OPEN SOURCE AUTOMATIC LECTURE SUBTITLING

Benjamin Milde[1,2]*, Robert Geislinger*[1]*, Irina Lindt*[1]*, Timo Baumann*[1]

[1]*Language Technology Group, Universität Hamburg*
[2]*Hamburger Informatik Technologie-Center e.V.*
‹surname›@informatik.uni-hamburg.de

Abstract: We present a fully automatic solution for German video subtitling, with a focus on lecture videos. We rely entirely on open source models and scripts for German ASR, automatic punctuation reconstruction and subtitle segmentation. All training scripts, 1000h of German speech training data, pre-trained models and the final subtitling program are publicly available. It can readily be integrated into lecture video platforms such as Lecture2Go. The automatically generated subtitles can also serve as a basis to make the video material more accessible (e.g. via search, keyword clouds, and the like) or for further manual revision, potentially helping in significantly speeding up manual work. A particular challenge that we observe in lectures are technical terms that are frequent in a particular lecture, but infrequent in a typical language model and that might be out of vocabulary for a general purpose ASR. We approach this challenge by extracting texts from accompanying lecture slides to adapt the language model of our TDNN-HMM based ASR system. We demonstrate the usability of the full system and its generated subtitles and evaluate on a dataset of manually transcribed lectures with an average of 26.3% WER.

1 Introduction

Subtitles display the spoken content of a video in written text. They primarily make speech in video accessible to persons with hearing limitations but can be helpful beyond this purpose, e.g. to consume video in silence (or in noisy environments), or as a basis for further text-based tasks such as search, summarization and translation. The European Accessibility Act [1] obligates the public sector to implement accessibility measures. At the same time, remote learning using lecture videos has become the norm during the Covid-19 pandemic in many higher learning institutions. However, video subtitling is a tedious manual task. Even trained transcribers might average 13-18x relative to the speech time without any automatic assistance [2]. As a result, manual subtitling it too costly for many potential uses.

Automatic subtitling primarily requires automatic speech recognition (ASR) decoding on the audio extracted from a video file, however there are further processing steps involved. The output needs to be segmented into appropriate segments and punctuation must be added for better readability. To account for peculiarities in the language used, it may be helpful or even necessary (such as for many lectures) to adapt ASR using textual material. For spontaneously uttered speech, such as lectures, further revision of speech, e.g. wrt. hesitations may be useful.

Our open source subtitling solution Subtitle2Go[1] is a good starting point for all kinds of German subtitling tasks; however, our primary use case is the automatic subtitling of all video lectures published on our university's lecture video platform Lecture2Go [3][2].

[1] Available as open source at `https://github.com/uhh-lt/subtitle2go`.
[2] `https://lecture2go.uni-hamburg.de/`

2 Related work

Respeaking is a popular method to semi-automatically generate subtitles [4, 5, 6]. In these systems, dialogs are respoken by a controlled speaker and decoded with a dictation-oriented ASR software. Speech is recorded in a clear and controlled environment without noise and from a known and professional speaker so that ASR errors are minimized. There is also the possibility to adapt the ASR towards a single speaker, reducing errors further. While this is faster than typing, it is still a tedious manual process.

Fully automatic subtitling systems on the other hand use audio channels extracted from a video directly to generate subtitles. In [7] a fully automatic broadcast subtitling system for several European languages was developed. The system used only closed source components and private datasets. The German ASR system was trained using 150h of data, with a closed source system. In [8], a fully automatic subtitling system for Slovak broadcasting news was developed, also based on the open source toolkit Kaldi for ASR decoding. KIT Lecture Translator [9] is a system that transcribes and translates lectures in real time.

We have previously published and evaluated a Kaldi [10] recipe for general purpose ASR in German based on 1000h of aligned speech data [11] which forms the foundation for the system presented here.

3 Subtitling Pipeline

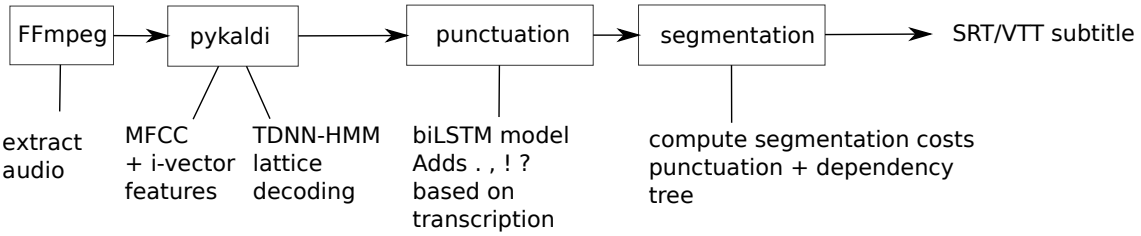

Figure 1 – Subtitling pipeline and processing steps.

Figure 1 shows our subtitling pipeline. We first run FFmpeg/pyFFmpeg to extract 16 kHz pcm mono audio from any supported FFmpeg media file (all popular video formats are supported). We then use feature derivation and decoding with PyKaldi [12], also mapping each word to the alignment information from the decoder. We then apply an LSTM-model that adds punctuation. Finally, we use a segment scoring function and apply beam search to search through different segmentation alternatives. Finally, based on the alignment information and the segment information, we write the complete subtitle file in either SRT or WebVTT format [13].

3.1 Punctuation

Typically, transcriptions from an ASR model lack punctuation, which is added in a separate post processing step. Punctuation aids readability and also helps to segment the subtitles in a subsequent step. We use Punctuator2 [14][3], a model based on bidirectional gated recurrent units (GRUs) attention. The training is based on omitting the original punctuation of a text and predicting it. Our training dataset is built with german-asr-lm-tools[4]: we crawl multiple sources of German texts (ARD subtitles, Tagesschau news articles, speeches of the EU Parliament and Wikipedia) and normalize the data in multiple steps.

[3] `https://github.com/ottokart/punctuator2`

[4] `https://github.com/bmilde/german-asr-lm-tools`

Punctuation	Precision	Recall	F1-score	Error Rate
,COMMA	90.9	87.9	89.4	
.PERIOD	89.2	92.2	90.7	
?QUESTIONMARK	81.5	68.1	74.2	
!EXCLAMATIONMARK	39.2	28.7	33.1	
Overall	89.7	88.4	89.1	
per token				2.08 %
per sentence				16.8 %

Table 1 – Punctuation prediction results on held out data.

At first, structural and meta data like wiki syntax, subtitle timings and XML data is removed. Numbers are converted to words, using the same normalization as for the ASR language model texts (ASR friendly format in subwords for numbers, e.g. 42 → zwei und vierzig), abbreviations are extended to full words (etc. → et cetera). In the next step the raw sentences with special characters (except .,?!) are filtered out. The next step is to tokenize the sentences with spacy[5].

We train the model on 5 million lines of German text and show performance metrics on held out data in Table 1. The model is able to restore the original punctuation correctly in 83.2% of the sentences; as can be seen in Figures 5/6, remaining issues are partially due to other kinds of punctuation (such as dashes).

3.2 Subtitle segmentation

We aim to segment the punctuated transcript to improve readability. The segmentation is based on a beam search with manually tuned weights. We want to balance average segment length with splitting at sentence punctuation (,.?!); all other potential splitting positions in a sentence are evaluated based on the shortest connecting path in a parse tree of that sentence. The intent is that we want to avoid splitting at words that are linguistically closely connected words/tokens, i.e. they should not be separated across two separate screens (or lines) if possible, so that the reading flow is not interrupted. Our beam search maximises a performance function that sums up all individual segmentation decisions. In particular, we defined the performance f_s for sentence s that outputs a reward for segmenting at the position between the tokens t_1 and t_{i+1} as :

$$f_s(t_i, t_{i+1}) = r_l + \begin{cases} 0.9 \cdot \text{len}(s) & \text{if } t_i \in \{\text{`.'}, \text{`?'}, \text{`!'}\} \\ 0.7 \cdot \text{len}(s) & \text{if } t_i = \text{`,'} \\ \text{scp}(\text{parsetree}(s),\, t_i, t_i + 1) & \text{otherwise} \end{cases}$$

where scp is the shortest connecting path in the syntax tree (as parsed with spacy) and r_l the length reward for being as close as possible to the target token length for a particular segment.

$$r_l = 2.3 * (ttl - |ttl - j|)$$

where ttl is the target token length and j the resulting length of the segment, if a split were to be placed between t_i and t_{i+1}. In our experiments we set $ttl = 10$. We then maximize $\sum f_s(t_i, t_{i+1})$ over all chosen segmentation paths, expanding the beam up to a maximum number of lookahead tokens, evaluating each forward position from the current position i up to $i + 40$.

Other constraints could easily be integrated: e.g. a performance function that depends on the length in characters rather than tokens, or adding hard constrains on the maximum character length for a subtitle segment. Another improvement could be to include pausing information into the performance measure, or to differentiate between line and screen breaks.

[5] `https://spacy.io/`

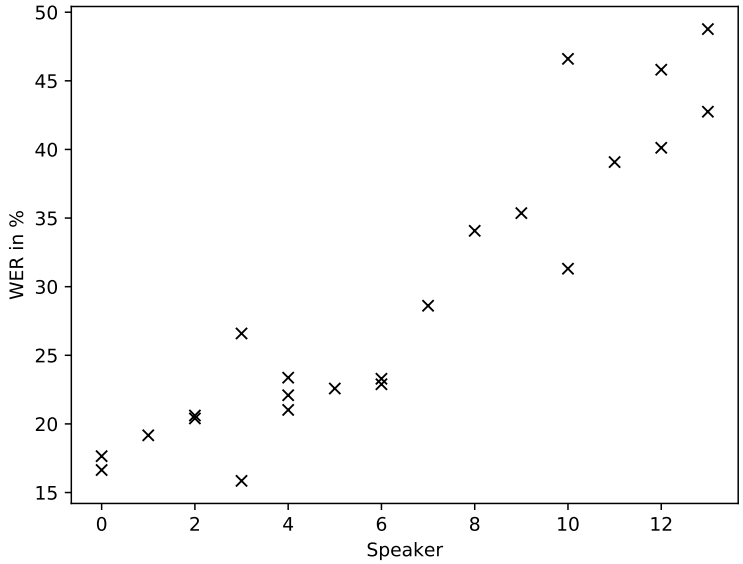

Figure 2 – WER for 17 German lectures, with topics ranging from education science to computer science. The 14 anonymised speakers are sorted according to their average WER.

4 Evaluation

Figure 2 shows word error rates (WER) evaluated on 17 Universität Hamburg (UHH) lectures from 2009 to 2014 that were manually transcribed, totalling 12h of video recordings with subtitles. The average WER over all speakers and lectures on this test set is 26.3. Many of the recorded lectures have challenging acoustics, as they were recorded in addition to students attending the lecture hall. We also tested Subtitle2Go on one newer video, a public speech of Dr. Frank-Walter Steinmeier from 2019, which yields a similar WER of 25.9.

4.1 Examples

```
1 Und diese Datenbank wollen wir jetzt im selben im Prolog System verarbeiten.
2 Und was wir dafür brauchen Sie ist ein Syntax Konstrukt.
3 Das Syntax Konstrukt sieht etwas anders aus, als wir das
4 von SQL etwa gewöhnt sind. Eben. Es ist aber
5 auch die Sache sehr stark kondensiert und auf den Kern gebracht.
6 Während also das Modell der relationalen Datenbanken in der Datenbank Welt
7 davon ausgeht, dass jedes Attribut einen Namen hat.
8 Also jede Tabellenspalten, einen Namen hat das an der im gesehen?
9 Ja, hier oben drüber steht ihr. Bei der Attribut Name
10 Machen wir jetzt die Vereinfachung, dass wir den Attributen am weggelassen,
11 dass es wieder eine Abstraktion, die wir machen.
12 Und wir ordnen die Information über die Positionen im Innen
13 in einem Tupel zu. Dass wir also wissen die erste
14 Argument Stelle ist die UNK der Identity Feier.
15 Die zweite Argument Stelle ist der ist das Einfamilienhaus und so weiter und so fort.
16 Der Unterschied besteht jetzt darin, dass dies ein geordnetes Tupel ist.
```

Figure 3 – Automatic output of our system for a section starting at 21:39 of "Relationale Datenbanken" from Prof. Dr.-Ing. Wolfgang Menzel, 24.10.2013, "Softwareentwicklung III: Logikprogrammierung (WiSe 13/14). URL: `https://lecture2go.uni-hamburg.de/l2go/-/get/v/15458`

In the following, we list two example outputs of our system, alongside manual subtitles that were available for these videos. Both videos are publicly available on the Lecture2Go platform. In Figure 3 and 4 we compare a snippet from a typical computer science lecture. Some key

```
1 und diese Datenbank
2 wollen wir jetzt eben
3 Prologsystem verarbeiten
4 und was wir dafür brauchen ist ein Syntaxkonstrukt
5 ja das Syntaxkonstrukt
6 sieht etwas anders aus als wir das von SQL etwa gewöhnt sind
7 es ist aber auch
8 die Sache sehr stark kondensiert und auf den Kern gebracht während also
9 das Modell der realtionalen Datenbanken
10 in der Datenbankwelt
11 davon ausgeht dass jedes Attribut einen Namen hat
12 also jede Tabellenspalte einen Namen hat das haben wir eben gesehen
13 ja hier oben drüber steht immer der Attributname
14 machen wir jetzt die Vereinfachung dass wir den Attributnamen weg lassen das ist wieder
eine Abstraktion die wir machen
15 und wir ordnen die Information über die
16 Positionen im
17 in einem Tupel zu das wir eben wissen die erste Argumentstelle ist
18 die der Identifier die zweite Argumentstelle ist der
19 ist das Einfamilienhaus und so weiter und so fort
20 der Unterschied besteht jetzt darin, dass dies ein geordnetes Tupel ist
```

Figure 4 – Manual subtitles for a section starting at 21:39 of "Relationale Datenbanken" from Prof. Dr.-Ing. Wolfgang Menzel, 24.10.2013, "Softwareentwicklung III: Logikprogrammierung (WiSe 13/14). URL: `https://lecture2go.uni-hamburg.de/l2go/-/get/v/15458`

```
1 Aber die Hochschule ist kein Schonraum und keinen Spielplatz,
2 was Sie eben von der lebensgeschichtliche weit früher liegenden Bildungsstätte des
Kindergartens unterscheidet.
3 Wer als Professor oder als Student glaub verhindern zu müssen,
4 das unorthodoxe wissenschaftliche Thesen zu Wort kommen.
5 Wer glaubt, Bücher mit kontroversen Inhalten sollten aus den Bibliotheken verschwinden.
6 Das gibt es tatsächlich auch wieder solche Ansichten,
7 der hantiert aus dem Inneren der Wissenschaft mit dem gleichen tödlichen Gift.
8 Weder fundamentalistische blinde melden aus dem Namen der Rose.
9 Kurzum, Forschung und Lehre müssen frei sein.
10 Diese unersetzliche Freiheit zu achten und nicht zu missbrauchen, ist Aufgabe aller.
```

Figure 5 – Automatic output of our system for a section starting at 10:33 of "Rede des Bundespräsidenten auf der HRK", Dr. Frank-Walter Steinmeier, 18.11.2019. URL: (`https://lecture2go.uni-hamburg.de/l2go/-/get/v/25345`).

```
1 Aber die Hochschule ist kein Schonraum und kein Spielplatz,
2 was sie von der lebensgeschichtlich weit früher liegenden Bildungsstätte
3 des Kindergartens unterscheidet.
4 Wer - als Professorin oder als Student - glaubt, verhindern zu müssen,
5 dass unorthodoxe wissenschaftliche Thesen zu Wort kommen,
6 wer glaubt, Bücher mit kontroversen Inhalten sollten aus den Bibliotheken verschwinden
7 - und es gibt tatsächlich wieder solche Ansichten -,
8 der hantiert aus dem Inneren der Wissenschaft mit dem gleichen tödlichen Gift
9 wie der fundamentalistische blinde Mönch aus dem Namen der Rose.
10 Kurzum:
11 Forschung und Lehre müssen frei sein!
12 Diese unersetzliche Freiheit zu achten und nicht zu missbrauchen,
13 ist Aufgabe aller.
```

Figure 6 – Manual subtitles for a section starting at 10:33 of "Rede des Bundespräsidenten auf der HRK", Dr. Frank-Walter Steinmeier, 18.11.2019. URL: (`https://lecture2go.uni-hamburg.de/l2go/-/get/v/25345`).

words, especially anglicisms such as "Identifier" are misrecognized ("Identity Feier"). In Figure 5 and Figure 6 we compare a speech from president Dr. Frank-Walter Steinmeier with a more broad topic. The quality of the automatically generated subtitles is better in this example. We note some issues that inflate our WER measures, e.g. short interjections being found by our system (‘Eben.’), and typos in the manual subtitles (‘realtionalen’).

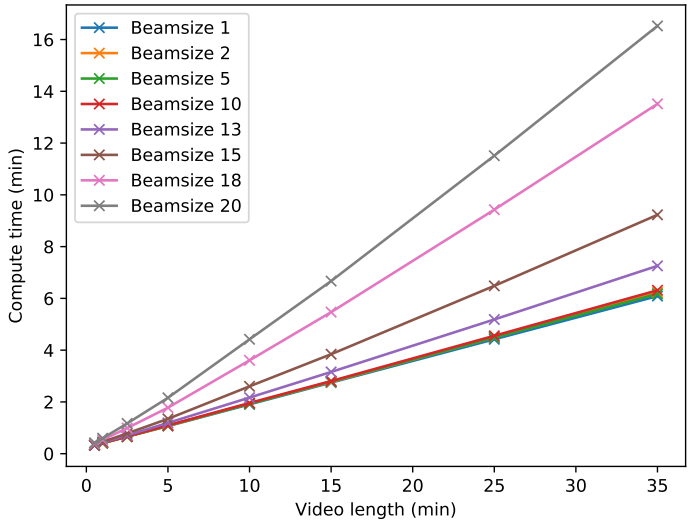

Figure 7 – Computation time needed for all processing steps for different video and beam sizes. Timings were measured on one core of an Intel server CPU (Xeon E5-2620 v4).

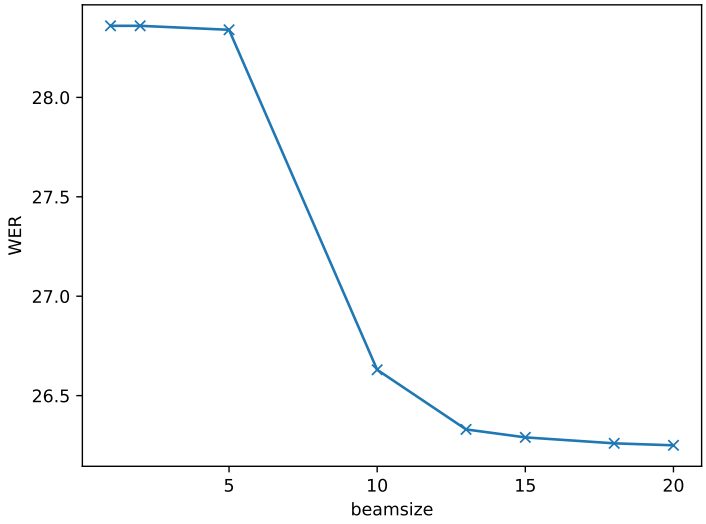

Figure 8 – WER on the Lecture2Go test set, depending on the beamsize.

4.2 Decoding speed

Decoding speed depends on many factors and for some there is a trade-off between speed and accuracy. In the following we report results for tuning the beam size of the ASR decoder with the default model size of kaldi-tuda-de (3.5GB), while using the default options of Subtitle2Go otherwise. In Figure 7 we plot the complete computation time needed for a particular video length, including startup and loading times. The run times were measured on an Intel Xeon E5-2620 v4 CPU (2.10GHz) using one core, with Kaldi linked against Intel MKL. As can be seen in the figure, compute time grows linearly with video duration. In Figure 8 we show WER on the Lecture2Go test set with the same beam sizes. Since there is little WER improvement for beam sizes larger than 13, but computation time increases considerably, we settle on 13 as the default. Using this setting a 35 minute video took 7m16s of computation time for all processing steps. This yields a real-time factor of 0.2 and a full 90-minute lecture could be subtitled in 20 minutes (i.e., before the next lecture period starts). While there is no parallelization beyond the

vectorization of the Intel MKL BLAS operations, multiple videos can be decoded at the same time with multiple cores.

5 ASR model adaptation

We identified OOV to be an issue in some lectures, that may use many technical terms or loan words. While the 683,000 word vocabulary of the generic ASR model covers a wide range of diverse topics, some technical terms will still be missing from this predefined vocabulary and can not be recognized at all. Additionally lecture-specific information should also yield better word transition probabilities.

For more generic terms and terms where we expect automatic grapheme-to-phoneme (G2P) conversion to fail with phonetic lexicon entries (e.g. anglicisms in German, see [15]), we use speech-lex-edit[6], a lexicon editor with active learning and text-to-speech (TTS) feedback. With texts from the target domain, we can rank OOVs by their frequency, as well as the G2P model confidence values, in order to focus the manual work on phonetization. Furthermore we use pdfplumber[7] to extract text snippets from accompanying lecture slides. These are added to the language model, as well as to generate automatic phonetic lexicon entries with G2P. Our process for adaptation is not yet fully automated. We gather snippets from multiple lecture PDFs and recompute the LM and the Kaldi FST (keeping the acoustic model), which requires several hours of computation time. We hope to extend this paradigm into a more dynamic process, where ideally the model can be adapted much quicker and also be adapted on a per-lecture basis.

We adapted our model with text snippets from 200 PDFs from various lectures. Unfortunately there were only three videos in our test set where PDF slides were available. For one video we could reduce the OOV rate significantly from 3.1% to 1.8%, also reducing WER from 45.03 to 43.98. However, on the whole test set OOV was only slightly reduced from 3.02% to 2.99%. Also overall WER increased from 26.33 to 28.82 with the adapted model.

6 Conclusion

We presented the fully open-source subtitling solution Subtitle2Go, and evaluated its performance on German lecture videos. In many cases, we obtain good and useful subtitles. Depending on the quality of the recorded audio, the subtitles can be used as is on a video platform or serve as a basis for further manual editing. While our intermediate goal was to support automatic subtitles on the Lecture2Go platform, with predominately German content, extending support to additional languages is straightforward. For many languages, Kaldi nnet3 models can be readily downloaded and/or training recipes exist. Spacy supports syntax parsing for 16 languages with pre-trained models. If both a Kaldi model and syntax parsing are available for a language, the only additional component required is punctuation reconstruction. This can be trained with crawled text and does not need further annotation. Additionally, a text normalization library or a TTS frontend are helpful, if available.

We currently rely on the ASR model to disregard non-speech, which works sufficiently well for lecture recordings. A full subtitling system might need to include diarization and/or voice activity detection, to improve robustness when non-speech segments are present and speakers change frequently (e.g., for automatic subtitling of movies).

Acknowledgements. We would like to thank the Lecture2Go team at Universität Hamburg for the collaboration, their support and testing of our prototype, as well as the ongoing integration. We also thank the student project on subtitling lecture videos in 2016 for their testing of different strategies to break long subtitles.

[6] `https://github.com/uhh-lt/speech-lex-edit`

[7] `https://github.com/jsvine/pdfplumber`

References

[1] COUNCIL OF EUROPEAN UNION: *Directive (EU) no 2019/882*. 2019. https://eur-lex.europa.eu/eli/dir/2019/882/oj.

[2] ROY, B. C. and D. K. ROY: *Fast transcription of unstructured audio recordings*. In *Proceedings of Interspeech 2009*, pp. 1647–1650. Brighton, UK, 2009.

[3] KRISZAT, M., I. STURM, and J. T. CLAUSSEN: *Lecture2go – von der vorlesungsaufzeichnung ins world wide web*. In *Digitale Medien für Lehre und Forschung*, pp. 25–38. 2010. URL http://nbn-resolving.org/urn:nbn:de:0111-pedocs-173167.

[4] SPERBER, M., G. NEUBIG, C. FÜGEN, S. NAKAMURA, and A. WAIBEL: *Efficient speech transcription through respeaking.* In *Proceedings of Interspeech 2013*, pp. 1087–1091. Lyon, France, 2013.

[5] ROMERO-FRESCO, P.: *Subtitling through speech recognition: Respeaking*. Routledge, 2020.

[6] VASHISTHA, A., P. SETHI, and R. ANDERSON: *Respeak: A voice-based, crowd-powered speech transcription system*. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pp. 1855–1866. Denver, Colorado, USA, 2017.

[7] ÁLVAREZ, A., C. MENDES, M. RAFFAELLI, T. LUÍS, S. PAULO, N. PICCININI, H. ARZELUS, J. NETO, C. ALIPRANDI, and A. DEL POZO: *Automating live and batch subtitling of multimedia contents for several european languages*. *Multimedia Tools and Applications*, 75(18), pp. 10823–10853, 2016.

[8] LOJKA, M., P. VISZLAY, J. STAŠ, D. HLÁDEK, and J. JUHÁR: *Slovak broadcast news speech recognition and transcription system*. In *International Conference on Network-Based Information Systems (NBiS-2018)*, pp. 385–394. Bratislava, Slovakia, 2018.

[9] DESSLOCH, F., T.-L. HA, M. MÜLLER, J. NIEHUES, T.-S. NGUYEN, N.-Q. PHAM, E. SALESKY, M. SPERBER, S. STÜKER, T. ZENKEL, and A. WAIBEL: *KIT lecture translator: Multilingual speech translation with one-shot learning*. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pp. 89–93. Santa Fe, New Mexico, USA, 2018.

[10] POVEY, D., A. GHOSHAL, G. BOULIANNE, L. BURGET, O. GLEMBEK, N. GOEL, M. HANNEMANN, P. MOTLICEK, Y. QIAN, P. SCHWARZ, J. SILOVSKY, G. STEMMER, and K. VESELY: *The kaldi speech recognition toolkit*. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011. IEEE Catalog No.: CFP11SRW-USB.

[11] MILDE, B. and A. KÖHN: *Open source automatic speech recognition for german*. In *Proceedings of ITG 2018*. Oldenburg, 2018.

[12] CAN, D., V. R. MARTINEZ, P. PAPADOPOULOS, and S. S. NARAYANAN: *Pykaldi: A python wrapper for kaldi*. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5889–5893. Calgary, Alberta, Canada, 2018.

[13] PFEIFFER, S.: *WebVTT: The web video text tracks format*. Candidate recommendation, W3C, 2019. URL https://www.w3.org/TR/2019/CR-webvtt1-20190404/.

[14] TILK, O. and T. ALUMÄE: *Bidirectional recurrent neural network with attention mechanism for punctuation restoration*. In *Proceedings of Interspeech 2016*, pp. 3047–3051. San Francisco, California, USA, 2016.

[15] MILDE, B., C. SCHMIDT, and J. KÖHLER: *Multitask sequence-to-sequence models for grapheme-to-phoneme conversion*. In *Proceedings of Interspeech 2017*, pp. 2536–2540. Stockholm, Sweden, 2017.