpropagation through time, where the gradients of the loss function approach zero with the growing length of the sequences, making it harder to train. To overcome this problem, different approaches have been proposed, and we will only be presenting two of these that we will use as building blocks for our Encoder-Decoder architecture. LSTMs were introduced by Hochreiter and Schmidhuber [5] in 1997 and quickly gained popularity for using memory cells to regulate which information to keep or discard while parsing sequences. These cells are made out of 3 gates: the Input gate decides what information to store in long term memory based on the current input and the short term memory from the previous step, the Forget gate decides which information from long term memory to keep by multiplying the incoming long term memory with a forget vector that is generated by the current input and short term memory, and lastly the Output gate generates the new short term memory to pass on to the next state based on the current input, the previous short term memory and the newly computed long term memory. On the other hand, Gated Recurrent Unit [6] networks more recently introduced a two-gate based approach to control the flow of information, with a Reset gate that is responsible for the short term memory of the network and an Update gate that is responsible for Long Term memory. The flow is such that a candidate hidden state is generated from the previous hidden state and the reset gate output, before the Update gate is called to generate the final hidden state. More details can be found on the cited papers. These memory cells allow RNNs to capture long term dependencies that they would otherwise forget.

B. Related work

Previous work has shown that deep neural networks can be successfully used in several natural language processing (NLP) tasks, such as paraphrase detection [7], language modeling [8] and, the focus of this paper, word embedding extraction (Word2Vec) [9, 10].

As per Chung and Glass [1], the first step to learn word embeddings from speech is, conventionally, to use an automatic speech recognition (ASR) system to transcribe speech into text. Traditionally, speech systems have used many hand-fixed processing stages, for instance with Hidden Markow Models (HMMs) [11]. However, Hannun et al. [11], and further explored by Amodei et al. [12], introduced an ASR system called "Deep Speech" using recurrent neural networks (RNNs), that was shown to have better accuracy and noise handling than traditional systems. The second step is to apply textual word embedding methods on the results from ASR transcripts [1]. Several research efforts have explored this, notably the work from Mikolov et al. [13] and the introduction of the Skip-gram model; a method for learning vector representations of words, or word embeddings, from unstructured text data. The model did not involve any dense matrix multiplications, unlike other neural network architectures, which made the training very efficient. In addition, the paper introduced the similar continuous bag-of-words (CBow) model. Later, the Skip-gram model was further developed and improved by sub-sampling frequent words [9]. The basic architecture of Skip-grams is a single-layer that is based on the inner product of

two word vectors. Similarly, a vector log-bilinear model, vLBL and ivLBL, was introduced in [14]. However, as noted by Pennington, Socher, and Manning [3] , these models fail to utilize the statistics of the corpus as they don't train on global co-occurrence counts. In light of this, GloVe was introduced to combine the advantages of local context and the global statistics of the corpus [3].

Learning distributed representations from speech by first transcribing and then training on text using Word2vec [9] or GloVe [3] has demonstrated good results. However, these techniques have two evident weaknesses. First, when converting speech signals into plain text, useful and rich information in speech is lost, which could have been used by the model to learn improved semantic representations. Second, these techniques are limited by the precision of ASR systems [1]. Alternatively, researchers have been exploring the idea of learning distributed vectors representations directly from speech [15, 16]. Extending on Word2Vec [9], Chung et al. [17] introduced a model for unsupervised learning called Audio Word2Vec based on two RNNs equipped with LSTM cells, which were proven to outperform prior state-of-the-art implementations. It was later expanded from word-level to utterance-level, to represent an utterance as a sequence of vectors with phonetic structure information [18]. Using two RNNs was further explored by Settle and Livescu [15] in a supervised setting, comparing different RNN-based structures with LSTM networks. The so called "Siamese RNN" embedding proved to outperform previous results on word discrimination tasks. Furthermore, Chen, Parada, and Sainath [19] also used RNNs with LSTM cells for word classification in a query-by-example task. These above techniques, however, are designed based on acoustic-phonetic similarities rather than semantic, meaning that different occurrences of the same underlying word map to different areas in the embedding space. Conversely, Chung and Glass [1] propose a novel deep neural network architecture, Speech2Vec, that focuses on the adjacent acoustic regions of a spoken word rather than the acoustic region associated with the word itself. The model, inspired by Word2Vec [9] integrates an RNN Encoder-Decoder framework with either skip-grams or CBoWs. It was proven to outperform Word2Vec trained on ASR transcripts. In addition, Speech2vec has been further explored in speech emotion recognition [20] and punctuation prediction [21], yielding good results.

In RNN Encoder-Decoder architectures, LSTM networks were proven to be effective [5, 1, 15]. Alternatively, GRUs have been recognized to be a viable alternative to LSTMs with usually faster training [22]. Ravanelli et al. [23] show that a GRU architecture reduced training time while achieving similar results. In addition, Settle and Livescu [15] showed that GRU networks outperformed LSTMs in a deep RNN model with a limited number of stacked layers, however, by increasing this number of stacked layers, the LSTMs outperformed GRUs. In the Speech2Vec model [1], LSTMs had better performance than GRUs.

In addition to different memory cells, the attention mechanism used by Chung and Glass [1] in their implementation of Speec2Vec, and that we also implement in this project, has also had a significant positive impact on RNN Encoder-Decoder model performance in literature. Adapted early by Bahdanau, Cho, and Bengio [24], these mechanisms help the decoder decide which parts of a sequence to pay attention to (explained as align), in order to alleviate the workload for the encoder by not having to encode all information in the sequence (explained as translate) [24]. Their proposed model could outperform previous implementations of Encoder-Decoder models. Furthermore, attention mechanisms have been applied in several NLP tasks like entailment [25], question answering [26] and semantic parsing [27] with good results.

C. Dataset

To train our models, we decided to make use of the LibriSpeech ASR corpus [28], and in particular the *train-other-500* dataset, which is a set of 500 hours of "other" speech, in other words, noisy or dirty speech data. We considered that a good compromise between our computational resources and the aim of achieving statistically significant results would be a twentieth of the full data set. We decided to obtain a subset of 25 hours of speech by performing random subsampling in order to ensure that the distribution of our subset is representative of the original dataset.

To obtain the right word segments, we used the LibriSpeech Alignments provided by [29] which come in a simple, condensed format (a .txt file for each utterance) that mirrors the corpus' folder structure where all words are mapped from the ground truth. Each line starts with the utterance id, followed by the ground truth words and finally the end time for each word. This was key for splitting every word into audio sequences.

These alignments were computed using the Montreal Forced Aligner algorithm, which is is a technique to take an orthographic transcription of an audio file and generate a time-aligned version using a pronunciation dictionary to look up phones for words [30]. Its main purpose is to serve as a resource to assist linguistics research by providing an easy way to align English voice recordings with scripts and/or word lists [31].

The Montreal algorithm consists of four steps of training [32]:

- 1st step: Using monophone models, where each phone is modelled the same regardless of phonological context.
- 2nd step: Using triphone models, where context on either side of a phone is taken into account for acoustic models.
- 3rd step: Performing LDA+MLLT to learn a transform of the features that makes each phone's features maximally different.
- 4th step: Enhancing the triphone model by taking into account speaker differences, and calculating a transformation of the MFCC features for each speaker.

The subset follows the following structure (for each folder):

- Audio files *(.flac)*
- Transcript *(.trans.txt)*
- Alignment *(.align.txt)*

The transcript indicates the name of the directory where the file *.flac* (audio data) is stored along with the transcript

of that audio.

In terms of preprocessing, we then extracted Mel Frequency Cepstral Coefficients (MFCC) and built the MFCC vectors for each audio segment. To this end, we used *librosa*, a Python package for audio analysis, to produce MFCC features with 13 coefficients using a sampling rate of 16000.

Furthermore, we also built two dictionaries to map back and forth between words (in their audio form, also referred to as audio segments) and their corresponding MFCC feature vectors so that we can both train on real valued vectors and be able to retrieve a word from its MFCC vector at inference time.

III. Methods

A. Implementation

We remind that the goal of this system is to learn embeddings, i.e compact vector representations of a fixed dimension, for speech segments that map to words. These words are represented by sequences of MFCC features that vary in length. The desired characteristic of these embeddings is to be able to capture rich semantic information of the underlying spoken words.

1) Model Architecture: The model architecture is inspired by Word2Vec, except that it replaces the fully connected neural layers with a Recurrent Neural Network Encoder-Decoder framework, this is done to account for arbitrary length audio segments. The encoder (with a randomly initialised hidden state) reads an input sequence and, for each entry in the sequence, updates its hidden state that is used as part of the input to process the following entry, to end up at the last hidden state that encapsulates information about every entry in the sequence. The decoder then takes the last hidden state of the encoder as its initial hidden state, and performs similar processing to generate an output sequence that may or may not be of the same length as the input. After training, the encoder's final hidden state is taken as the acoustic word embedding, analogous to the hidden layer's weights in a classic Word2Vec architecture. All embeddings corresponding to the same underlying word are averaged, the scope and effects of this averaging operation are discussed in section 3 of [1].

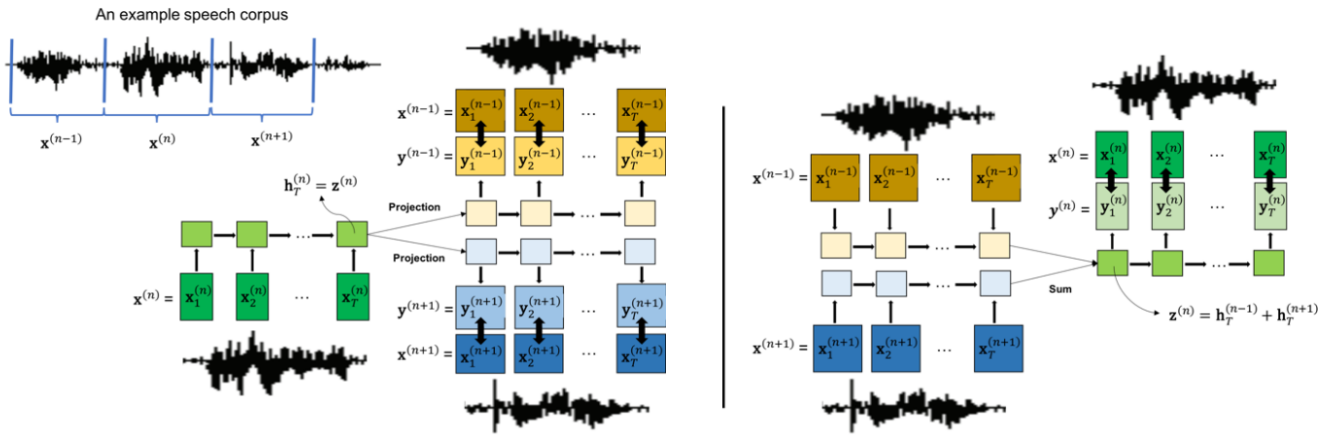
2) Training: The Word2vec inspiration mostly lies in the training methodologies. Two variants can be used , see figure 1 taken from [1], to varying degrees of success with respect to evaluation metrics that will be further discussed in section III-B. Training with skipgrams aims to predict the context (neighbouring segments) from the focus segment (given as input to the model). That is, by sliding a window over the corpus, for each segment at index i, predict all segments at index i-l, for l non-zero ranging from -k to k, where k is a hyperparameter that decides the size of the sliding window. The model is trained by minimising the gap between the output sequences and the ground truth neighbouring segments. On the other hand, training with CBoW (Continuous Bag of Words) aims to predict the focus segment from its context. By sliding a window over the corpus, the model encodes all segments within a k range from the current (focus) segment, computes an aggregate of the respective encodings (using a sum or a mean), which is then fed to the decoder that has to generate the focus segment.

3) Implementation and training details: We implemented our own model using PyTorch. It essentially consists of three building blocks: an Encoder, implemented as a either a single-layer bidirectional Long Short Term Memory (LSTM) network or a single-layer Gated Recurrent Unit network (GRU), and a Decoder which, matching the Encoder, is either a single-layer unidirectional LTSM or a single-layer unidirectional GRU, equipped with an additional component that implements an Attention mechanism [33], which allows the Decoder to use, on top of the output from its previous hidden state, the last hidden state of the encoder to generate every output sequence. The encoder was chosen to be bidirectional to allow it to learn representations of the words that encapsulate both past and future context, which is achieved by a forward pass that uses previous tokens, followed by a backward pass which inverts the order of tokens in a sequences and allows the model to learn from "future context". Since the decoder's goal is only to produce outputs using what the encoder has learnt, it doesn't make sense for it to be bidirectional which is why it was chosen to be unidirectional. The window size was set to three, and the model was trained using stochastic gradient descent without momentum and a fixed learning rate of $1e$-3 for 50 epochs. The two variants of the model were trained on the same data and under the same hyper-parameter settings. The orignal paper's results suggest that an increase in the embedding space dimension does not necessarily yield a worthwhile boost in the performance. Considering this and in order to stick to our computational cost budget, we decided to restrict our experiments to embedding dimensions of 50 and 100, which yielded some of the best results in [1]. To be able to run several experiments, we trained using a virtual machine instance with an Nvidia Tesla T4 GPU provided by Google Cloud Platform using the Compute Engine API with the credits that were allocated to us from the course coordinators. Training took roughly 2.5 hours for GRU based models and about 5.5 hours for their LSTM counterparts, adding up to about 16 hours of total training time (GRU-50d, GRU-100d, LSTM-50d and LSTM-100d). The training losses are depicted in figures 2 and 3, where we observe that the classifiers maintain quite high error rates and do not seem to overfit on the training data, which did not warrant to incorporate any form of regularisation on our end.

B. Evaluation

13 benchmarks were used to evaluate the performance of the model. Faruqui and Dyer [34] developed the tool we used, which tests semantic relationships between words. The 13 benchmarks let us measure how accurately the embeddings our models produce convey the meanings of words. Each benchmark contains multiple pairs of English words and a similarity rating, which has been assessed by humans.

The tool calculates the similarity between two words by taking their word embeddings and computing the cosine similarity. It then compares this value with the human ranking by



(a) Speech2Vec with skipgrams (b) Speech2Vec with cbow

Figure 1: Model Architecture

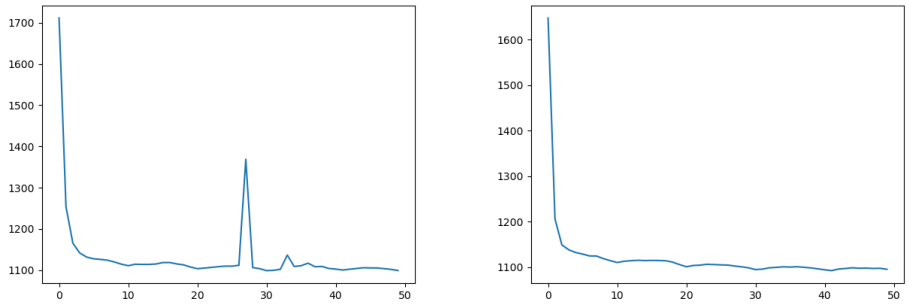


Figure 2: Training losses w.r.t epochs for the GRU-based models with embedding dimension of 50 (left) and 100 (right).

calculating the Spearman's rank correlation coefficient ρ [35], which is used when the data is ordinal but not necessarily on a numerical scale. ρ takes values from -1 to 1 where a value of 0 means no correlation.[36] If the correlation is high, the ranking given by the humans is similar to the one calculated from the embeddings. In other words, a high correlation is interpreted as the embedding being accurate enough to capture good semantic relationships, according to a human authority. This is a good metric as it allows us to quantify and upscale the way we would confront a computed word similarity to prior knowledge about the language in order to estimate if the embeddings are any good.

IV. Results and Discussion

Table I delineates the results from the evaluation using Spearman's rank correlation coefficient on the corresponding models. The best performance for each model is marked in **bold**. The best performance *between our implementations* for each benchmark is marked in red.
The first column is the benchmark, the second and third are the ρ-values achieved for the original Speech2Vec model (using Skipgrams and LSTM) with embedding sizes of 50 and 100, respectively. Column 4 and 5 show the results for our implementation of the model using GRU memory cells for embedding sizes of 50 and 100. The last two columns show the result for our implementation of the LSTM model with embedding sizes of 50 and 100, respectively.

For our models using 50-dimensioanl embeddings, the correlation between the human similarity ranking and the embeddings' similarity ranking is low, sometimes negative. The models with 100 dimensional embeddings perform better.

The number of benchmarks each of our own models performs best on is rather evenly distributed between the first 3. The GRU model with 100 dimensional embeddings performs best on the highest number of benchmarks, 5.

A. Discussion

Derived from the results illustrated in table I, we can conclude that with an embedding dimension of 50, the LSTM model performed better than the GRU model, with an average score of 0.0273 compared to an average score of 0.010. In addition, using the same dimension, the LSTM model outperformed the GRU model at seven out of 13 benchmarks. These results are in line with the previous findings in [1],

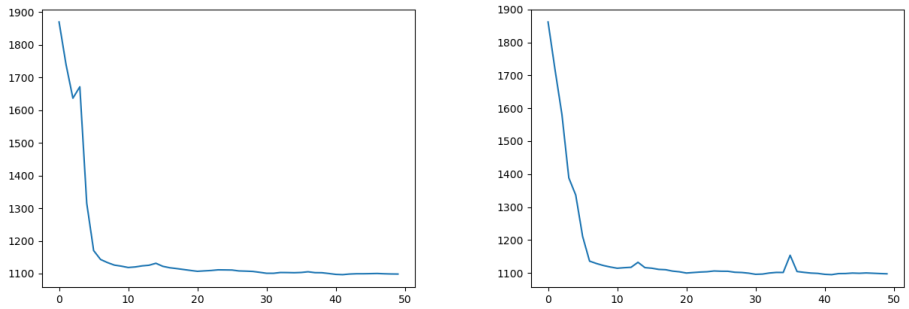


Figure 3: Training losses w.r.t epochs for the LSTM-based models with embedding dimension of 50 (left) and 100 (right).

Benchmark	Original 50d	Original 100d	GRU 50d	GRU 100d	LSTM 50d	LSTM 100d
MEN-TR-3k	0,619	0,606	-0,024	**0,341**	-0,004	**0,3281**
MC-30	**0,846**	**0,815**	-0,129	0,136	0,041	0,1255
MTurk-771	0,521	0,503	0,055	0,028	0,058	0,0258
SIMLEX-999	0,292	0,317	-0,026	0,063	-0,046	0,056
VERB-143	0,315	0,276	**0,136**	0,064	**0,123**	0,037
YP-130	0,321	0,334	-0,086	0,050	-0,242	0,049
RW-STANFORD	0,323	0,321	0,058	0,044	0,027	0,0472
RG-65	0,790	0,756	-0,045	0,280	0,115	0,244
WS-353-ALL	0,508	0,502	0,086	0,020	0,117	0
WS-353-SIM	0,663	0,653	0,036	0,025	0,083	0,0323
WS-353-REL	0,346	0,332	0,046	-0,050	0,119	-0,051
MTurk-287	0,468	0,442	0,031	-0,157	-0,007	-0,1578
SimVerb-3500	0,157	0,183	-0,007	0,090	-0,028	0,0932

Table I: ρ-value for all benchmarks for each model.

but do not match our expectations influenced by literature [37] suggesting that GRUs can perform better than LSTMs when the data set is smaller. However, when the dimension is increased to 100, the GRU model outperforms the LSTM model, with an average score of 0.072 and 0.064 respectively.

The GRU model saw a large improvement in performance when increasing the embedding dimension. A dimension of 50 had an average score of 0.010, and a dimension of 100 had an average score of 0.071. Similarly, when increasing embedding dimension for the LSTM model from 50 to 100, the average score went up from 0.027 to 0.064. This is the opposite of how the dimensions compared in the original paper, where the performance was better for embeddings of size 50.

Our implementation performs poorly compared to the original implementation in [1]. However, they also conducted experiments on a smaller data set of 50 hours using an embedding size of 50. When using a small data set they found the performance of the embeddings to be poor, and the performance increased with the size of the data set. Since our data set is half the size of the smallest one they tested, the subpar performance is to be expected.

The performance of our models makes the results unreliable. This means that our results are very limited to this specific case and the conclusions drawn from them cannot be extended further.

B. Conclusions and Future Work

We reimplemented Speech2Vec, an RNN Encoder-Decoder framework that we trained with Skipgrams to learn acoustic word embeddings, i.e fixed-length word vectors learnt directly from speech. We experimented with two memory cell variants: LSTM and GRU, to obtain empirical results on model performance in the context of scarce and noisy audio data. We showed through our experiments that GRUs ought to be considered in these specific settings and can yield results that are as good as what their greedier and more complex counterparts can provide, at a significantly lower cost. However, our findings can only be accepted and presented as viable model selection guidelines if we prove their significance with more extensive statistical hypothesis testing, which we consider to be the next step in order to confirm or reject any of these findings. Our models can also be improved by the incorporation of negative sampling, introduced in [2], to speed up the training process. Another area for improvement would be to experiment with Transformer architectures, inspired by similar work in NLP where dropping RNNs to only rely on attention has yielded promising results [38].

References

[1] Yu-An Chung and James R. Glass. “Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech”. In: *CoRR* abs/1803.08976 (2018). arXiv: 1803.08976. URL: http://arxiv.org/abs/1803.08976.

[2] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[4] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[6] Junyoung Chung et al. "Gated Feedback Recurrent Neural Networks". In: *CoRR* abs/1502.02367 (2015). arXiv: 1502.02367. URL: http://arxiv.org/abs/1502.02367.

[7] Richard Socher et al. "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection". In: *Advances in neural information processing systems* 24 (2011).

[8] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model". In: *Advances in Neural Information Processing Systems* 13 (2000).

[9] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013a).

[10] Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[11] Awni Hannun et al. "Deep speech: Scaling up end-to-end speech recognition". In: *arXiv preprint arXiv:1412.5567* (2014).

[12] Dario Amodei et al. "Deep speech 2: End-to-end speech recognition in english and mandarin". In: *International conference on machine learning*. PMLR. 2016, pp. 173–182.

[13] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013b).

[14] Andriy Mnih and Koray Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation". In: *Advances in neural information processing systems* 26 (2013).

[15] Shane Settle and Karen Livescu. "Discriminative acoustic word embeddings: Tecurrent neural network-based approaches". In: *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2016, pp. 503–510.

[16] Keith Levin et al. "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings". In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 410–415.

[17] Yu-An Chung et al. "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder". In: *arXiv preprint arXiv:1603.00982* (2016).

[18] Yu-Hsuan Wang, Hung-yi Lee, and Lin-shan Lee. "Segmental audio word2vec: Representing utterances as sequences of vectors with applications in spoken term detection". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 6269–6273.

[19] Guoguo Chen, Carolina Parada, and Tara N Sainath. "Query-by-example keyword spotting using long short-term memory networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 5236–5240.

[20] Panagiotis Tzirakis et al. "Speech emotion recognition using semantic information". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6279–6283.

[21] Jiangyan Yi and Jianhua Tao. "Self-attention based model for punctuation prediction using word and speech embeddings". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7270–7274.

[22] Apeksha Shewalkar. "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU". In: *Journal of Artificial Intelligence and Soft Computing Research* 9.4 (2019), pp. 235–245.

[23] Mirco Ravanelli et al. "Light gated recurrent units for speech recognition". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.2 (2018), pp. 92–102.

[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[25] Alexis Conneau et al. "Supervised learning of universal sentence representations from natural language inference data". In: *arXiv preprint arXiv:1705.02364* (2017).

[26] Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. "Question answering through transfer learning from large fine-grained supervision data". In: *arXiv preprint arXiv:1702.02171* (2017).

[27] Li Dong and Mirella Lapata. "Language to logical form with neural attention". In: *arXiv preprint arXiv:1601.01280* (2016).

[28] Open Speech and Language Resources. "Librispeech: An ASR corpus based on public domain audio books". In: URL: https://www.openslr.org/12.

[29] "LibriSpeech Alignments". In: URL: https://github.com/CorentinJ/librispeech-alignments.

[30] "Montreal Forced Aligner - User Guide". In: URL: https://montreal-forced-aligner.readthedocs.io/en/latest/user_guide/index.html#user-guide.

[31] "Forced Aligners". In: URL: https://web.uwm.edu/forced-aligner/.

[32] "Montreal Forced Aligner - Pipeline of training". In: URL: https://montreal-forced-aligner.readthedocs.io/en/latest/user_guide/index.html#user-guide.

[33] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[34] Manaal Faruqui and Chris Dyer. "Community Evaluation and Exchange of Word Vectors at wordvectors.org". In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 19–24. DOI: 10.3115/v1/P14-5004. URL: https://aclanthology.org/P14-5004.

[35] Jerome L Myers, Arnold D Well, and Robert F Lorch Jr. *Research design and statistical analysis*. Routledge, 2013.

[36] Gunnar Blom et al. "Korrelation". In: *Sannolikhetsteori och statistikteori med tillämpningar*. 5th ed. Studentlitteratur, 2005, pp. 232–235.

[37] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. *A comparison of LSTM and GRU networks for learning symbolic sequences*. 2021. DOI: 10.48550/ARXIV.2107.02248. URL: https://arxiv.org/abs/2107.02248.

[38] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

APPENDIX

Our report has evolved from the state it was in when we submitted the draft to include our experimental results as well as edits in response to the peer-reviews we received that we note below.

1) ***A suggestion of improvement would be to explain why the Spearman rank correlation coefficient is a good comparison criterion***: We explained the motivation behind using Spearman's rank correlation coefficient and added a reference to forward the reader to more details in the evaluation section.
2) ***It might also make sense to add a conclusion section in the report to dwell upon the obtained results and present your point of view***: We added the results, discussion and conclusion sections as they were missing in the draft.
3) ***One small thing to add is maybe a reference to the "Attention is all you need" paper, since attention is used in the model***: We added a paragraph under "related work" to highlight previous work related to the attention mechanism used in the original Speech2Vec implementation. Note, however, that the reference mentioned in the peer-review isn't inherently relevant to attention as a concept because it did not propose the mechanism but, rather, built upon it to propose an architecture that drops the RNNs and only keeps attention in the encoder-decoder framework. We have added another reference to attention but also included this suggestion as possible future work to build embeddings using transformer architectures (and thus only needing attention).
4) ***A small suggestion would be to add an image with the different components of the network***: We have added a figure displaying the structure of the network to support the textual explanation in the methods section.
5) ***Results and graphs around the evolution of training and testing accuracies of the model could be added***: We added graphs showing the evolution of the different models' training losses under different settings and used the observations to justify our decision not to perform regularisation. We did not run our classifiers on test data because we are not interested in classification per se, but rather the latent embedding space that we can generate through training the classifiers.
6) ***It would be nice to add some references and explanations for concepts like 'Spearman's rank correlation coefficient' and 'Vanishing Gradient Problem' mentioned in the report***: We explained the problem of "Vanishing gradients" in the context of Recurrent Neural Networks and added a citation for more details. A similar suggestion regarding Spearman's rank correlation was addressed perviously.
7) ***How did you obtain the subset of 50hrs in the data? It should be ensured that the distribution of this subset is a good representation of the entire subset with 500hrs of data***: We explained how we obtained the training subset and how we made sure it was representative of the original data set. We initially intended to train on 50 hours but ran out of RAM and had to downscale to 25.
8) ***The details on how the mfccs are generated are missing. The sampling rate, window length etc. should be mentioned***: We added experimental details about how we performed MFCC extraction in section II.C.
9) ***Why was the encoder chosen to be bi-directional while the decoder was chosen to be unidirectional? A motivation for this choice could be added***: We explained the motivations for this choice in section III. 3.