An Analysis Tool for Spoken Language in VR

Gal Gilbert
Efi Arazi School of Computer Science,
IDC - Interdisciplinary Center, Herzliya

Submitted: March 2^{nd}, 2019

1 Introduction

The audio record analysis tool's goal is to generate an automated prosody and textual analysis for a given data recorded at any experiment (at the AVL – Advanced Virtuality Lab in the IDC). The idea of creating such a tool came after a year of working on a thesis aimed to connect speech to body-movement of an avatar (virtual character). A short brief of this effort is described at Appendix A.
The main design concepts of this tool are simplicity of use and robustness, in the sense of fitting the tool not only to a certain experiment at the AVL. The tool was not designed for a specific task, meaning that its output would serve as raw data that should be post-processed by the user for his defined purposes.
The document contain a description of the tool, a necessary background on the main modules within it and provide a general performance analysis of the tool.

2 Audio Record Analysis Tool Description

This tool accepts two inputs - audio file and *lab streaming layer* (LSL) output file[1] (XDF format, with timestamps insicating when the participant started and stopped speaking) - and produce the outputs described in this section:

[1]LSL - a data streaming system that provides synced recorded data from experiments

2.1 Audio splitter

Given an audio file (*wav* format) recorded at a certain experiment and a matching XDF file, this module split the audio file into separate time-labeled sentence/utterance audio files (in *wav* as well). The main concept of this module is roughly dividing the input audio file into sub-files according to the utterances timestamps in the XDF file. At the next phase, the module iterates over each sub-file and performs an optimization. With the detection of silent segments at each file the module decides whether the file represents a valid utterance (the non-silent segment is long enough) or if it should be split again (meaning splitting an utterance into two separated sub-utterances).
In principle, this module could have worked solely on the audio file. Implementation of such feature requires that the record starting time would be part of the audio file name and that the record quality would be good enough (in terms of microphone's quality and low background noise level). For example, if loudspeakers are used as part of the experiment, this module might not be able to distinguish between the loudspeakers and the participant's speech.
The Audio splitting module can be adjusted using the following parameters:

1. Minimal speech threshold (dB).
2. Minimal silent time required between utterances.
3. Minimal length of the first audio segment/.

2.2 Speech-to-text converter

This module converts each of the aforementioned audio sub-files into a written text using Google's Cloud speech-to-text API (https://cloud.google.com/speech-to-text/). This module has no actual parameter to tune, but since this service is not free (when using it more than a certain times per month) an API-key is required.

2.3 Sentence embedding

$Doc2Vec$ model is a shallow neural network trained to generate text embeddings. This model would be described extensively in Section (3), so this section is not meant for covering the technical aspect of it. The embedding module use a pre-trained $Doc2Vec$ model to infer each of the utterances generated by the speech-to-text module. In practice, given a certain text as input, the module's output is a numeric 300-dim vector representation of the text (the dimension size is one of the training's parameters that can be

changed).
This module offers two mechanisms:

1. Sentence inference - given an unseen sentence (at the training phase) as input, inferring it with $Doc2Vec$ (meaning generating an embedded numeric representation) is equivalent to training the model with this sentence as input. This process is demonstrated later (Section 5.3.2). The user can control the number of iterations, the initial learning rate and the minimal learning rate that are in use for the inference step.

2. Model training - the user can train a new $Doc2Vec$ model. All that is required from him is providing a path to a dataset (txt file containing sentences, or paragraphs, separated by lines). The user can control the model's training parameters, if desired (more on these parameters at Appendix C).

2.4 Prosody analysis

The technical aspect of this module is covered at Section (4). This module relies on an open-source tool based on two articles - [12] and [6]. By getting each utterance audio file as input, a dedicated prosody analysis is generated using statistic tools and methods applied on the amplitude, fundamental frequency and energy of the input audio file. The following features are generated as part of the prosody analysis:

1. 38 numeric **static features** - statistical data that characterize the input file (such as maximal energy, talk pause rate, etc.).

2. Additional 13 so-called **"dynamic" features** describing these data (file duration and two types of Legendre polynomial coefficients).

3. **Data plots** of the aforementioned features describing the energy, $F0$ (fundamental frequency), amplitude and dynamic features. The base-data used for these plots is saved to a dedicated csv file.

The user can define which algorithm type he wants to use for the fundamental frequency computation, as well as whether to generate and/or store plots for each utterance.

2.5 Requirements

In order to use the *Analysis Tool for Spoken Language*, the experiments this tool would analyze should implement the following:

1. Audio record of the entire experiment (the spoken part).

2. Utterance recognition - identifying when the participant(s) starts to talk when the talking stops.

3. Implementation of LSL in the Unity project. This system should generate a XDF file with event marks of the audio record start, audio record end and each of the recognized utterances start and end times (these event marks should be at a certain text format).

Further details can be seen under the user's guide at Appendix D.

3 Text Embedding

3.1 Background

The main aspect of this project, both in the literature research phase and the chosen model adoption, was finding a method that measures similarity between sentences. This research resulted in preferring approaches that represent words as vectors such that shared-meaning words can be grouped together. Bengio et al. [3], taking the statistical language modeling approach, suggested a trained model to reduce the dimensionality of possible word sequences in a certain language by learning a distributed representation for words. This is done in order to deal with unseen word sequences. Mikolov et al. [10] showed two novel models, generally named $Word2Vec$, for representing words from a very large unstructured data as continuous vectors. Apparently, these neural-network models are very efficient to train, and syntactic and semantic word similarities can be detected from the generated vectors. This model was improved [11], and later on served as the basis for Mikolov et al. [8] work on representing documents and sentences. This method, called $Doc2Vec$, is similar to $Word2Vec$ with the addition of document sets as input, such that during the training phase there are vector representations of documents as well ("document" might refer to a paragraph or a sentence). Eventually, the chosen model for this project was Doc2Vec [8].

3.2 Doc2Vec

In order to get a better understanding of the chosen model, its ancestor - named $Word2Vec$ [10] - should be explained. In brief, the goal of the $Word2Vec$ model is to create a distributed representation of words that would capture semantics and syntactic information within it. This model,

using a trained shallow neural-network containing one hidden-layer only, generates a *N-dim* numeric representation for each word that the model receive as input. The training itself takes a corpus of sentences as input. It has two different training algorithms, but its concept, maximize conditional probability, remains the same in both of them: For each input sentence (represented as *N-dim* random vector initially), the model tries to maximize the conditional probability of a given word to appear given its neighboring words (a method named continuous bag of words, or $CBOW$ in short), or vice versa - maximize the conditional probability of neighboring words to appear given a center word (named *skip-gram*).
In order to demonstrate the training phase, the $CBOW$ model will be described. Initially, each word is assigned with one-hot vector at the size $1 \times V$ (V is the size of the corpus). The model has one hidden layer, so two weight matrices are needed. The first weight matrix W, at the size of $V \times n$ (n is the desired output dimension), projects the input of the model to the hidden layer using linear transformation. The second weight matrix W' connects the hidden layer to the output layer. Formally, row k in weight matrix W and column k in W' represent the k'th word in the corpus. Therefore, V_k (row k vector from W) is named input vector, where V'_k (from W') is named as output vector.
The training is done by iterating word by word and set its surrounding words, named context window (with size C which is user-defined), as the input. The iterated word is called **center word**, and the input that represent the context window, denoted W_k, is generated by averaging the multiplication of these C surrounding words' one-hot vectors with W. Then, the objective of the training model is to maximize the conditional probability of $P(W_t|W_k)$, where W_t represents a possible center word. This conditional probability is implemented using hierarchical softmax or negative sampling as an activation function applied on the output layer, and it depends in practice in the dot-product between the averaged input vectors and the output vector mentioned before.
The original center word serves as ground truth (i.e., we want the model to assign the highest probability to this word to appear in the given context window comparing to all the other words). The learning process is done by backpropagation and stochastic gradient descent.
Doc2Vec works quite similarly. It has two training methodologies as well - *PV-DBOW*, which is *skip-gram* like, and Distributed Memory Model of Paragraph Vectors (*PV-DM*), which is $CBOW$ like.
There are two main differences $Doc2Vec$ has:

1. Using paragraphs as its input (or documents, meaning the model can

be fed with more than one sentence at a time).

2. Generation of additional paragraph vectors that are trained as part of the training as well. $Doc2Vec$ model is still predicting words, with one main addition: taking an input document, $Doc2Vec$ objective is maximizing the conditional probability of words within the document to appear in this document (thus, training a document vector as some sort of a word vector in a different vector space). In this way, the trained model can infer (generate) an embedded representation of any given paragraph.

While *PV-DBOW* try to predict the words of a certain paragraph given a paragraph vector as input, *PV-DM* inherently train word-vectors as part of its training process.
Finally, it turns out that paragraphs (words) that are more related to one another by meaning have $Doc2Vec$ ($Word2Vec$) "closer" embedded representations (measured with cosine similarity).

3.3 Model Training

The main intuition I had when approaching this stage was that there is a need for a dataset containing short sentences and utterances. Since my initial work involved short utterances spoken throughout conversation (described at Appendix A), such a dataset is required in order for the trained model to fit AVL experiment's characteristics. Models that were trained using the familiar datasets, which contain long texts such as Wikipedia values, movie reviews or news articles, did not provide satisfying results when I tried to infer short utterances with them and compare the resulted similarity.
Therefore, I have combined seven datasets of what I thought to be short enough texts to form my training dataset. The datasets are:

1. Subset of the British National Corpus (http://www.natcorp.ox.ac.uk/) that is transcribed unscripted spoken dialogue. This dataset was extracted and preprocessed as part of this project (https://github.com/Phylliida/Dialogue-Datasets).

2. Cornell Movie–Dialogs Corpus [5]

3. DailyDialog: A Manually Labeled Multi-turn Dialogue Dataset [9]

4. Stanford's NLP Group - a New Multi-Turn, Multi-Domain, Task-Oriented Dialogue Dataset [7]

5. TickTock and IRIS from WOCHAT (Workshops and Session Series on Chatbots and Conversational Agents) (http://workshop.colips.org/wochat/data/index.html)

6. CIC dataset, based on original data from the human evaluation round of The Conversational Intelligence Challenge (CIC). Various Chatbots are used in the dialogues (https://dbd-challenge.github.io/dbdc3/datasets.html).

7. OpenSubtitles2016 - a collection of translated movie subtitles from http://www.opensubtitles.org/. From this huge stockpile I used the English part of the English-Spanish movies translations.

Prior to training, the aforementioned datasets went through some preprocessing, included removal of (some of the) duplications, lower-case capital letters, punctuation mark removal, digits to text conversion, converting most of the shortcuts into standard format (for example, $that's$ to $that\ is$), removal of invalid and single characters and slang and correction of some of the spelling mistakes.
In total, by using datasets 1-6 I managed to create a dataset containing almost $547K$ sentences. To this amount I have added part of the open-subtitles dataset with some variations. After generating few dataset versions, with a varying number of sentences taken from "OpenSubtitles2106", the final data set size I used was $3,410K$ sentences.
I trained dozens of models throughout the last year using different parameters and dataset variations. The chosen model, trained with the dataset at the aforementioned size, proved to succeed more than the others in a short inference test I conducted (found better similarity between context-matching sentences than the other models and vice versa). A demonstration of the model results on the recorded experiments' data is shown at section 5.3, and the model's general performance is demonstrated at Appendix C.

4 Prosody Analysis

According to Wikipedia, "in linguistics, prosody is concerned with those elements of speech that are not individual phonetic segments (vowels and consonants) but are properties of syllables and larger units of speech" (https://en.wikipedia.org/wiki/Prosody_(linguistics)). Since this subject is out of

the scope of the original thesis, it will be described briefly.
As stated before, the prosody analysis implemented in this project mostly relies on three main properties:

1. **Fundamental Frequency (F0)** – the basic definition of $F0$ is the lowest frequency of a periodic wave. At the speech domain, $F0$ is often referred as the pitch of an audio signal – how high or low is the sound. $F0$ can also be described as the frequency at which our vocal cords are operating at a given time. Since this frequency varies throughout an utterance, its computation over time is done by dividing the audio signal into small segments and extracting the value of $F0$ from each. There are several algorithms that estimates $F0$. In this specific tool two of them are in use: The default one is based on the intended $PRAAT$ software [4] - a computer program that analyze, synthesize, and manipulate speech data and present the output in high quality. The other algorithm named $RAPT$[2] (robust algorithm for pitch tracking), and is widely used within speech applications.

2. **Amplitude** - can be defined as the difference between the zero level and peak of the audio signal.

3. **Energy** - [1] defines the signal's energy as the reflection of the signal's amplitude variation, or how much signal there is at a given time. Energy is measured with dB and is calculated on a short period basis (small windows of time) by the following equation:

$$Energy\ (dB) = \frac{1}{N}\sum_{n=1}^{N} V(n) \tag{1}$$

   where N is the number of samples at a time window and $V(n)$ is the signals value at time n. Usually there is a correlation between voiced segments at an audio signal (high energy, when a person speaks) to unvoiced segments (low energy, when the person is silent).

5 Tool Evaluation

5.1 Audio Splitter Evaluation

The initial stage at the tool's pipeline does a sufficient job. Tested on few experiments' audio records from the AVL (medium quality microphone and participants with varying level of English speech and accent), the *Audio Splitter*

managed to extract more than 90% of the spoken sentences (113/125 sentences from 6 participants), while giving only 3 false-positives (background noises that were recognized as speech).

5.2 Speech-to-Text Evaluation

The speech to text conversion feature is crucially dependent on the quality of audio record, as well as on the quality of speech. An attempt to convert speech of participants with bad level of English, or with a significant accent, is likely to result in texts without a meaning. Nevertheless, good audio records of allegedly native speakers can also result, sometimes, in an undesirable results due to local low volume/noisy segments or the use of unique words (such as names of unfamiliar places, names of people, etc.). A speech-to-text conversion demonstration, of relatively good record, is shown at Appendix B.

5.3 Sentence Embedding Evaluation

Originally, $Doc2Vec$ model did not aim to deal with spoken utterances. The main goal of $Doc2Vec$ model was finding similarity between paragraphs for classification tasks such as sentiment analysis. Accordingly, some of the common large datasets available on-line are good for this purpose. Although the $Analysis\ Tool\ for\ Spoken\ Language$ does not meant to support a certain use, I find it impossible to evaluate its performance without defining some task that it can be tested on. Therefore, and despite $Doc2Vec$'s original intention, I set the model's goal to find similarity between pairs of utterances in such a way that responses to a certain question (at an experiment) would be more similar than responses to other questions.
After training dozens of models using different variations of few datasets (existing datasets and ones that was created by me), I can state that my goal is yet to be achieved. However, taking into account the fact that the recorded data used for the model evaluation was not optimized (straightforward transcription or speech2text conversion), as well as the unrestricted answers domain (participants were not guided what and how to respond), than we did receive some meaningful results.
The following figure demonstrates the prediction (inference) of the chosen $Doc2Vec$ model on **some** (not all) of the data. For generating this figure I mapped responses from five different participants to the same five questions (not all the participants answered all the questions) and used PCA (the first two components) to represent part of these answers.

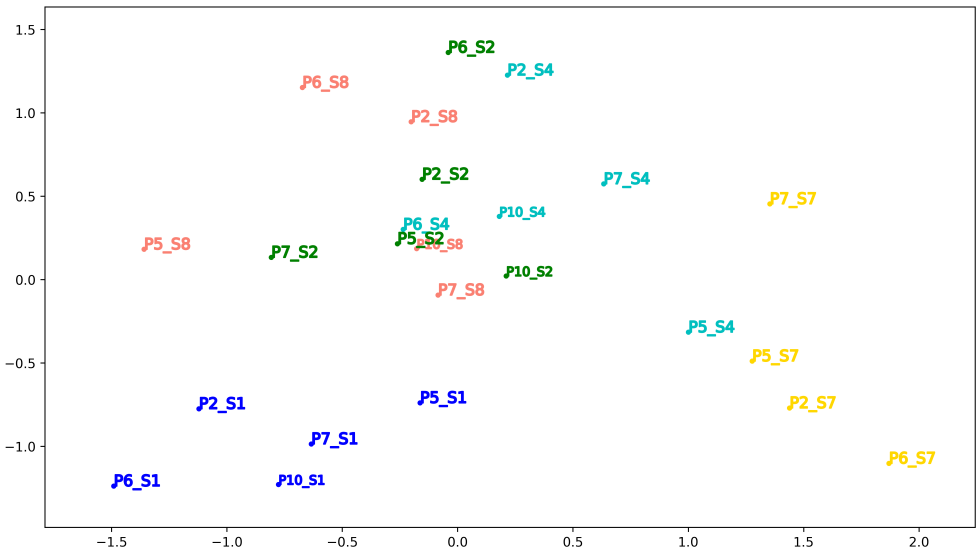

Figure 1: PCA representation of embedded sentences ($\mathcal{R}^{300} \rightarrow \mathcal{R}^2$ dimension reduction) where each color represents an answer to a certain question. For example, "$P2\_S1$" is answer of participant 2 to question #1

As seen at the figure, we can group some of the answers to the same questions next to each other using a reduced representation of their embeddings.
The questions I used were the following:

1. $S1$ – "Hello, thanks for coming. First, tell us a little bit about yourself".
2. $S2$ – "Alright, I get it. So, why should we hire you?"
3. $S4$ – "Describe the best boss you ever had."
4. $S7$ – "You know this job would require some mathematical skills. You have 30 seconds, I want you to please count the prime numbers, starting from 1 and getting as far as you can in 30 seconds."
5. $S8$ – "How od you like to work – Alone? Part of a team?"

To evaluate the embedding, I chose KMeans algorithm. I ran it on the original embeddings ($\mathcal{R}^{300}$) as input, and set the number of desired clusters to five (which is the number of different questions that I chose to embed their responses). The figure below represent the results. Each color represent a different cluster, and the use in PCA ($\mathcal{R}^{300} \rightarrow \mathcal{R}^2$) was done for display purposes only (running KMeans on the PCA result (with dimension $\mathcal{R}^{10}$)

did not show better results). The interesting fact is that we got purity values varied between 0.7 to 0.83 for this clustering, while clustering a random data generated purity values at the range 0.42-0.58.

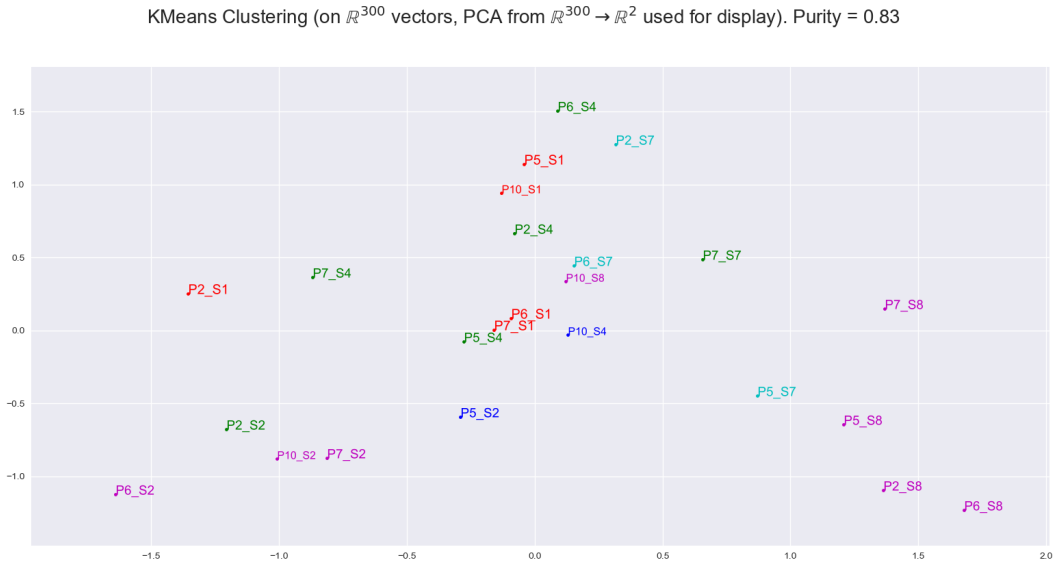

Figure 2: KMeans clustering results on the original input ($\mathcal{R}^{300}$). Each color represent a certain cluster. Each data point labeled with a format indicating the participant and question number. For example, "$P2\_S1$" is answer of participant 2 to question #1. The PCA representation of embedded sentences ($\mathcal{R}^{300} \rightarrow \mathcal{R}^2$ dimension reduction) is for display purpose

In order to examine how significant is the purity values calculated from the clustering, I ran the following test, showed at the figure below: I made 1000 trials of KMeans on the recorded data and calculated the purity value for each. I made additional 1000 reference trials on a random data, where in each trial I generated a newly random data (matching to the recorded data by dimensionality and quantity). For each random data trial I ran KMeans 25 times, calculated the purity value of each and chose the best-out-of-25 value. The distributions of these values are at the figure below. Eventually, I used *Wilcoxon rank-sum test*, with these two distributions as inputs, to measure the significance of the findings. I got a $p\text{-}value < 0.001$, meaning a significant result in term of contradicting the assumption that the purity values from our recorded data, which were better than the random data purities, were received by chance.

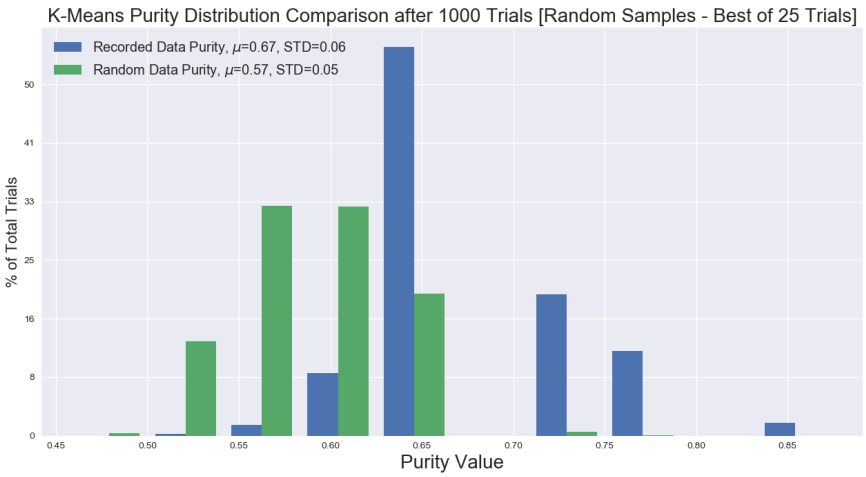

Figure 3: Purity values' comparison between 1000 trials of KMeans ran on a random data where each trial is the best purity score of another 25 KMeans trials, and another 1000 trials on the recorded data

Additional comparison domain is to alternative models. When comparing our trained model to a reference model (available online), trained with Wikipedia dataset, we can see that the resulting PCA representation looks worse comparing to our model representation, and the KMeans clustering are poorer as well (purity values of 0.41-0.54).

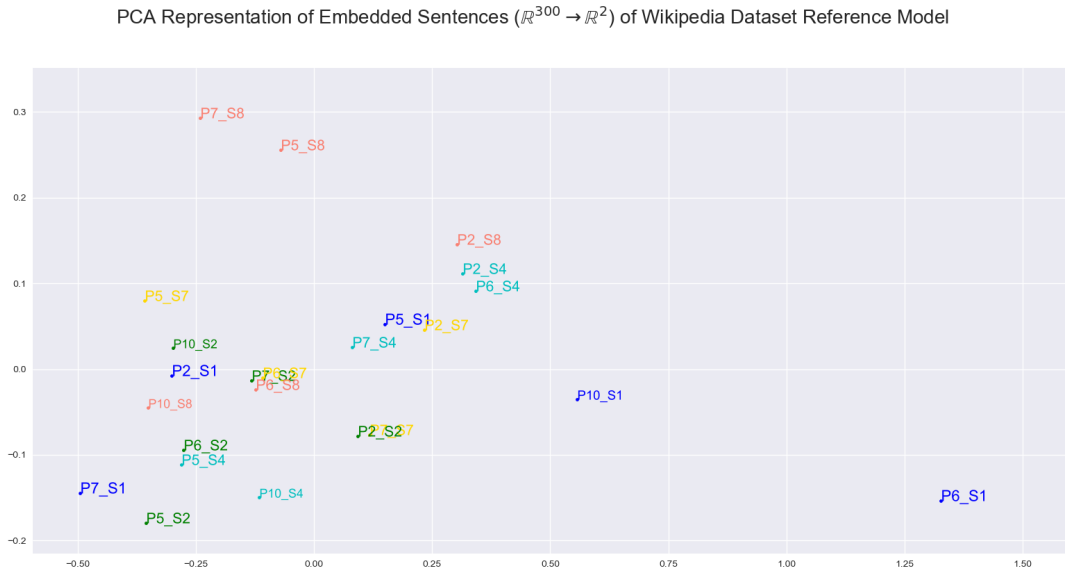

Figure 4: The same PCA and KMeans clustering as in the previous figure performed on results generated by a reference model, trained with Wikipedia dataset

Additional evaluation can be done by observing the correlations - Pearson and cosine similarity - between every possible pair of embeddings. These comparison is shown in the following figures:

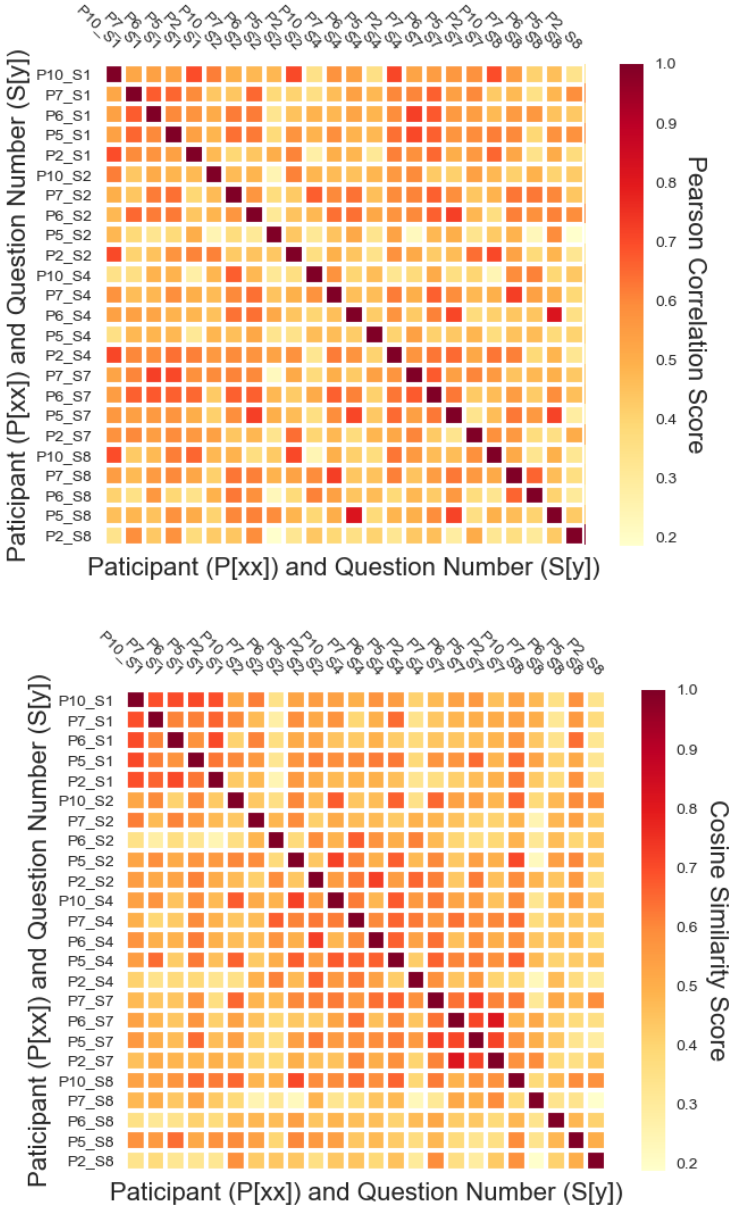

Figure 5: Pearson correlation (upper figure) and cosine similarity (lower figure) between pairs of embeddings calculated on the data in the original dimension ($\mathcal{R}^{300}$).

However, as seen in the table below, this correlations do not help in clustering the data. When comparing the average in-cluster correlation (similarity) to the average correlation of a certain cluster's embeddings with the other clusters' embeddings, it cannot be said that the in-cluster correlations are higher than the other in most of the cases (especially when observing the Pearson correlation).

	Cosine Similarity		Pearson Correlation	
Question	In-cluster	W\ rest of clusters	In-cluster	W\ rest of clusters
1	**0.659**	0.484	**0.592**	0.52
2	0.476	**0.498**	0.437	**0.502**
4	**0.585**	0.507	0.47	**0.495**
7	**0.702**	0.508	**0.534**	0.53
8	0.428	**0.462**	0.426	**0.487**

Table 1: **Correlations comparison**. Presenting the average correlation value of embeddings at the same cluster comparing to the correlation between this certain cluster's embeddings to the other clusters. The better correlations in each comparison are **emphasized**

In addition to the prediction demonstration and the general performance, which described at Appendix C (alongside the model's parameters), two main aspects of this module can be evaluated: The first is the difference between model-generated sentences (using the speech2text module) and transcribed sentences. The second aspect is the inference phase (how to fine-tune the post-training embedding process). Both of these aspects will be covered in the following sections.

5.3.1 Transcribed vs. Automated Utterances

This section is a co-evaluation of the model's performance alongside the speech2text API evaluation and the *steps* effect on the inference (which is demonstrated at the next section, 5.3.2). Below lies a comparison of the embedding representation of two sets of responses to the same questions – the first responses were generated with speech2text API and the second one by transcription (Note that there are more transcribed sentences since the model did not manage to automatically convert all the utterances at the speech2text module).

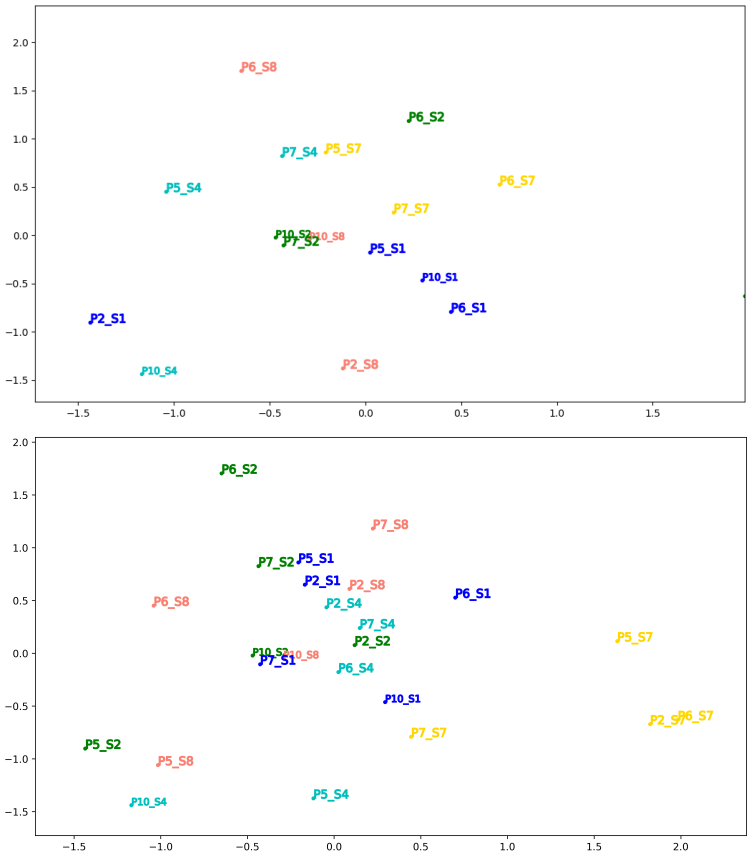

Figure 6: PCA representation of embedded sentences ($\mathcal{R}^{300} \rightarrow \mathcal{R}^2$, using $steps = 20$) where each color represents an answer to a certain question. The upper image represents the model's speech2text output sentences and the lower image is the transcribed ones.

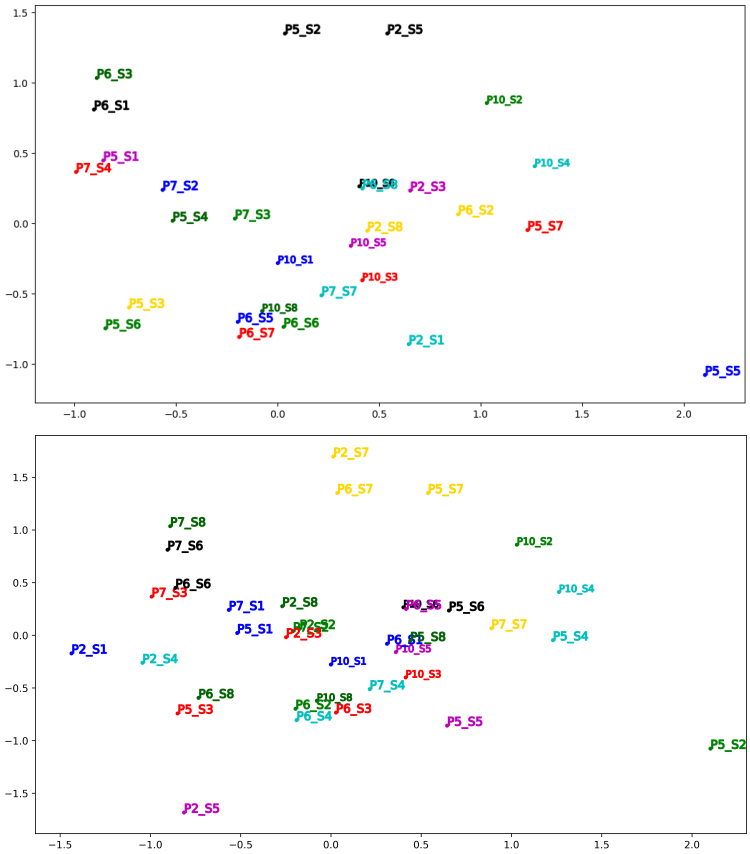

Figure 7: A similar example to Figure 5 - this time with different number of inference steps (25) and additional questions' responses.

From the two pairs of images, and additional ones I examined, I got the following insight: Although numerically the generated vectors using different *steps* number at the inference of a certain sentence differ from one another, the general relation between sentences stays the same. In this work's case, where a possible goal is to separate the responses of each question from the other responses (creating "groups of colors" in lower-dimension, as seen in the upper images), it is not clear that using transcribed sentences outscores by large margin the model's automated speech2text sentences (although we should keep in mind the speech2text API fails converting some utterances). In this work I chose presenting the transcribed sentences' embedding at the model evaluation section since they showed more accurate results.

5.3.2 Inference Steps

Additional parameters that require adjustment are related to the inference - the learning rate and number of steps (iterations). Changing these parameters impact significantly on the result, as demonstrated in the figure below. In this model case, keeping the original learning rates (from the training phase) and choosing 10 steps gave the most satisfying results.
The following figure demonstrates the effect of the *steps* number on the output. As final note for this section, I have no recommendation regarding the optimal *steps* number other than test and evaluate them.

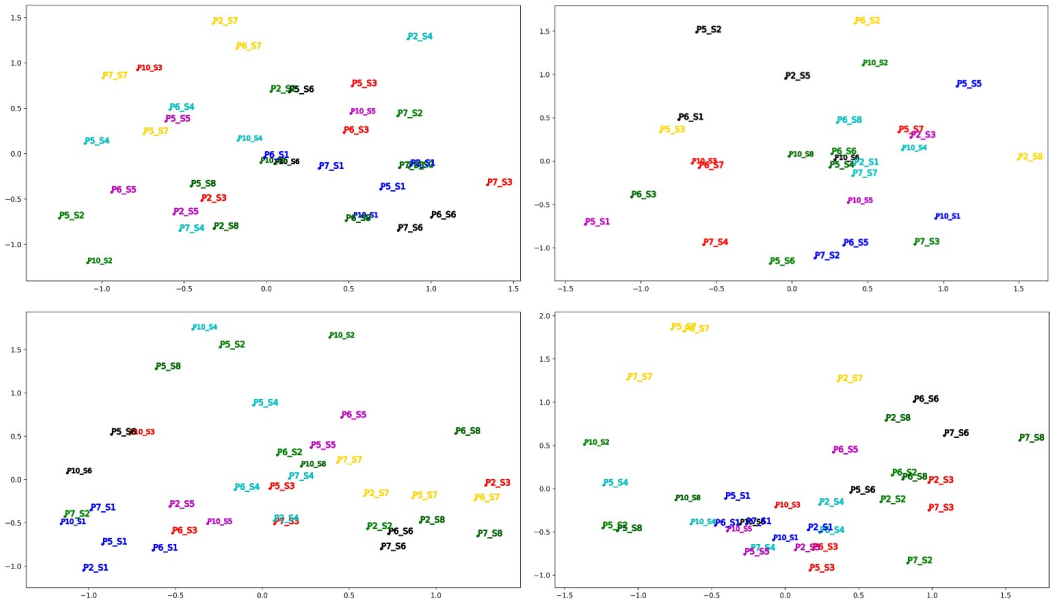

Figure 8: PCA representation of embedded sentences ($\mathcal{R}^{300} \rightarrow \mathcal{R}^{2}$) using different number of *steps* for inference. Each color represents an answer to a certain question. The *step* values from upper-right, clockwise, are 20,10,5,15.

5.4 Prosody Analysis Evaluation

In general, it is hard to evaluate prosody analysis since this analysis relates to the context of what you seek and the manipulations that are done on the data. Hence, I feel confident to say that the prosody analysis module provides a stable initial analysis of the input fed into it.
The following figures serves as a demonstration of the output for each given utterance (four different figures for each utterance). A small but important note - the entire data used for these plots generation is saved during the tool's runtime, so it can be manipulated and later-presented according to the users definitions:

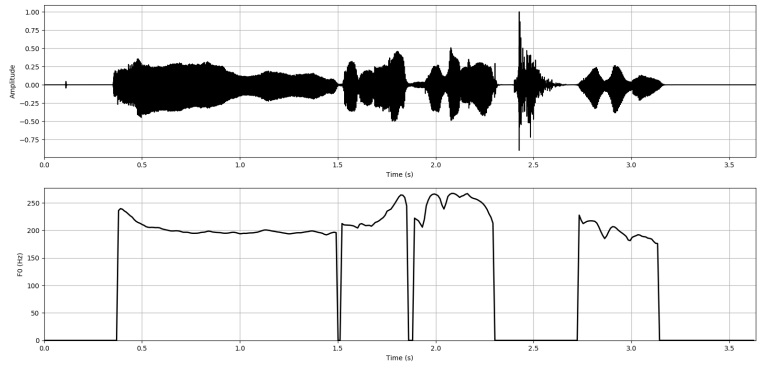

Figure 9: Amplitude and fundamental frequency of an utterance generated using *PRAAT* software.

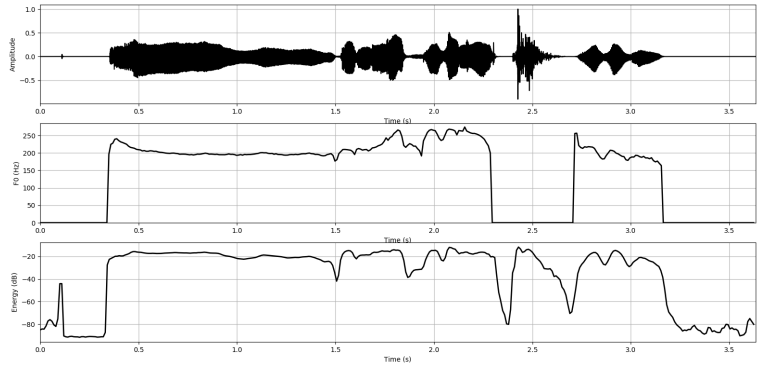

Figure 10: Energy, amplitude and fundamental frequency of an utterance generated using *RAPT* algorithm.

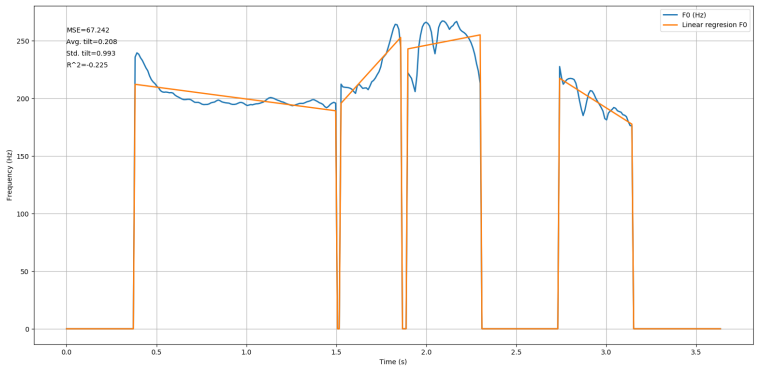

Figure 11: The fundamental frequency and the appropriate linear regression using *PRAAT* software.

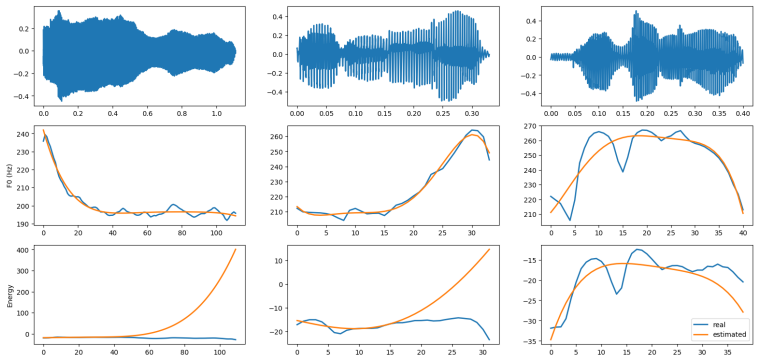

Figure 12: The energy and fundamental frequency, generated by the Legendre polynomial coefficients estimation, comparing to the measured ones (using *PRAAT* software).

Furthermore, when running a comparison between different participants' responses to the same questions it is clear that there is some correlation between the responses and some of the prosody features. In other words, looking at some feature - there is a trend between the responses of a certain participant that is observed in other participants responses as well. This is demonstrated in the following figure:

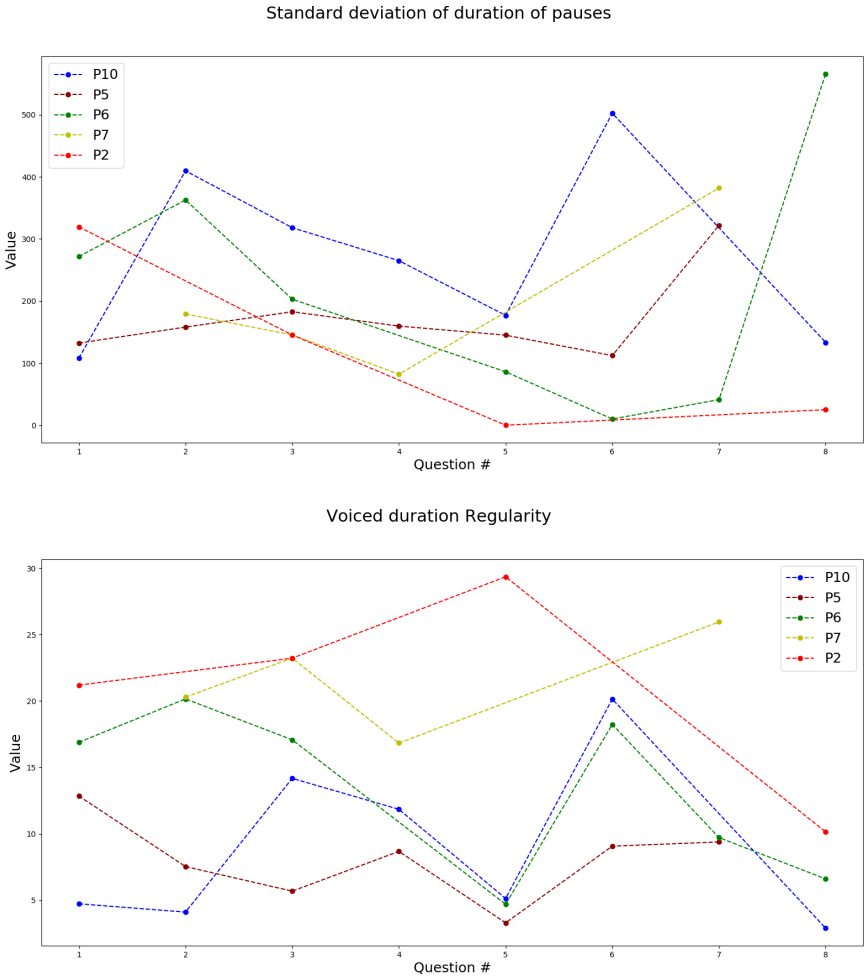

Figure 13: A comparison of 2 of the prosody features - *duration pauses STD* and *Voiced duration regularity*. Each color represent a different participant. Each x-axis number represents a certain question. Note that there are some unanswered questions by some of the participants - the circle marker indicates which questions were answered.

To generate an evaluation for the prosody features, I chose four questions for whom I had answers from the same five participants. For each prosody feature i ($1 \leq i \leq 38$) I generated five participants' vectors with four entries each ($P_n = x_1, x_2, x_4, x_8,\ n = 2, 5, 6, 7, 10$ and correspond to the participants ID described at section 5.3). Each entry x_t is the calculated value of prosody feature i from participant's n answer to question number t (the values of t correspond to the questions with the same numbers described at section 5.3).
Eventually, I calculated a correlation matrix **between the participants' vectors**. This was done using both Pearson and Spearman correlations (separately) between every pair of participant's vectors. Intuitively, For each prosody feature, the correlation results represent participants behavior along questions, or how similarly different participants answered to the same questions (in the context of a certain prosody feature). I plotted some of the 38 features' results as demonstration:

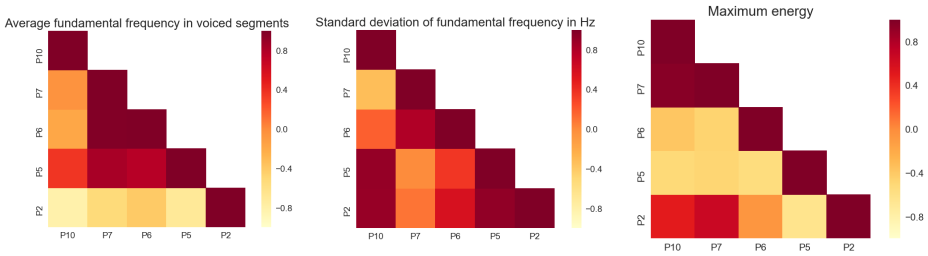

Figure 14: Correlation matrices of three different prosody features. Possible correlation values are at the range $[-1\ 1]$.

To evaluate how significant are the correlations depicted above, I conducted the following trial: for each prosody feature I generated a $4X5$ matrix, denoted by M, from its corresponding participants vectors (each vector as $1X5$ column).

$$M_{4X5} = \left(P_{i_2}\ P_{i_5}\ P_{i_6}\ P_{i_7}\ P_{i_{10}}\right),\ 1 \leq i \leq 38 \tag{2}$$

Using M, I generated additional 100 permutations of M, denoted by M', by shuffling the order of the participants vectors P_{i_2}, P_{i_5}, P_{i_6}, P_{i_7} and $P_{i_{10}}$ (individually, without mixing values between different participants). For each of these 100 M' permutations I calculated the correlation matrix as depicted above (using both Pearson and Spearman correlation values). Since each correlation matrix contains ten relevant unique values (the correlation values between different pairings of participants), I ended up with ten original correlation values and additional 1000 random values. Ultimately, my goal was searching for features with correlation matrix that shows significant connection between the sets of answers to my set of questions given by different

participants.
In order to find these features I ranked the data (1010 entries), and extracted the ranks of the ten original correlation values. I decided to use *Wilcoxon rank-sum test* to express the aforementioned significance, and explicitly its output *p-value*. By using this test, as seen at the table below, I managed to show the odds that the ranks of my ten original correlation's values was randomly selected from the entire ranked data (1010 entries). Features with *p-value* lower than 0.05 are the ones that I looked for. Both correlations – Pearson and Spearman – gave similar results (with slight preference to Pearson's ones), so only the *p-value* based on Pearson correlations' data are presented in the following table:

prosody Feature	**P-Value**
Average fundamental frequency in voiced segments	0.9
Standard deviation of fundamental frequency in Hz	**0.01**
Variability of fundamental frequency in semitones	**0.03**
Maximum of the fundamental frequency in Hz	**0.01**
Average energy in dB	0.41
Standard deviation of energy in dB	0.66
Maximum energy	0.74
Voiced rate (number of voiced segments per second)	0.72
Average duration of voiced segments	0.82
Standard deviation of duration of voiced segments	**0.05**
Pause rate(number of pauses per second)	0.62
Average duration of pauses	0.43
Standard deviation of duration of pauses	0.92
Average tilt of fundamental frequency	0.91
Tilt regularity of fundamental frequency	0.22
Mean square error of the reconstructed F0 with a 1-degree polynomial	0.31
(Silence duration) / (Voiced + Unvoiced durations)	0.28
(Voiced duration) / (Unvoiced durations)	0.69
(Unvoiced duration) / (Voiced + Unvoiced durations)	0.35
(Voiced duration) / (Voiced + Unvoiced durations)	0.39
(Voiced duration) / (Silence durations)	0.23
(Unvoiced duration) / (Silence durations)	0.66
Unvoiced duration Regularity	0.5
Unvoiced energy Regularity	0.24
Voiced duration Regularity	0.57

Voiced energy Regularity	0.62
Pause duration Regularity	0.92
Maximum duration of voiced segments	0.94
Maximum duration of unvoiced segments	0.29
Minimum duration of voiced segments	0.56
Minimum duration of unvoiced segments	0.68
rate (# of voiced segments) / (# of unvoiced segments)	**0.01**
Average tilt of energy contour	0.33
Regression coefficient between the energy contour and a linear regression	0.55
Mean square error of the reconstructed energy contour with a 1-degree polynomial	**0.02**
Regression coefficient between the F0 contour and a linear regression	0.96
Average Delta energy within consecutive voiced segments	0.9
Standard deviation of Delta energy within consecutive voiced segments	0.71

Table 2: The P-Value of each prosody feature generated by Wilcoxon rank-sum test on the data from the trials depicted above. Emphesized numbers are with P-Value> 0.05.

As observed from the emphasized P-values on the table above, there are some features which imply on a possible relation between how a person answer a question to these features' values. To conclude these section and the quality of the features with low *p-value*, we ran calculated the FDR (false discovery rate) of the entire features, with $\rho = 0.05$ and $i = 6$ (the number of features with *p-value* equal to ρ or less):

$$FDR = \frac{\rho * N}{i} = \frac{0.05 * 38}{6} = 0.317 \tag{3}$$

6 Conclusions

The *Analysis Tool for Spoken Language* is a robust tool designed for IDC's AVL (advanced virtuality lab). Alongside the independence of some of its component (i.e. prosody features stays the same, regardless to the sentence embedding results), it supplies a simple GUI for users (and easy-to-change code for developers) to control and adjust its output. Furthermore, this tool's performance can be improved. The functionality added to this tool's code allows both to improve its current building blocks (training a better text embedding model, for instance) and replacing entire building block (i.e., using another embedding method) quite easily.
On a personal note, this tool combines features that each one, independently, can be a subject for extensive research. Though this work quite diverged from the original research question I had (connecting speech to movement), my hopes are that this tool will be in use for years to come in the AVL.

7 Acknowledgments

Few people helped me with setting up the VR environment and conducting the experiment: Jonathan Giron for the general help at the lab and his encouragement, Inbar Marom for conducting the experiments and willingness to help and Amir Hagafny, who programmed the initial VR environment and its functionalities. Special thanks to Maxine Hanrieder, which technically speaking made the experiments' data capture possible as she merged my demands into her work on a bigger project. Finally, I would like to thank Prof. Doron Friedman for his guidance, support and patience. Hope this project will be a sufficient compensation for his time and effort spent on my research efforts in the last year and a half.

Appendices

A Thesis Brief

Connecting speech to body-movement is a complicated task. More than a year ago, a research proposal submitted under the name *Matching Nonverbal Content to a Virtual Human's Speech Content: A Data-Driven Approach*. This proposal aimed, using a VR (Virtual Reality) environment, to implement the following framework:

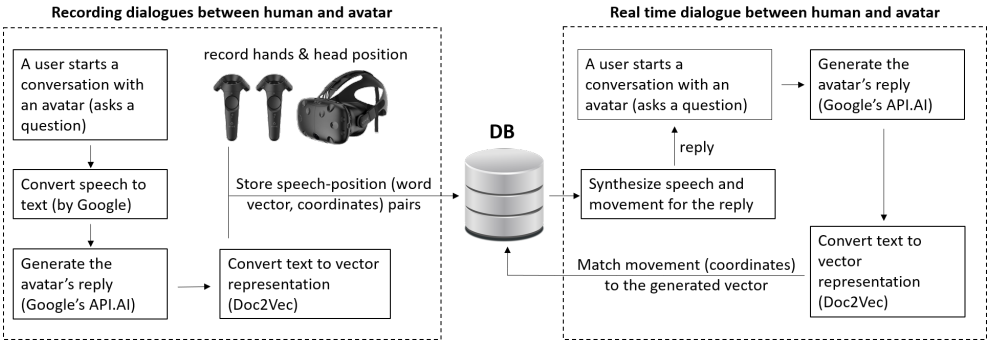

Figure 15: The thesis' framework. The left side demonstrates how the database (DB) is constructed. On the right there is a demonstration of a conversation with a virtual character that uses the recorded data in order to match text to body language (movement).

Throughout the last year I faced some difficulties that caused me to shift my original thesis plan into the project that was described in the aforementioned sections.

The research at the heart of this work was finding a relation between spoken content and body language. Prior to that stage, some preparation work had to be done in order to capture appropriate data. My work on the thesis included the following main stages:
The first stage of my work involved **building a virtual environment for the data capture scenario**. This phase included adding audio and movements (headset and hand-trackers spatial and rotational coordinates) data capture abilities to an existing virtual environment at the AVL (so this data capture mechanism can be used for other projects as well).
The second stage, which probably required the greatest amount of time, was deciding on the **text embedding (representation) model**. The chosen

model was Doc2Vec [8], which relies on its ancestor named Word2Vec [10]. Prior to the model training, I conducted a research on datasets to fit the experiment scenario (in general, short utterances). This research resulted in generating a new dataset by merging and post-processing few existing ones. Both the model and dataset generation were described in Section 3.
Over the course of the year there was a main conceptual change at the data we decided to record (due to practical reasons) - instead of **dialogue with an avatar** scenario, the data we recorded was **job interview** scenario where a participant faced two avatars interviewers. The **data record stage** took place at the second (spring) semester of 2018. A total amount of fifteen participants were recorded as part of the job interview experiment, and the data recorded from twelve of them was valid for use (relating the quality and amount of data from each record).
The last two stages, which meant to be the heart of this work, were **data analysis and manipulations** (normalization, extracting features from movements etc.) and **matching attempts between the extracted features and embedded sentences** by applying different methods on the data (principal components analysis – PCA, canonical correlation and two-branch neural network).

During the last year there were more than a few difficulties as part of this work. The most critical among these difficulties is related to the recorded data. Its small quantity (total of $\sim$250 sentences and matching movements), as well as its quality (most of the recorded data, which was the hand movements, turned out not to be useful), made the correlation research almost irrelevant and prevented me from using NN (lack of data). In addition, I found out that the $Doc2Vec$ model I trained (I actually trained and tested more than dozen models) did not provide satisfactory results in terms of obtaining high similarity between sentences with closer meaning (my assumption was that the dataset used for training does not emulate sufficiently the spoken content of a job interview). More on that in a dedicated section 3.2.
Despite the unsuccessful matching attempts aforementioned, I found that some of the features generated for the thesis' data record, as well as the data itself, can be useful in other projects. This was the initial seed for the audio *Analysis Tool for Spoken Language* project.

B Speech-to-Text Conversion Demonstration

#	Actual Text	Speech2Text result
1	"sure. ny name is Stephanie I am 24 years old I was born in argentina and i moved here four years ago, I am currently studying communication at the IDC"	"sure my name is Stephanie I am 24 years old I was born in Argentina and I moved here four years ago I'm currently studying Communications"
2	"Well i think I am a very good learner that can do pretty much any job. I like working hard and I really like learning all the time"	I think I'm a good learner that can do pretty much any job I love working hard and I really like learning all the time"
3	"I think it could be my ability just to improve myself in any position and to learn from everyone"	"I think it's that could be my ability to improve myself in every position and to learn from everyone"
4	"Yes I would. I would really like to work on my own designs. I am mainly interested in product design and interface and UX so i would really like to improve on that"	"yes I would I would really like to work on my own design some reason to sing in their product design and interface"
5	"well I am currently making about 60 shekels an hour so I would not have want to go below that but if the benefits are good enough it is a possibility"	"I'm currently making about 60$ an hour in the benefits are good enough"
6	"I think I standby my position. I would not leave my job, if not"	"I think I stand by my position I wouldn't leave my job"
7	"Best boss I ever had it is probably my current boss. He is very welcoming to criticism. he lets me work on my own projects. He always asks for my opinion"	"Best I Ever Had it's probably in my car and boss is very welcoming to create a system he lets me work on my own projects always ask for my opinions"

8	"I think I handle it pretty well. I work very well under pressure. I am currently a class representative of my entire class of 120 students and i manage them all. I listen to them I hear their complaints I raise issues and I solve them so I think I am really good"	"I think I did pretty well I work very well under pressure I'm currently at for my entire class and I listen to them I sure that complains I erase issues and I sought them so I think I'm pretty good"
9	"I think I work really well alone but I am also good with teams and eventually I think teamwork is more fun and it gets you farther than working by yourself"	"I think I work really well alone but I'm also good with teams and eventually I think he work it's more fun and it gets you farther than working by yourself"
10	"My biggest accomplishment"	"the biggest accomplishment"
11	"It will be just be here and the person I am today I think. It was a big step to move from my home and come here and start over so I really like the place I am right now "	"it will be easier on the person I am today I think it was a big stuffed animals from my home and start over so I really like the place I am right now"
12	"actually I am. I am the top of my class"	"actually I am on the top of my class"
13	"My greatest weakness. Sometimes I like to do it all everything by myself so it is hard for me to listen to criticism, but I am working on it"	"my greatest weakness sometimes I like to do it all everything myself it's hard for me to listen to criticism by the way I'm working on it"
14	"I think I like most that it is very practical"	-
15	"It is something I can really apply in my work area. I am learning by doing which is really the best for me"	"something I can apply in my work area and I'm learning by doing which is the best for me"

Table 3: A comparison between the model's speech2text output and a manual transcription of the audio file. The audio record is considered good - both by power and accent means.

C Doc2Vec Details

C.1 Model Parameters

I used the following parameters for my model (using explanations from https://radimrehurek.com/gensim/models/doc2vec.html):

1. $dm = 1$ (using PV-DM training algorithm)
2. $dbow\_words = 1$ (train word-vectors simultaneously with doc-vector training. Irrelevant for PV-DM algorithm)
3. $dm\_concat = 1$ (using concatenation of the context vectors rather than sum/average them)
4. $hs = 1$ (using hierarchical softmax for model's training rather than negative sampling)
5. $alpha = 0.025$ (initial learning rate)
6. $min\_alpha = 0.0001$ (Learning rate will linearly drop to this value as training progresses)
7. $iter = 5$ (number of iterations over the dataset while training)
8. $epochs = 10$
   note: this is not an actual parameter of the model. I used 10 epochs, and in each of these epochs I did 5 iterations ($iter = 5$) over the dataset with a fixed learning rate, resulting in total of 50 iterations over the data. The learning rate was manually reduced in a linear manner between epochs at the range $[alpha, min\_alpha]$
9. $size = 300$ (dimension of the input feature vectors)
10. $min\_count = 7$ (ignores all words with total frequency lower than this)
11. $sample = 1e-5$ (The threshold for configuring which higher-frequency words are randomly down-sampled, useful range is [0, 1e-5])
12. $window = 3$ (maximal distance between the current and predicted word within a sentence)

C.2 Model's Performance

In the following section I will briefly demonstrate what can be achieved using the chosen $Doc2Vec$ model. Since there are many ways to evaluate its performance, the focus will be on the aspects that I find more relevant to this project's scope. In general, there are two types of comparisons - prediction (finding the best similarity with some word/sentence from the training dataset) and similarity (evaluate how inputs are related to each other, without relating to the dataset itself).

C.2.1 General Behavior

In term of single words ($Doc2Vec$ trains word vectors as part of its training) prediction, we get the results in the following table. The fact that the model contains 39883 different words that appeared at least 7 (set by *min_count* parameter) times in the dataset should be considered. For example, the following figure shows what are the most similar words to "$Paris$" and "*good*" that were found in the training corpus:

Word	Similarity to **'paris'**	Word	Similarity to **'good'**
'moscow'	0.483	'late'	0.515
'rome'	0.448	'fine'	0.512
'berne'	0.445	'right'	0.491
'canada'	0.44	'afternoon'	0.477
'colombia'	0.437	'nice'	0.476
'amsterdam'	0.43	'bad'	0.468
'bremen'	0.428	'great'	0.462
'guangzhou'	0.427	'well'	0.459
'frankfurt'	0.424	'real'	0.441
'lahore'	0.423	'hold'	0.44

Table 4: A demonstration of the most similar words to 'paris' (left) and 'good' (right). Note that the model was trained lowercase words. The similarity columns indicates the cosine similarity score between 2 word vectors.

Another example is finding the word that does not match to the others in a given list of words (in practice, it is the word further away from the mean of all the other words). When given the following list of words - 'paris', 'cat', 'lion', 'dog', 'tiger' - the model returns 'paris' as the un-matched word.

When dealing with document vectors, the results are much less clear, or useful, in terms of prediction. The model contains 3409547 document (sentences) vectors, when some of the sentences appear more than once. One quite good prediction example is finding what are the 10-most similar dataset's sentences to the following one *are you trying to insult me*, which is from the dataset as well:

1. "violence of some kind"
2. "resorting to violence"
3. "i abominate violence"
4. "and resorted to violence"
5. "no violence here"
6. "violence is so"
7. "i watched every bit of news about you"
8. "makes you wonder"
9. "it kind of makes you wonder do not it"
10. "no more violence"

It is clear that the word "insult" got the context of "violence" throughout training.
A far less impressive example is trying to find the similarity to a new sentence (was not part of the training dataset). This is done by inferring the new sentence to form a vector representation (at $\mathcal{R}^{300}$) and than looking into its similarity with the document vectors from the dataset. In this case, these are the 10-most similar sentences from the dataset to "*where have you been until now it is really late and i was worried*:

1. "she gets back late"
2. "you are late i walk"
3. "you are late"
4. "jim you won"
5. "of course how stupid of me"
6. "it is very late"

7. "too late for the hospital"
8. "it is late"
9. "well we won jim".

After examining few similar cases it looks as the model "sticks" to a dominant word in the input sentence when trying to predict similar sentences from the dataset.

Another domain of comparison, which is probably the most relevant to this tool's main use, is finding similarity between pairs in a group of sentences (utterances). One use-case for such a task is the ability to classify responses according to what they originally related to. I tested this task by trying to take two groups of 4 sentences each - one group contains possible answers to the question "how is the weather", where the other group contains answers for "what is your favorite food?". For display purposes, I labeled each sentence with a number:

1. S1 = "the weather in israel is hot and wet"
2. S2 = "it is very cold here right now"
3. S3 = "today is a sunny day"
4. S4 = "the temperature now is minus ten degrees so it is cold"
5. S5 = "pizza is the best food in the world"
6. S6 = "i really like to eat chocolate"
7. S7 = "i love cakes"
8. S8 = "i am trying to eat healthy food so salad"

The following table shows the similarity of each pair of sentences. I inferred them with $alpha = 0.025$, $min\_alpha = 0.0001$ and $steps = 10$:

-	S1	S2	S3	S4	S5	S6	S7	S8
S1	-	-0.025	**0.227**	-0.003	-0.063	-0.072	0.006	-0.068
S2	-0.025	-	-0.025	**0.294**	-0.073	-0.003	0.038	0.021
S3	**0.227**	-0.025	-	-0.028	-0.044	0.018	0.029	0.003
S4	-0.003	**0.294**	-0.028	-	0.264	-0.025	-0.01	-0.135
S5	-0.063	-0.073	-0.044	**0.264**	-	-0.042	0.054	0.122
S6	-0.072	-0.003	0.018	-0.025	-0.042	-	-0.043	**0.137**
S7	0.006	0.038	0.029	-0.01	**0.054**	-0.043	-	-0.112
S8	-0.068	0.021	0.003	-0.135	0.122	**0.137**	-0.112	-

Table 5: Similarity between the aforementioned sentences. $S1$-$S4$ are the weather group sentences, $S5$-$S8$ are from the "food" group. The most similar pair in each row is emphasized with **bold**.

As seen in the table above - on one hand, the similarity value are relatively low (the cosine similarity's maximal value is 1). However, when the task is classification, than the model showed satisfying results. Other than one sentence from the "food" group ("pizza is the best food in the world"), that its best pair was from the "weather" group ("the temperature now is minus ten degrees so it is cold"), rest of the sentences got a matching pair from their original group.

D User Guide

D.1 Basic User

Welcome to AudextBox!
This tool provides, for a given audio and XDF files recorded at a certain experiment, the following features: splitting the audio file into utterances sub-files, converting them into text, generating an embedded representation for each utterance as well as comprehensive prosody analysis raw-data for further inspection.
To use this tool we have supplied you a basic GUI (**G**raphical **U**ser **I**netrface) which requires to do the following:

1. Double-click on the "ABM_Run" icon on the Desktop.
2. A window with the following parameters would be displayed shortly after you click:

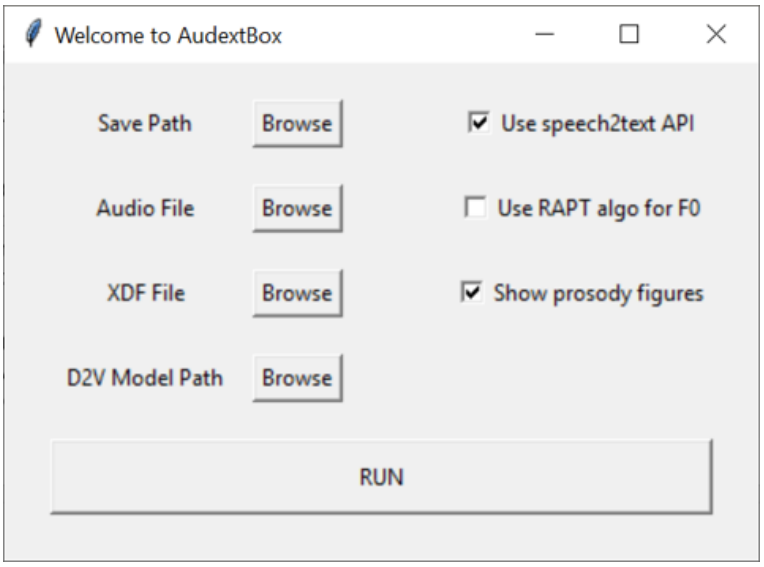

Figure 16: GUI layout

(a) *Save Path* - defines the directory where the results will be saved in.

(b) *Audio File* - The audio file that was recorded in the experiment and needs to be analyzed.

(c) *XDF File* - The matching XDF file that was created at the same experiment [of the audio file].

(d) *D2V Model Path* - A path to the desired *Doc2Vec* model. If not chosen, the tool will use its default model (the one whose performance was analyzed at the previous sections).

(e) *Use speech2text API* - If checked, the program will convert the audio files into text, using Google's online speech2text API.

(f) *Use RAPT algo for F0* - if checked, *RAPT* algorithm would be in use for the *F0* estimation, otherwise *PRAAT*.

(g) *Show prosody figures* - if checked, the program will show every prosody analysis figure that is created, and wait for closing the figure's window upon continuing to the next figure (since there are 4 different figures for each utterance, it's recommended to uncheck this option).

3. When finished - hit the *RUN* button!

That's it! Once the program starts running, it creates a new folder using the input audio file name under *Save_path* you defined (after removal of *.wav* suffix and *record_* prefix, if exists. At *JOY* experiment it means the folder name is the date&time of the record). The data is saved inside this new folder in the following structure:

1. *wavs* directory - contains a dedicated *wav* file for each utterance (split from the main input audio file)

2. *audio_data* directory - contains *csv* file for each *wav* file (named *utterance_xx_audio_data*, where *xx* is the utterance's number). Each file stores a timestamped raw-data of *F0*, amplitude and energy of the certain utterance. The timestamps are "global", meaning related to the initial input audio file, and not to the matching audio file at the *wavs* folder.

3. *dynamic_features* directory - contains *csv* file for each *wav* file with the prosody analysis' dynamic features (which are the coefficients of Legendre polynomial model of both the energy and fundamental frequency extracted from the audio file)

4. *figs* directory - stores a graphic representation of the prosody analysis results of each utterance (based, among the rest, on the data stored at *dynamic_features* and *audio_data* folders). For each utterance there are 4 different figures.

5. *utterances_data.json* - the main output file. Stores the following fields for each utterance:

(a) *File Name* - the specific utterance *wav* file name

(b) *Timings* - *Start Time* and *End Time* (relative to the initial input audio file) in *ms*

(c) *Text* - the speech2text conversion for the utterance audio file

(d) *Doc2Vec* - the embedded representation (named *Embedded sentence* in the file) of the text (dimension - 300)

(e) *Prosody Analysis* - contains 38 static features (described here D.1.1)

D.1.1 Prosody Analysis Static Features

1. Average fundamental frequency in voiced segments
2. Standard deviation of fundamental frequency in Hz
3. Variablity of fundamental frequency in semitones
4. Maximum of the fundamental frequency in Hz
5. Average energy in dB
6. Standard deviation of energy in dB
7. Maximum energy
8. Voiced rate (number of voiced segments per second)
9. Average duration of voiced segments
10. Standard deviation of duration of voiced segments
11. Pause rate(number of pauses per second)
12. Average duration of pauses
13. Standard deviation of duration of pauses
14. Average tilt of fundamental frequency
15. Tilt regularity of fundamental frequency
16. Mean square error of the reconstructed F0 with a 1-degree polynomial
17. (Silence duration) / (Voiced + Unvoiced durations)

18. (Voiced duration) / (Unvoiced durations)
19. (Unvoiced duration) / (Voiced + Unvoiced durations)
20. (Voiced duration) / (Voiced + Unvoiced durations)
21. (Voiced duration) / (Silence durations)
22. (Unvoiced duration) / (Silence durations)
23. Unvoiced duration Regularity
24. Unvoiced energy Regularity
25. Voiced duration Regularity
26. Voiced energy Regularity
27. Pause duration Regularity
28. Maximum duration of voiced segments
29. Maximum duration of unvoiced segments
30. Minimum duration of voiced segments
31. Minimum duration of unvoiced segments
32. rate (# of voiced segments) / (# of unvoiced segments)
33. Average tilt of energy contour
34. Regression coefficient between the energy contour and a linear regression
35. Mean square error of the reconstructed energy contour with a 1-degree polynomial
36. Regression coefficient between the F0 contour and a linear regression
37. Average Delta energy within consecutive voiced segments
38. Standard deviation of Delta energy within consecutive voiced segments

References

[1] Koteswara Rao Anne, Swarna Kuchibhotla, and Hima Deepthi Vankayalapati. *Acoustic Modeling for Emotion Recognition*. Springer Publishing Company, Incorporated, 2015.

[2] E. Azarov, M. Vashkevich, and A. Petrovsky. Instantaneous pitch estimation based on rapt framework. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2787–2791, Aug 2012.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[4] Paul Boersma and David Weenink. Praat: doing phonetics by computer [computer program], 2018.

[5] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Comp utational Linguistics, ACL 2011*.

[6] Najim Dehak, Pierre Dumouchel, and Patrick Kenny. Modeling prosodic features with joint factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:2095–2103, 2007.

[7] Mihail Eric and Christopher D. Manning. Key-value retrieval networks for task-oriented dialogue. *CoRR*, abs/1705.05414, 2017.

[8] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196. JMLR Workshop and Conference Proceedings, 2014.

[9] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*, 2017.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[12] Juan Rafael Orozco, Juan Vasquez, J Vargas-Bonilla, R Arora, N Dehak, P.S. Nidadavolu, Heidi Christensen, Frank Rudzicz, M Yancheva, Hamid R. Chinaei, A Vann, N Vogler, Tobias Bocklet, Milo Cerak, Julius Hannink, and Elmar Noeth. Neurospeech: An open-source software for parkinson's speech analysis. *Digital Signal Processing*, 07 2017.