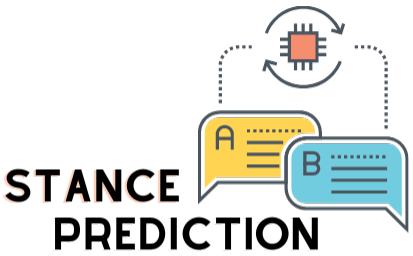
NATURAL LANGUAGE PROCESSING - PROJECT REPORT

Lorenzo Pratesi, Martina Rossini, Riccardo Foschi, Vairo Di Pasquale
{lorenzo.pratesi2, martina.rossini3, riccardo.foschi4, vairo.dipasquale}@studio.unibo.it
https://github.com/NLP-Team-Unibo/stance-prediction

IMPLICIT STANCE PREDICTION ON POLITICAL DEBATES USING SPEECH FEATURES



Artificial Intelligence Master's Degree

Alma Mater Studiorum - University of Bologna

June 22 2022

Contents

1 Summary

Nowadays, it is becoming more and more crucial for large corporations to predict the position of users on specific topics using the available information. While humans are quite capable of assessing correct stances, Machine Learning models are often not up to the task; therefore, in this project, we focused on the Stance Prediction task.

Stance prediction is the task of classifying the stance of a particular argument expressed by a speaker, towards a certain target (motion). We are going to address the problem where only two types of stances are available (*supporting* or *opposing*) without considering the motion explicitly, but trying to infer it from the speech.

This work focused on the IBMDebater "Debate Speech Analysis" dataset (8), which provides both speeches and their transcriptions labeled with the motion and their stance.

The main focus of this work is to try to see if using spoken features together with text could lead to an improvement of performance in detecting stances.

We address the stance prediction task by developing three kind of models:

- **Text model**: tries to predict the stances of the speech transcriptions. Its core is DistilBERT (11).
- **Audio model**: tries to predict the stances of the speeches. Its core is wav2vec 2.0 (1).
- **Multimodal model**: tries to predict the stances using both speeches and their transcriptions. It combines both the aforementioned models.

Each model features a pre-trained architecture with a classification head attached at the end of it. We then fine-tuned each model by freezing the weights of the pre-trained model up to a certain layer and training the remaining part, including the classification head.

We evaluated our models using accuracy as a metric. We chose the various hyperparameters of the models by validating the results on the provided validation set. The evaluations on the test set showed that the best Text model achieved 93.82% accuracy, the best Audio model achieved 92.04% accuracy and the best Multimodal model achieved a 94.65% accuracy, showing a little improvement in using both audio and text signals.

This paper summarizes all the steps that we have followed, in particular, describes the used dataset, the models created, and finally, an analysis of the obtained results.

2 Background

Today, it is becoming increasingly important for large companies to predict users' attitudes on certain topics based on data they collected. While humans are quite capable of estimating correct attitudes, machine learning models are often not up to the task; therefore, in this project, we focused on the task of Stance Prediction. It's a well-established procedure in Natural Language Processing that plays an influential role in analytical studies measuring public opinion on media platforms, particularly on political and social issues; its main objective is to allow us to understand a person's position on a topic of discussion by having in input the topic and a comment made by an author. The result of this analysis is usually from this set: "Favor", "Against" and optionally a position of neutrality towards a given argument. Stance detection is also related to sentiment analysis, where we aim to define whether a piece of text is positive, negative, or neutral in general instead of concerning a given topic of interest.

In this work, we approached the problem by developing models based on neural architectures; more in detail, we created three different models, one based on the text data, one on the audio data, and a third that combines these two. Our main objective was to see whether employing multimodal information could help the models' performance.

Note that while we formally refer to the task we are trying to solve as "stance prediction" our work lies at a crossroad between sentiment analysis and actual stance prediction. Indeed, all our models receive as input only the text/audio from which the stance should be extracted: the target, which is called *motion* in our dataset, is left implicit. The rationale behind this choice is twofold: the topic of discussion is more likely to be known in advance for textual input than it is for speech recordings and thus an acoustic model that doesn't rely on the presence of this input information could have more practical uses. Moreover, we wanted to try and analyze whether a language model based on transformers (i.e.: BERT or DistilBERT) would be able to automatically – and implicitly – infer the topic of discussion from the text; if that's not the case, when analyzing the predictions, we should be able to notice a strong correlation with the polarity of the used tokens, meaning that the predictions more closely resemble those of a sentiment analysis task. Notice that such an implicit prediction of stance is further complicated by the fact that not all the motions in our dataset are "positive", meaning that we have both motions of the type "We should ban X" and motions of the type "We should adopt X". This is a common characteristic of stance detection and it impacts whether an instance is labeled as "pro" or "con" (Table 1). In addition, the same stance toward a given target can be deduced both from texts that use words with negative polarity and from texts that use words with positive polarity (7).

Table 1: An example of possible motion-argument pairs and their corresponding stance

Motion	Speaker argument	Stance
We should adopt open source software	Adopt open source software	pro
We should not adopt open source software	Adopt open source software	con
We should adopt open source software	Do not adopt open source software	con
We should not adopt open source software	Do not adopt open source software	pro

2.1 Audio signal processing

As previously stated, the stance prediction task aims to understand a person's position on a certain topic. The input data is usually of textual format, but in recent years attempts (6) have been made to also include audio data, which represents - in principle - a richer signal, from which a model could learn to capture not only words but also emotions and moods.

A sound signal is produced by a disturbance of particles through an air medium or, more generally, through any medium. Associated with this we usually have variations in air pressure, which cause the ear's diaphragm to vibrate and allows the listener to perceive the sound (9). We can record the intensity of the pressure variations and plot them over time to obtain a sound wave. To digitize this continuous signal, we must measure the amplitude of the wave at fixed intervals of time: each of these measurements is a *sample* and we call *sample rate* the number of samples per second.

Waveforms are rich signals and have high variability and thus have been considered, historically, hard to approach directly: traditional audio machine learning relied on digital signal processing techniques to extract features. In recent years deep learning approaches forgone this feature engineering phase and simply converted the waveform to a spectrogram and applied standard 2D convolutions on this image. These architectures can reach impressive results in tasks like sound classification (5) or audio separation (4) but are based on the assumption that spectrograms are images, which is not completely accurate. Indeed, images have spatial meanings while the spectrograms' axes stand for time and frequency: moreover, traditional CNN architectures are designed so that they can detect and combine shapes/edges, but it's not clear whether this domain assumption is correct when applied to audio. Thus, researchers have also started to develop ad-hoc architectures which are based on temporal convolutions (i.e.: they apply 1D convolutions over time directly on the waveform) and are thus able to receive as input the raw audio files.

2.2 Wav2vec 2.0

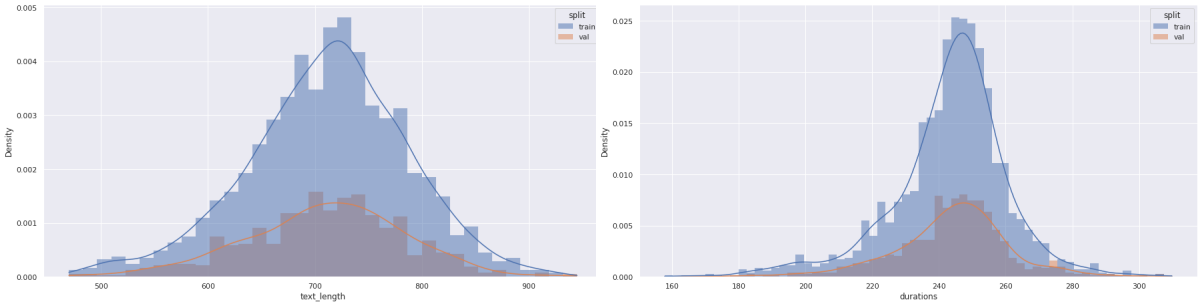
wav2vec 2.0 is one of the current state-of-the-art models for contextual speech representation: it's composed of a multi-layer convolutional feature encoder that processes a raw input waveform and outputs a latent representation for every time step. Parts of this sequence are then masked, similarly to what is done for masked language modeling; then, the latent representations are fed into several Transformer layers, which captures dependencies over the entire sequence of latent representations (1).

2.3 DistilBERT

DistilBERT is a language model based on Transformers that was trained using knowledge distillation from BERT. The basic idea of this technique is to transfer knowledge from a large model (also called *teacher*) to a smaller one (called *student*) which is trained to mimic the teacher's output. Indeed, large models have a higher knowledge capacity but also have often prohibitive computational requirements. DistilBERT is 40% smaller than BERT but achieves similar performances on a variety of downstream tasks, while also being 60% faster at inference time (11).

2.4 Dataset and Preprocessing

For this project we used the IBM Debater 'Debate Speech Analysis' dataset (8): it contains a set of speeches discussing a single motion and they can be either *supporting* – meaning that the speaker is arguing in favor of the topic of interest – or *opposing*, meaning that the speaker is arguing against the motion. Each speech is also accompanied by its textual transcription, which has on average a length of 738 tokens; the distribution of the number of tokens in each text and that of the duration in seconds of each audio file are shown in Figure 1.



(a) Text sequence lengths after tokenization

(b) Audio file duration in seconds

Figure 1: Texts and audios length distribution

The textual data were pre-processed using the DistilBert tokenizer provided by the HuggingFace `transformers` library (14): it retrieves a word sequence from a string by splitting it into spaces and

punctuation and then uses a subword tokenization algorithm called WordPiece to decompose rare words into meaningful subwords. This method also inserts a particular token called `[CLS]` at the start of each input sequence and then takes care of truncating the text so that it is no longer than 512 tokens, the maximum supported length for DistilBERT and many other Transformer models.
Notice that, while all our scripts can be executed using both the cased and uncased version of this tokenizer, we expect the choice to have a limited impact on performance: indeed, Figure 2 shows how the most common cased words are not proper names but regular, every-day words like 'we', 'so', 'the' and 'it'.

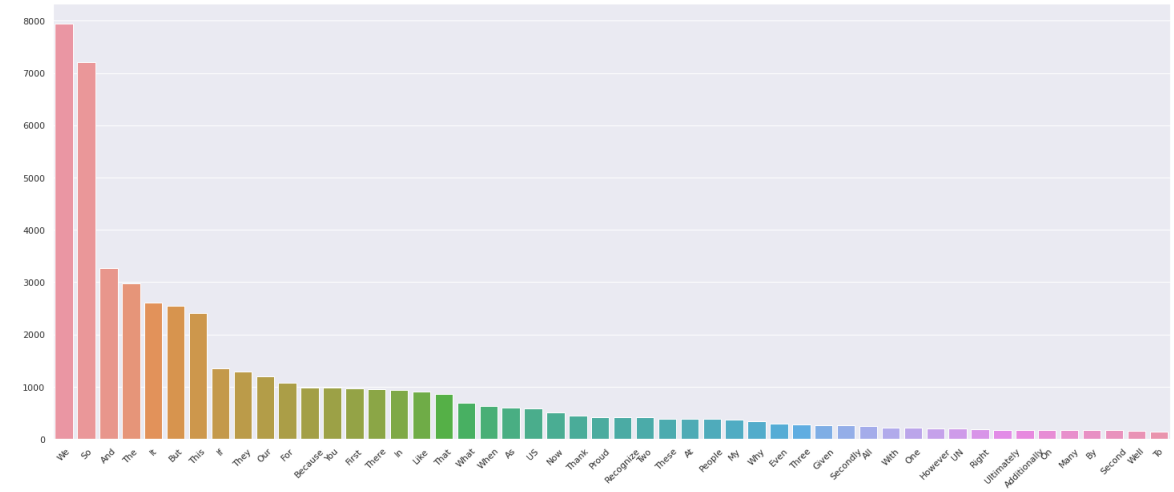


Figure 2: Most common words containing at least one upper case character

As we can see from Figure 1b, all audios have a duration of at least 160 seconds (i.e. almost 3 minutes), which makes them too long to be directly processed by a deep learning model: we thus choose to truncate each of them to just 15 or 10 seconds of length. While this choice implies discarding a big portion of the wavelength, we believe that those first seconds may already be enough for the model to gather some information regarding intonation, rhythm, and stress of the speech (prosody), which can be useful when trying to decide if a speaker is in favor or not of a certain motion. Finally, since wav2vec 2.0 (i.e.: our selected audio model) expects an input with a sample rate of 16000, we also adapt our audio files to this.

During our experiments, we used the train, validation, and test split proposed in (8), where 60% of the motions were reserved for training and 20% each went into test and validation. It's important to note that, when generating these splits, the original authors took care to keep the number of supporting and opposing statements well balanced and that, indeed, this dataset doesn't suffer from a class imbalance problem.

3 System Description

We decided to develop three different models to tackle the stance prediction problem. Since the data is in both text and audio form, we decided first to use two independent architectures, one for the text and one for the speech data respectively (Text and Audio models). We also developed an architecture that takes as input both text and speeches, to see if using both would be beneficial in discriminating the stances. For example, a speaker could say sentences in a specific emotional manner that could be better represented by an acoustic model rather than a language model.

3.1 Text Model

We decided to use as language model DistilBERT (11) because of its powerful representative capacity and its low resource demands. DistilBERT is an architecture trained using knowledge distillation on

BERT (3). As mentioned before, DistilBERT takes as input a tensor of tokens id and transforms them into a sequence of contextualized word embeddings with the hidden size of 768 using 6 consecutive Transformer layers. The word embedding corresponding to the special `[CLS]` token is the only one used for classification, as done in (12). A classification head is then attached to that embedding vector. We experimented with two different types of classification head:

- A linear layer of input size of 768 and output size of 768 with a ReLU activation called **pre-classifier**, followed by a last linear layer of input size of 768 and output size of one called **classifier**.
- A ReLU followed by a linear layer of input size of 768 and output size of one called **classifier**.

A sigmoid function is attached at the end of the classification head to create a binary classification setting.

3.2 Audio Model

We used as acoustic model wav2vec 2.0 (1), an architecture trained for learning representations from raw audio data in a self-supervised way. wav2vec 2.0 can be fine-tuned to address different downstream tasks. We decided to use this architecture because of its powerful representation capacity. Wav2vec takes as input a raw audio signal (a waveform) and encodes it using several blocks of temporal convolutions. The output is then passed as input to 12 consecutive Transformer layers, producing several contextualized representations with the hidden size of 768. We decided to use an average time pooling on those representations to have a unique vector for classification, as done previously here (13). A classification head is then applied to the resulting vector. We experimented with the same classification heads used for the Text Model.

3.3 Multimodal Model

To combine the text and the audio model, we simply decided to drop their classification heads (or just the classifier part of them), concatenate the output of the two architectures, and apply a new classification head to it. This one is composed of a ReLU followed by a linear layer of input size of 1536 and output size one, with the usual sigmoid function attached at the end.

(a) Text model (b) Audio model

Figure 3: Text and audio models

4 Experimental Setup and Results

We are now going to describe the experimental setup used to train each architecture described before. In all the experiments, we used the provided validation set to benchmark the training procedure. The loss function used to train all the architectures is binary cross-entropy. We optimize using Adam, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ used to compute the running averages of the gradient and its square respectively. Different starting learning rates are used for each experiment. We also experimented with different starting learning rates for different components of the same architecture. For some experiments, we used also a step learning rate scheduler with a step size of 5 and $\gamma = 0.2$. For all the architectures, we employed some dropout layers with variable values and an early stopping criterion to add regularization.

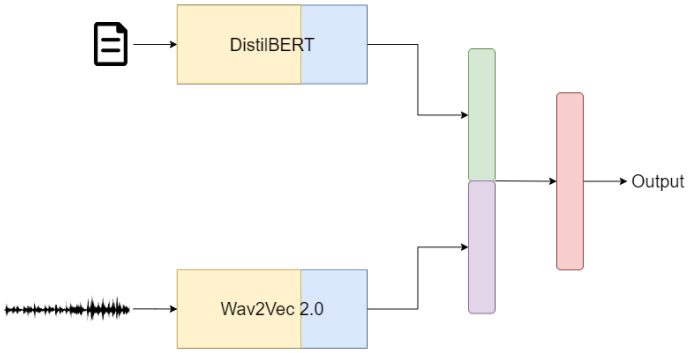


Figure 4: Multimodal model

4.1 Text Model

For all the experiments, we froze all the weights of the DistilBert module but the ones of the last 2 Transformer layers, to fine-tune those and train the classification head. We also tried different dropout values and both the case-sensitive and insensitive tokenizers. All the experiments are shown in Table 2.

Table 2: Hyperparameters for the Text Model

TEXT MODEL	CASED	DROPOUT POST-BERT	DROPOUT PRE-CLASSIFIER	PRE CLASSIFIER	LR	LR SCHEDULER
default		0.3	0.3	✓	2e-5	
cased	✓	0.3	0.3	✓	2e-5	
no_pre		0.3	0.3		2e-5	
high_drop		0.4	0.4	✓	2e-5	
multiple_lr		0.3	0.3		2e-5 distilbert | 1e-3 others	
default_sched		0.3	0.3	✓	2e-5	✓
super_high_drop		0.5	0.5	✓	2e-5	
no_pre_high_drop		0.4	0.4		2e-5	
cased_no_pre_high_drop	✓	0.4	0.4		2e-5	

4.2 Audio Model

We froze all the weights of the feature encoder of wav2wec 2.0 (the temporal convolutions), to fine-tune only the layers with Transformers (13) and train the classification head. We decided also to cut some Transformer layers to reduce the computational complexity while keeping a good representation capacity. We also tried different dropout values. All the experiments are shown in Table 3.

Table 3: Hyperparameters for the Audio Model

AUDIO MODEL	CHUNK LENGTH	DROPOUT POST-AUDIO	DROPOUT PRE-CLASSIFIER	N TRANS-FORMERS	N TRAINABLE LAYERS	PRE CLASSIFIER	LR	LR SCHEDULER
default	15	0.3	0.3	4	4	✓	2e-5	
transformers_6_6	15	0.3	0.3	6	6	✓	2e-5	
high_drop	15	0.4	0.4	4	4	✓	2e-5	
super_high_drop	15	0.5	0.5	4	4	✓	2e-5	
multiple_lr	15	0.3	0.3	4	4	✓	2e-5 wav2vec2 | 1e-3 others	
default_sched	15	0.3	0.3	4	4	✓	2e-5	✓
no_pre	15	0.3	0.3	4	4		2e-5	
transformers_6_6_super_high_drop	15	0.5	0.5	6	6	✓	2e-5	
transformers_6_6_no_pre	15	0.3	0.3	6	6		2e-5	
transformers_8_8_no_pre	15	0.3	0.3	8	8		2e-5	
transformers_10_10_no_pre	15	0.3	0.3	10	10		2e-5	
transformers_12_12_no_pre	10	0.3	0.3	12	12		2e-5	

4.3 Multimodal Model

We tried different settings for building this model. Most of the decisions are based on the results obtained on the validation set with the two independent models. After we decided on the design of the model, we experimented with three macro setups:

- freeze the weights as described above for each independent module and train the whole architecture;
- warm-start by loading the weights of the independent modules, freeze all the weights but the new classifier and train it;
- warm-start by loading the weights of the independent modules, freeze the weights up to a certain point and fine-tune.

All the experiments are shown in Table 4.

Table 4: Hyperparameters for the Multimodal Model

MULTIMODAL MODEL	MODEL TEXT NAME	MODEL AUDIO NAME	LOAD TEXT	FREEZE TEXT	LOAD AUDIO	FREEZE AUDIO	DROPOUT CLASSIFIER	LR
default	default	default					0.3	2e-5
load_both_freeze	default	default	✓	✓	✓	✓	0.3	2e-5
text_2_audio_6_6	no_pre_high_drop	transformers_6_6_no_pre					0.3	2e-5
text_2_audio_6_2_load	no_pre_high_drop	transformers_6_2_no_pre	✓		✓		0.3	2e-8 audio/text | 2e-5 classifier
text_2_audio_6_2_freeze	no_pre_high_drop	transformers_6_6_no_pre	✓	✓	✓	✓	0.3	2e-5
text_2_audio_8_8	no_pre_high_drop	transformers_8_8_no_pre					0.3	2e-5
text_2_audio_8_2_load	no_pre_high_drop	transformers_8_2_no_pre	✓		✓		0.3	2e-8 audio/text | 2e-5 classifier
text_2_audio_8_2_freeze	no_pre_high_drop	transformers_8_2_no_pre	✓	✓	✓	✓	0.3	2e-5
text_2_audio_10_10	no_pre_high_drop	transformers_10_10_no_pre					0.3	2e-5
text_2_audio_10_2_load	no_pre_high_drop	transformers_10_2_no_pre	✓		✓		0.3	2e-8 audio/text | 2e-5 classifier
text_2_audio_10_2_freeze	no_pre_high_drop	transformers_10_2_no_pre	✓	✓	✓	✓	0.3	2e-5
text_2_audio_12_12	no_pre_high_drop	transformers_12_12_no_pre					0.3	2e-5
text_2_audio_12_2_load	no_pre_high_drop	transformers_12_2_no_pre	✓		✓		0.3	2e-8 audio/text | 2e-5 classifier
text_2_audio_12_2_freeze	no_pre_high_drop	transformers_12_2_no_pre	✓	✓	✓	✓	0.3	2.00e-5

4.4 Results

Since the dataset does not suffer from a class imbalance problem, we decided to evaluate our models using accuracy as a metric. We also analyzed the confusion matrix as well as some other metrics like precision, recall, and F1-score, but we did not use them as metrics for taking decisions. All the obtained results for the Text, Audio and Multimodal models are shown in Tables 5, 6 and 7 respectively.

Table 5: Text Models Results

TEXT MODEL	val accuracy	test accuracy
default	98.13%	92.27%
cased	97.56%	92.59%
no_pre	98.13%	92.86%
high_drop	**98.28%**	93.27%
multiple_lr	97.99%	**93.82%**
default_sched	97.85%	92.31%
super_high_drop	98.13%	93.41%
no_pre_high_drop	**98.28%**	**93.82%**
cased_no_pre_high_drop	**98.28%**	93.41%

Table 6: Audio Models Results

AUDIO MODEL	val accuracy	test accuracy
default	91.97%	88.06%
transformers_6_6	93.98%	90.26%
high_drop	90.78%	87.24%
super_high_drop	90.54%	86.00%
multiple_lr	93.12%	88.88%
default_sched	91.40%	87.79%
no_pre	91.11%	87.10%
transformers_6_6_super_high_drop	93.55%	91.08%
transformers_6_6_no_pre	95.12%	91.49%
transformers_8_8_no_pre	95.84%	91.76%
transformers_10_10_no_pre	**96.99%**	91.63%
transformers_12_12_no_pre	96.70%	**92.04%**

Table 7: Multimodal Models Results

MULTIMODAL MODEL	Validation accuracy	Test accuracy
default	98.14%	93.00%
load_both_freeze	97.99%	93.83%
text_2_audio_6_6	97.71%	93.96%
text_2_audio_6_2_load	97.85%	93.28%
text_2_audio_6_2_freeze	97.71%	93.55%
text_2_audio_8_8	97.99%	93.42%
text_2_audio_8_2_load	97.99%	93.69%
text_2_audio_8_2_freeze	97.99%	93.69%
text_2_audio_10_10	98.28%	93.92%
text_2_audio_10_2_load	97.85%	94.10%
text_2_audio_10_2_freeze	97.85%	94.24%
text_2_audio_12_12	98.42%	**94.65%**
text_2_audio_12_2_load	97.71%	93.96%
text_2_audio_12_2_freeze	**98.57%**	93.28%

5 Analysis of Results

For this analysis of the results, we focused on our models' predictions on the provided test set and manually inspected a sample of the misclassified examples report our findings in Table 8. This Table shows a brief description of the error types, along with an example of where those problems were detected. Note that these errors are common in both the text-only and the multimodal versions of our architecture, even if the number of errors of type *motion/text mismatch* seems to go down in the second case.

Table 8: Common error types

speech-id	motion	text	error type
TL_3173_open-source_con_WS_implicit	We should adopt open source software	"Adopt open source software.So, we think there are a couple reasons why open source software is less preferable, than proprietary software" ... "For all these reasons, we should not adopt open source software."	cropping speech loss
TL_2107_agriculture_con_DJ_implicit	We should not subsidize agriculture	"We should subsidize agriculture.In this round, we're gonna make three main arguments about why subsidizing agriculture is a positive action" ... "For these reasons, we should subsidize agriculture."	motion/text mismatch

As we can see, there are two main families of common errors; the first one is related to the fact that we choose to deal with the 512-token limit for the Text Model by simply cutting off each example's text. This comes with an obvious disadvantage: oftentimes when one is arguing against a certain topic, he may decide to start by enumerating its good qualities only to later dismantle them. By restricting ourselves to working only with the start of the transcript we may completely ignore this important part of the speech, which can lead to misclassified examples.

The second common error type we found when analyzing our results is related to our choice not to pass the motion as input to our models: indeed, while it seems that the architectures we used can understand what is the topic of discussion as well as the speaker's opinion on it, it's also true that they have no way of knowing whether the motion, according to which an example was labeled, is "positive" (i.e.: we should enforce X) or "negative" (i.e.: we should refuse X). Thus, a big portion of our models' errors are made on instances whose original motion contains a negation: suppose you have a text i which was labeled as "pro" concerning a negative motion, i.e.: we should refuse X; this means that the speaker is actually arguing against X. What seems to happen in practice is that our models can understand that X

is the topic of discussion and thus predict text i to be of class "con" instead of "pro". A way to limit this kind of error could be to manipulate the dataset so that all the motions are positive and change the labels accordingly; one could also try to use a flag to communicate to the model whether the motion for a certain example is positive or negative.

It's also worth pointing out that there are instances that were misclassified as in favor of a certain motion even when the speaker used words with a very strong negative polarity like "evil", "horrible" and "crazy" (i.e.: instance with id 1706, which is arguing against abstinence-only sex education). This proves that our models do not rely on just word polarity to make their predictions and that, even if we do not provide them with an explicit target, they can get a sense of what the main topic of discussion is.

6 Discussion

In this work, we approached a problem that lies at a crossroads between sentiment analysis and stance prediction using Transformer-based models. Our main objective was to explore whether audio-related features could help improve the performance of these models: we thus tried to solve the problem first by using two baseline architectures (respectively, only taking text and audio input) and then using a Multimodal network.

Even if we obtained some improvements by using the Multimodal model concerning the Text model, we think that those are not sufficient to state that spoken features are useful for stance prediction tasks. In any case, by looking at the samples correctly classified by the Multimodal model but not by the Text model, it seems that spoken features could be useful to solve the motion/text mismatch error. We think that further research could be done to create a system able to align the argument extracted from the speech to the motion, by exploiting different spoken features like speaker sentiment or prosody. It could also be interesting to try and see whether more sophisticated strategies to combine text and audio input such as those employed in the Multimodal Transformer architecture (10) can improve our results.

Our error analysis also pointed out how simply removing the last tokens in the text input to abide by the 512 words limit of classical Transformer models could harm performance. Thus, one could try to divide the whole textual transcription into chunks, feed them to the network one by one and then average them or take a majority vote on the outputs to obtain the actual prediction. Another option could be to employ different architectures that do not impose this length limitation like the Longformer (2). A similar point could be made also for the audio input, seeing as only employing the first 10 or 15 seconds of a recording which lasts about 3/4 minutes can be a limiting factor on the performance of our audio model.

Finally, it could be extremely interesting to see how much an explicit reference to the motion (or at least to its positive or negative nature) would impact our results.

References

[1] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman, and Michael Rubinstein. Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. *ACM Transactions on Graphics*, 37(4):1–11, August 2018. arXiv:1804.03619 [cs, eess].

[5] Zheng Fang, Bo Yin, Zehua Du, and Xianqing Huang. Fast environmental sound classification based on resource adaptive convolutional neural network. *Scientific Reports*, 12(1):6599, April 2022.

[6] Rafael Mestre, Razvan Milicin, Stuart Middleton, Matt Ryan, Jiatong Zhu, and Timothy J. Norman. M-Arg: Multimodal Argument Mining Dataset for Political Debates with Audio and Transcripts. In *Proceedings of the 8th Workshop on Argument Mining*, pages 78–88, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[7] Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. Stance and sentiment in tweets. *ACM Trans. Internet Technol.*, 17(3), jun 2017.

[8] Matan Orbach, Yonatan Bilu, Assaf Toledo, Dan Lahav, Michal Jacovi, Ranit Aharonov, and Noam Slonim. Out of the echo chamber: Detecting countering debate speeches. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7073–7086, Online, July 2020. Association for Computational Linguistics.

[9] Thomas F Quatieri. *Discrete-time speech signal processing: principles and practice*. Pearson Education India, 2006.

[10] Wasifur Rahman, Md Kamrul Hasan, Sangwu Lee, AmirAli Bagher Zadeh, Chengfeng Mao, Louis-Philippe Morency, and Ehsan Hoque. Integrating multimodal information in large pretrained transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2359–2369, Online, July 2020. Association for Computational Linguistics.

[11] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[12] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583, 2019.

[13] Yingzhi Wang, Abdelmoumene Boumadane, and Abdelwahab Heba. A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding, 2021.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.