INTERSPEECH 2019: Show & Tell Contribution
September 15–19, 2019, Graz, Austria

CaptionAI: a real-time multilingual captioning application

Nagendra Kumar Goel, Mousmita Sarma, Saikiran Valluri, Dharmeshkumar Agrawal, Steve Braich, Tejendra Singh Kuswah, Zikra Iqbal, Surbhi Chauhan, Raj Karbar

Go-Vivace Inc., McLean, VA, USA

{nagendra.goel, mousmita, valluri, dharmesh, steve.braich}@govivace.com,
{tejendra, zikra.iqbal, surbhi.chauhan, rajk}@govivace.com

Abstract

We demonstrate CaptionAI, the system that can be used for speech to text transcription, multilingual translation, and real-time closed captioning. It can also broadcast the audio and translated text to personal devices. There are three components of the application, namely, speech to text conversion, machine translation, and real time broadcast of audio and its multilingual text transcription. CaptionAI makes meetings, conference, and events accessible to global audience members with its real-time multilingual captioning and broadcast capabilities, improving comprehension and retention. In this application, we support English and Spanish real-time speech transcription. It also supports seventeen popular languages for real-time Machine Translation of transcribed speech. The front-end is coded on c# and in back-end we use combination of python and c++ based software and packages such as Janus, Gstreamer, and libwebsockets.

1. INTRODUCTION

CaptionAI is a windows application which can be used for live closed captioning and translation and broadcast for meetings and conferences. It can also be used in class room lectures, webcasts, and remote classrooms. The application provides a silk-screen allowing a caption overlay on the same computer window where the presentation is being made. The application can also dynamically generate a QR code that can be printed and distributed to the audience or a link that can be shared via email for those who want to participate via their mobile devices. By scanning the QR code, audience can follow the presentation with transcripts and translation in the language of their choice. The number of lines, box width, background color, font color, font size and the font family for closed captions can all be set directly within the application, allowing customization for a broad range of situations.

At the moment, we support two languages for speech to text transcription: English and Spanish, and 17 other languages for translation: Arabic, Croatian, Chinese, Dutch, Farsi, French, German, Hindi, Italian, Polish, Portuguese, Romanian, Russian, Slovenian, Spanish and Turkish. One important feature of this application is its ability to Broadcast the audio, the text and translations, vi a unique web-link, and a QR code generator that encodes the link to an image for mobile device scanning, so that many people can connect for the same live broadcast, and select their choice of language for translations. The unique web-link and the corresponding QR code can be generated in advance so that it can be printed and distributed for an event to others who wish to follow the live streaming at the same time. Some important settings of the application such as Audio input language, translation language, closed caption display language, fonts, colors, size, opacity, location and size of the captions box can all be saved into multiple favourites settings.

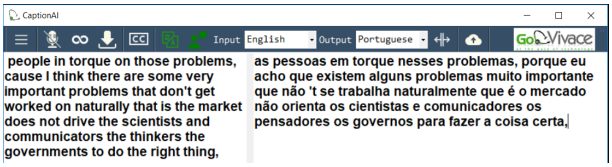

Figure 1: *Application front view*

2. Description of the system front end features

The front-end is a windows based application created using c# programming language. The front-end view the application is shown in Figure 1. In the front-end part of the application, people can do the recording, audio uploading, and loop-back recording for the speech to text transcription via clicking the respective icons. For the machine translation user can select the input language and output language via clicking the translation symbols on tools bar. After clicking translation symbols it shows a list of input and output language selection possibilities. Currently we support two source languages (as ASR input) - English and Spanish. Seventeen other languages can be selected for live translations. For the broadcast presenters can generate their own QR code via clicking the broadcast QR option from the main menu. Users can also save their respective preferences for the application like: font size, font style, background colour of closed captioning, font colour of the caption. window size and position, languages of input and translation etc. into their favourite settings.

3. Description of back end technology

In the back end we use Python, C++, Gstreamer Plugin, Janus gateway and WebRTC. A python Tornedo server based websocket API receives the audio from the front-end application. Gstreamer is used for live conversion of audio file format to 16 bit PCM, and send the audio file to automatic speech recognition server where speech is converted to text. Traceback is performed every half a second to provide partial ASR updates to the Machine Translation server. Translations are updated every half a second. After performing Speech to text decoding and Machine translation, the server sends the results back in the form of JSON structures. For live streaming we use Janus gateway that connects the Gstreamer to WebRTC for broadcasting the real-time generated captions to a web page. In the back-end application there are basically 3 types of components namely, 1) Speech to text transcription, 2) machine translation, 3) broadcasting as shown in Figure 2. Here, the input audio is transcribed in text via speech to text module. Then we pass text to the Machine Translation module which can translate text in

Figure 2: *Internal components of the application*

real-time from English to 17 supported different languages. For broadcast we use Janus gateway to interface Gstreamer and WebRTC in order to broadcast the input audio and output text in both languages.

3.1. Speech to text module

Speech recognition server utilizes Voice Activity Detection to perform real-time Speech activity detection, and to generate speech segments from the arriving audio samples, which are propagated to the real-time Automatic Speech Recognition (ASR) engine. We use the recent technology of Lattice-free MMI based sequence discriminatively [1] trained DNN models for acoustic modeling. Multiple public and private data resources are combined to generate a large amount of English speech training data to train the Deep Neural Network acoustic model. We train an Ivector extractor model from the audio recordings of various speakers present in the training data. The trained model is used for extracting the Ivector features for each speaker, whose speech is included in the acoustic model training data. The Ivector features are appended to the Mel filter-bank features for each speech frame, to generate the input features to the DNN acoustic model, which are used for supervised training of the acoustic model along with the corresponding frame labels. Appending these real-time generated (once in every 100ms) Ivector features to the Mel filter-bank features, as input to the inference acoustic model, has empirically been found to enhance robustness of the ASR system to speaker and accent variabilities in real-time scenarios. For language modeling, we used web crawled data from Wikipedia articles and broadcast newswire text, along with the public text corpora such as fisher english and switchboard transcripts. We obtain close to human accuracy in transcribing English speech for various speakers and noise conditions.

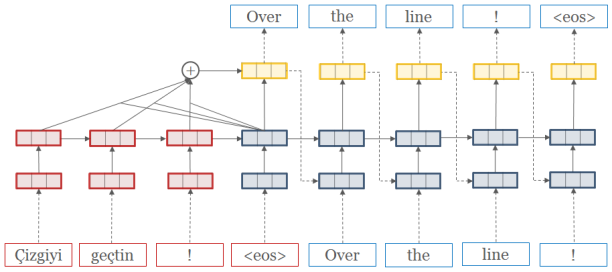

Figure 3: *Neural Machine Translation encoder-decoder architecture (see https://www.tradonline.fr/en/value-of-neural-machine-translations)*

3.2. Machine translation module

We currently support real-time Machine Translation (MT) of the closed caption subtitles into 17 different languages. We use the state-of-the-art Neural Machine Translation [2] system as backend for this module. NMT consists of two main modules - encoder and decoder. The encoder contains Attention mechanism based Recurrent Neural Network, which transmits the encoder state to the decoder at the end of the input sentence. The input sentence of source language is first tokenized, and the embeddings of the word tokens are sequentially fed as input to the encoder RNN. The final encoder state is derived as the Attention mechanism based weighted sum of the hidden layer history states of the encoder RNN. The decoder takes the encoder state and the sequentially generated previous output token as inputs in order to predict the next output token using another Recurrent Neural Network as shown in Figure 3. The sequentially generated output word tokens are then concatenated to form the target language translated sentence for the given input sentence. The output text is further punctuated for ease of human readability.

4. Conclusion and future work

We provide a brief overview of the architecture of our real-time transcription system which generates closed caption subtitles. It also supports the real-time Machine translation of the generated subtitles into 17 other languages. We are working to further improve the Machine Translation accuracy and reliability for some languages, We are exploring the implementation of transformer-based architectures like BERT [3], which have recently proven to largely enhance the BLEU evaluation scores for Machine translation for various language pairs. Supporting speech input in additional languages, and making the User interface more friendly is another focus of future work.

5. References

[1] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, “Purely sequencetrained neural networks for ASR based on lattice-free MMI”, in Proceedings of INTERSPEECH, 2016.

[2] Denny Britz, Anna Goldie, Minh-Thang Luong, Quoc Le, Google Brain team, “Massive Exploration of Neural Machine Translation Architectures”, ACL 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”.