MeMAD

Methods for Managing Audiovisual Data

memad.eu
info@memad.eu

Twitter – @memadproject
LinkedIn – MeMAD Project

MeMAD Deliverable

D2.3 Software and demonstration of human-like content description generation

Grant agreement number	780069
Action acronym	MeMAD
Action title	Methods for Managing Audiovisual Data: Combining Automatic Efficiency with Human Accuracy
Funding scheme	H2020–ICT–2016–2017/H2020–ICT–2017–1
Version date of the Annex I against which the assessment will be made	23.6.2020
Start date of the project	1.1.2018
Due date of the deliverable	30.9.2020
Actual date of submission	25.11.2020
Lead beneficiary for the deliverable	Aalto University
Dissemination level of the deliverable	Public

Action coordinator's scientific representative
Prof. Mikko Kurimo
AALTO–KORKEAKOULUSÄÄTIÖ, Aalto University School of Electrical Engineering, Department of Signal Processing and Acoustics
mikko.kurimo@aalto.fi

MeMAD project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780069. This document has been produced by theMeMAD project. The content in this document represents the views of the authors, and the European Commission has no liability in respect of the content.

Authors in alphabetical order		
Name	Beneficiary	e-mail
David Doukhan	INA	ddoukhan@ina.fr
Zixin Guo	AALTO	zixin.guo@aalto.fi
Ismail Harrando	EURECOM	ismail.harrando@eurecom.fr
Mikko Kurimo	AALTO	mikko.kurimo@aalto.fi
Jorma Laaksonen	AALTO	jorma.laaksonen@aalto.fi
Matias Lindgren	AALTO	matias.lindgren@aalto.fi
Tiina Lindh-Knuutila	LLS	tiina.lindh-knuutila@lingsoft.fi
Pasquale Lisena	EURECOM	pasquale.lisena@eurecom.fr
Selen Pehlivan Tort	AALTO	selen.pehlivantort@aalto.fi
Alison Reboud	EURECOM	alison.reboud@eurecom.fr
Aku Rouhe	AALTO	aku.rouhe@aalto.fi
Raphaël Troncy	EURECOM	raphael.troncy@eurecom.fr
Anja Virkkunen	AALTO	anja.virkkunen@aalto.fi

Internal reviewers in alphabetical order		
Name	Beneficiary	e-mail
Maija Hirvonen	UH	maija.hirvonen@helsinki.fi
Umut Sulubacak	UH	umut.sulubacak@helsinki.fi

Abstract

This deliverable describes the last development iteration of the joint collection of libraries and tools for multimodal content analysis and description from AALTO, EURECOM, INA, Lingsoft, LLS and Limecraft. The methods generate unimodal descriptions of video contents in the aural and visual domains. These descriptions can be used as such and also combined to describe the contents in a multimodal fashion as enriched video captions. The ultimate goal of the development is to be able to generate human-like content descriptions, in which the persons, objects, actions and environments, both seen and heard, are described in a sophisticated way. In this deliverable, a number of improvements we have made in that direction are elaborated. We describe a small-scale dataset used in the experiments and demonstrate and evaluate various multimodal combinations of the unimodal inputs with that data. As part of this deliverable, the existing open source components gathered into a joint software collection of tools and libraries have been updated and new components have been added. Finally, the abstracts of academic theses together with full texts of scientific publications appear at the end of the report.

Contents

1 Introduction

The goal of the MeMAD project's Work Package WP2 *Automatic multimodal content analysis* is to develop the tools and libraries of AALTO, EURECOM, INA, Lingsoft, LLS and Limecraft for multimodal analysis, description and indexing of audio and video content. The last task of WP2, Task T2.3 *Generating human-like narrative media content description* aims at the development of multi-layered models that can generate *human-like* descriptions for videos, taking into account both the visual and aural domains. The ultimate goal of such development is to devise computer algorithms that are able to generate human-like content descriptions, in which the persons, objects, actions and environments, both seen and heard, are described in a sophisticated way. In this deliverable, a number of improvements we have made in that direction are elaborated and evaluated.

This deliverable presents the results of MeMAD's Task T2.3 and the final development iteration of the MeMAD partners' joint collection of libraries and tools which have been developed and extended further during the last year of the project. One important novel development described in this deliverable is the model for identification of the spoken languages. Due to the multilingual contents in MeMAD's data collections and requirements of media asset management systems in general, including such a functionality in MeMAD's multimodal and multilingual media analysis framework was considered necessary. New functionalities have been added also in the postprocessing techniques that are used to combine the automatically generated video captions with aural and facial gender and person identification.

First, we briefly describe in Section 2 all aural inputs – speaker diarisation, spoken language identification, speech recognition, gender identification and aural environment recognition – that have been extracted and used. Next, Section 3 similarly describes the visual inputs, namely face recognition, visual captioning and visual environment detection. Section 4 describes the test dataset selected from the MeMAD partners' data and used in the small-scale multimodal experiments and demonstrations of this deliverable. Section 5 addresses the postprocessing techniques used for correcting and enriching the automatically generated video captions with information about the environments, person names, genders and spoken languages that are obtained from the other inputs. The speech recognition results have not been used in these enrichments, but they have been made available as subtitles and used as inputs for, e.g., named entity recognition in WP3 and machine translation in WP4. Furthermore, Section 5 presents the results when these techniques are experimented with the test data described in the previous section. Section 6 discusses the obtained results, their applicability and future prospects.

Finally, in Section 7, we include an updated summary of the components and the open source collection of software that form the primary contents of this deliverable. At the end of this report, in Appendix A, we have included a set of theses and scientific publications or their drafts that describe the technological advances made in the project.

This report accompanies the software components stored in a GitHub repository with brief descriptions and evaluations of their use. The address of the GitHub repository is:

`https://github.com/MeMAD-project/mmca`

After this deliverable, Deliverables D3.3 *TV moments detection and linking, final version* and D6.9 *Evaluation report, final version* will report results that are closely related to the work carried out in WP2 and Task T2.3. These include work on video summarisation in D3.3 and work on semantic or topical video segmentation in D6.9. From the perspective of WP2, these developments can be considered applications and extensions of the work carried out in WP2.

2 Aural inputs

Two fundamentally different types of aural inputs have been used to extract information and to describe the contents of the videos: The first one is the human voice that has been used for speaker diarisation, speech recognition and gender identification. The second are non-human sounds that can be used to identify audio events and to recognise environments.

2.1 Speaker diarisation

Speaker diarisation refers to segmenting the audio into distinct segments and applying a speaker label to each of them. It can be then combined with speech recognition — augmenting and improving the speech recognition result by, for example, adding sentence borders at speaker changes. In fast multi-speaker conversation, sentence boundaries cannot always be automatically detected, hence recognizing speaker turns makes the speech recognition output more understandable. In addition, to make the task more difficult, in spoken language people do not usually speak in full-formed sentences but 'utterances' delineated by pauses or other breaks.

In spoken language identification, speaker diarisation is a useful starting point. Speaker diarisation provides a clear segmentation of audio into meaningful chunks, on which language identification can then be performed. While bilingual people may switch languages even within a sentence, expecting a single language for a single speaker turn is a reasonable assumption in practice.

Considerable improvements for the speaker diarisation were achieved in August 2020 by Lingsoft by creating a new custom diarisation model. The model is based on sparse auto-tuning spectral clustering by using normalized maximum eigencap (ASC-NME) [4][1] which improved performance considerably as shown in Table 1. This new diarisation model replaces the one with Variational Bayes (VB) resegmentation [2], which provided the best results earlier. The VB model was computationally intensive, and thus less suitable for production use than the new ASC-NME model. The improved diarisation results are immediately available for use to the MeMAD consortium via the Lingsoft Speech Service API.

The diarisation seems to work really well for multiple speakers in our 10-hour test set. Further investigation with a variety of different programming revealed, though, that the noise robustness still requires work. Especially music in the background easily distorts the speaker diarisation result by causing a too-large number of speakers recognized.

Diarisation error rates (DER)

Language	no VB	with VB	ASC-NME
Finnish	23.37	20.07	**5.87**
Swedish	27.23	29.34	**9.99**

Table 1: Improved diarisation error rates (DER), lower is better, for Finnish and Swedish with the multilingual multispeaker conversational 10-hour Yle media dataset described in MeMAD Deliverable D2.1. VB stands for Variational Bayes resegmentation that was experimented with in MeMAD Deliverable D2.2. The ASC-NME results have been achieved in August 2020.

[1] The custom model is based on `https://github.com/tango4j/Auto-Tuning-Spectral-Clustering` open source contributions.

[2] Variational Bayes resegmentation model was based on https://speech.fit.vutbr.cz/software/vb-diarization-eigenvoice-and-hmm-priors

2.2 Spoken language identification

Originally, language identification was not part of the MeMAD work plan. However, most of the language processing tools, such as automatic speech recognition (ASR), machine translation (MT) and named entity recognition (NER) are language-dependent in the way that, to select the right tool, the language must first be known. During the project it was realized that, in general, language identification is not provided in the metadata of the broadcast material, or, even worse, the same program can include multilingual material where the language is changing abruptly. Even if the main language could be correctly guessed, the small parts in which the language suddenly changes can cause speech recognition to provide completely incorrect text. Although those parts are often clearly visible for a human viewer, a reliable automatic detection is not easy. Such erroneous speech recognition results also propagate errors further downstream in the process, as ASR in wrong language often produces a large number of name-resembling items that get wrongly recognized as named entities. Thus, we realized that language identification must be included in the multimodal data processing pipeline.

Spoken language identification (SLID) is the task of identifying the language of a spoken utterance. The task is considerably more difficult than language identification from text, because there is no reliable automatic mapping to any symbolic representation such as letters or phonemes. Basically, to use ASR one would first need to know the language.

Although several approaches to deep learning based end-to-end SLID have been proposed, there is no easy, unified way to train and compare several SLID models. To find the best model and develop it further we implemented a software toolkit built on the popular TensorFlow deep learning framework[3] for easier end-to-end training of deep learning based SLID models across several speech datasets. We tested it for three SLID benchmark datasets including Oriental Language Recognition challenge 2019 (AP19-OLR) [5], Multi-Genre Broadcast challenge (MGB-3) [6] and Dataset of Slavic Languages (DoSL) [7].

The toolkit was applied to implement one baseline model for each of the SLID benchmarks, one model that AALTO had developed for speaker recognition now applied for SLID, and three x-vector architecture variations. The choice of baselines for MGB-3 and DoSL was motivated by the success previously reported [6], [7] for these two datasets. The baseline architecture for AP19-OLR was defined as the competition baseline by [5] and we chose the same baseline. Our results presented in [1] included all of these seven model architectures:

1. The regular x-vector [8] approach, but TDNN layers replaced with temporal convolution layers as in [9]. This is our baseline model for AP19-OLR. In addition, this is the baseline x-vector architecture for creating the variations, i.e. models 5, 6, and 7.
2. 1D CNNs with average pooling over the time dimension and three FC layers [9]. This is our baseline model for MGB-3.
3. Two bi-directional gated recurrent units (BGRU) and three FC layers [7]. This is our baseline model for DoSL.
4. SphereSpeaker architecture, that has recently been successful for speaker recognition [10], now applied to SLID.
5. Model 1 with increased robustness by applying channel dropout on input during training, using probability 0.5 [11].
6. Model 1 extended with temporal convolution layers before the statistics pooling layer as in [12, Table 3].
7. Model 1 with a small 2D CNN front-end for gathering frequency information [13].

[3] `https://tensorflow.org`

In our Interspeech 2020 paper [1] (see Appendix A.2) all seven models were trained on all three SLID benchmark datasets. All models were trained both separately on each dataset (closed task) and on a combination of all datasets (open task), after which we compared if the open task training yielded better language embeddings. We began by training all models end-to-end as discriminative classifiers of spectral features, labeled by language. Then, we extracted language embedding vectors from the trained end-to-end models, trained separate Gaussian Naïve Bayes classifiers on the vectors, and compared which model provides best language embeddings for the back-end classifier. Our experiments show that the open task condition led to improved language identification performance on only one of the benchmark datasets. In addition, we discovered that increasing x-vector model robustness with random frequency channel dropout as in our model 5 significantly reduced its end-to-end classification performance on the test set, while not affecting back-end classification performance of its embeddings. Finally, we noted that two baseline models consistently outperformed all other models: model 1 outperforms other models on AP19-OLR and DoSL, while model 2 is best on its reference dataset MGB-3 and the best overall model on average.

All results were published in Interspeech 2020 [1] and Matias Lindgren's Master's Thesis [14] (see Appendix A.1). The new end-to-end SLID toolkit for running multiple SLID experiments on multiple datasets was released as free open source software[4]. The toolkit was used to implement seven existing SLID architectures and run experiments on three SLID datasets. We implemented also the SphereSpeaker speaker recognition architecture developed earlier at AALTO [10] (MeMAD Deliverable D2.2) on the new toolkit and applied it to SLID. In addition, we published[5] the configuration files, dataset metadata, and scripts for all the experiments discussed in [1].

AALTO has collected a large new SLID training dataset called YTN-Aalto2019[6] that contains almost 1200 hours in six languages. It was collected from YouTube news channels and labeled weakly with the assumption that each news channel contains only speech in a single language. The metadata and collection scripts are freely available, but access to the videos depends on their producers. The YTN-Aalto2019 dataset was used to develop a real-time SLID demo[7] based on the x-vector architecture. It has been implemented in such a way that it can be executed in a web browser taking audio input in real-time from the user's microphone. The demo shows how difficult it is to reliably detect the spoken language from short time windows (here two seconds). More work and training data would be required to make the segments more coherent for practical use. This demo does not try to model the language transitions which vary a lot depending on the target data. Nor does it separate speech and noise regions to produce a smooth output. However, a new model was later trained using more data from totally 67 languages for the x-vector language embeddings. These embeddings were then used by a classifier for five MeMAD languages (Finnish, Swedish, English, German and French) and for samples with no detected language (mostly background music). These five languages were chosen because they appear in the programs used for the evaluation. A class for samples with no language was also included, since the segmentation using speaker diarisation produces segments with no speech.

2.3 Speech recognition

Notable improvements in Lingsoft's speech recognition for the Finnish and Swedish languages were already demonstrated in the revised version of MeMAD Deliverable D2.2. In that de-

[4] https://github.com/py-lidbox/lidbox
[5] https://github.com/py-lidbox/interspeech-2020-lidbox
[6] http://urn.fi/urn:nbn:fi:lb-2020041701
[7] https://youtu.be/cyL4UkVh_oQ

liverable it was also shown that with respect to the word error rate (WER) measures, the results were the state of the art when compared with the performance of the Google Speech Recognition service.

At AALTO, seeking for improvements for the state-of-the-art ASR, we explored deep transformer architectures BERT and Transformer-XL as a language model for a Finnish ASR task with different rescoring schemes. Recently, these BERT and Transformer-XL based architectures have achieved strong results in a range of NLP applications, but not so much as language modeling in ASR, because of their computational complexity for evaluating a very large number of candidate sentences. In [2] (Appendix A.3 of this Deliverable) we published strong results in both an intrinsic and an extrinsic task with Transformer-XL. We achieved 29% better perplexity and 3% better WER than our previous best LSTM-based approach [15] (in MeMAD Deliverable D2.1). We also introduced a novel three-pass decoding scheme which improved the ASR performance by 8%. To the best of our knowledge, this is also the first work 1) to formulate an alpha smoothing framework to use the non-autoregressive BERT language model for an ASR task, and 2) to explore subword units with Transformer-XL for an agglutinative language like Finnish. AALTO's new ASR results were published in Interspeech 2020 [2] and in Abhilash Jain's Master's Thesis [16] (see Appendix A.1).

The evaluation data chosen for these first tests reported above were an older Yle News test set and the identical acoustic model and language model training data as used in our earlier best results [15] for a fair comparison. Next, they will be tested also with the new MeMAD ASR test set prepared in MeMAD Deliverable D2.2. At the same time, we plan also to update the acoustic and language model training data to the bigger ones currently used at AALTO.

2.4 Aural gender identification

INA's SpeechSegmenter is a CNN-based audio segmentation toolkit. It splits audio signals into homogeneous zones of speech, music and noise. Speech zones are split into segments tagged using speaker gender (male or female). Male and female classification models are optimized for French language since they were trained using French speakers (acoustic correlates of speaker gender are language-dependent). Audio segments corresponding to speech over music or speech over noise are tagged as speech. SpeechSegmenter has been continuously updated since MeMAD Deliverable D2.1 and its current version is available as part of MeMAD's GitHub repository.

2.5 Audio event and environment recognition

MeMAD's Deliverable D2.1 already included the *AudioTagger* software [17] which was used to describe the aural contents video programs with 527 audio tags derived from the annotated AudioSet data by Google Research [18]. The full AudioSet data consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds.

In MeMAD Task T2.2, the audio tagging software was integrated more tightly in the PicSOM framework and can now be widely used for audio feature extraction from videos and, consequently, also for content description including caption generation with the *DeepCaption* library. In Task T2.3, a subset of the available audio events have been used to recognise the audio background or aural environment of the video content. This recognition can be used as such or combined with the automatically generated video captions, as will be described in Section 5.5.

3 Visual inputs

Visual inputs have been used for three different purposes as will be described in the following sections. First, the identities of persons visible in the video programs have been obtained with face recognition. Second, automatic video content descriptions have been generated with deep learning based captioning. Third, the visual location or environment of the videos has been obtained with scene recognition.

3.1 Face recognition

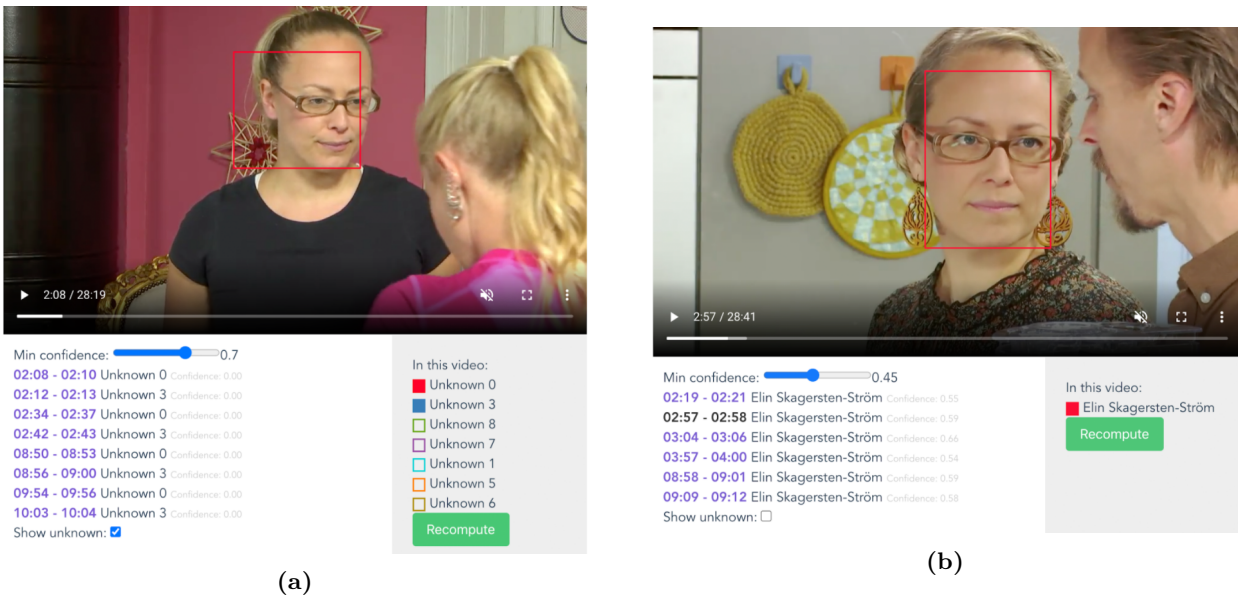
With the aim of extending EURECOM *FaceRec* face recognition system to recognising a larger set of faces, it was necessary to improve the scalability of the classification. In order to do so, the single multi-class classifier used in MeMAD's Deliverable D2.2 has now been replaced with a set of N binary classifiers, where N is the number of distinct individuals to recognise. Each of the N classifiers has been trained in a one-against-all approach [19], in which the facial images of the selected individual are used as positive samples, while all the others are considered negative samples. In this way, each classifier produces as its output a confidence value that is independent of the outputs of all other classifiers. This allows to set a confidence threshold for the candidate identities which does not depend on N. This new version of the software has slightly improved the average precision by 0.04 to the level of 0.67 while simultaneously improving the average recall considerably by 0.22 to the level of 0.93.

In addition to the earlier pipeline based on images crawled from the web as described in MeMAD Deliverable D2.2, a face clustering algorithm has been integrated in *FaceRec* in order to detect non-celebrities or any persons one cannot name in advance. At runtime, the FaceNet features [20] extracted from faces in video frames are collected. Once the video has been fully processed, these features are aggregated through hierarchical clustering based on a distance threshold that has been set empirically, producing a variable number K of clusters. Those clusters are filtered in order to exclude 1) those for which we can already assign a name from our training set, 2) those having a distance — computed as the average distance of the elements from the centroid — larger than a second, more strict threshold, and 3) those having instances of face profile views in the center of the cluster. In particular, we empirically observed that the latter case produces unreliable results by grouping profile views of different people.

For understanding the benefit that results from the face clustering, we include in Figure 1 an example use case. In Figure 1a, the clustering algorithm identified a set of unknown people, among which *Unknown 0* happens to be Elin Skagersten-Ström, who was not part of our training set. We extracted four frames from the centers of each segment in which *Unknown 0* appeared and included those images in the training set. Retraining the classifier with this new data, it was possible to correctly detect Elin Skagersten-Ström in other videos, as seen in Figure 1b. This approach can be applied to any individuals, including those for whom one cannot find enough face images in the web for training a classifier. In the near future, the visualiser of EURECOM *FaceRec* will include the possibility for domain experts to assign a label to those faces and automatically include them in the training set.

3.2 Visual captioning

AALTO's PicSOM team participated in TRECVID 2020 Video-to-Text (VTT) task with the *DeepCaption* library's new version developed during the winter and spring 2020. The novel implementation uses a transformer-like architecture [21] paired with an LSTM [22]. It improves



(a) **(b)**

Figure 1: The clustering output found a set of unknown persons in the video (1a). Using the frames of *Unknown 0* we are able to build the model for Elin Skagersten-Ström and recognise her in other videos (1b).

sequential representations with recurrency by refining connections between videos and captions and by utilizing multi-level stacked relational attention. Figure 2 shows the overall architecture of *DeepCaption*'s new caption generation model. The visual features and captions are treated as inputs to the encoding and decoding layers, and the last output of the decoding layers are processed by an LSTM. All the outputs from the decoding layers are collected and used to attend the representations generated by the recurrent language model to produce the output words.

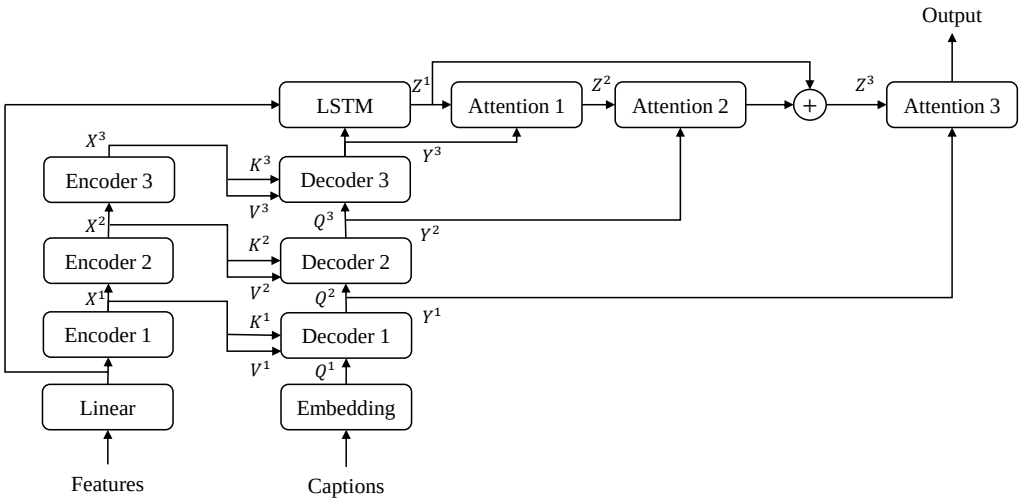


Figure 2: The architecture of *DeepCaption*'s new stacked attention model.

The stacked attention model is based on the Transformer model [21], in which the intra- and cross-relations between the visual and the text features are calculated via scaled dot-product attention. The attention function receives three sequential sets with length s, and d_{model} dimensions, denoted as queries Q, keys K, and values V. The attention function is

defined as

$$Attention(Q, K, V) \;=\; softmax(\frac{QK^T}{\sqrt{d_{model}}})V \;, \tag{1}$$

where $Q \in \mathbb{R}^{s \times d_{model}}$ is a matrix of query vectors and K and $V \in \mathbb{R}^{s \times d_{model}}$ are matrices of key and value vectors. Given a set of features from videos, intra-modality attention is obtained in the encoder with self-attention on the different feature inputs. Cross-modality dependencies are modeled in the decoder via cross-modal attention operations between the visual and textual features. Multihead attention is employed for improving the feature representation and with k heads it is formulated as

$$Multihead(Q, K, V) \;=\; concat(h_1, \ldots, h_k)W^O \tag{2}$$

$$h_i \;=\; Attention(QW_i^Q, KW_i^K, VW_i^V) \;,\; i = 1, \ldots, k \tag{3}$$

with matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_{model}/k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_{model}/k}$ and $W_i^V \in \mathbb{R}^{d_{model} \times d_{model}/k}$ used in each of the k attention heads, and $W^O \in \mathbb{R}^{d_{model} \times d_{model}}$.

In our stacked attention model, we have used the depth of $N = 3$ layers, as seen in Figure 2. Both the encoding and decoding layers are stacked sequentially. The stack of N encoding layers generates multi-level outputs $X = (X^1, \ldots, X^N)$ which are used as the key and value inputs, K and V, of the cross-modal attention in each corresponding decoder. The query inputs Q come there from the word embeddings of the caption. The multi-level cross-modal relations of visual and textual features provide refined inputs for the attention on the recurrent language model. The decoding layers depend on the visual features and the previously generated words. We collect them and exploit the outputs level by level.

The stacked attention mechanism always uses the decoder output Y^{N-j+1} to attend the attention-stacked LSTM output Z^j. First, with $j = 1$ and given the decoder output Y^N and the LSTM output Z^1, the stacked attention mechanism concatenates Y^N and Z^1 and transforms them linearly to the same dimension with Z. The stacked attention is then defined as element-wise or Hadamard product

$$StackedAttention(Y^{N-j+1}, Z^j) \;=\; \alpha(Y, Z) \odot Z \;, \tag{4}$$

where we have dropped the superscripts on the right for clarity and $\alpha(\cdot, \cdot)$ is a function that generates a element-wise multiplication matrix which has the same dimensions as Z. The function $\alpha(\cdot, \cdot)$ is defined as

$$\alpha(Y, Z) \;=\; \sigma(W\,[Y, Z] + b) \;, \tag{5}$$

where $[\cdot, \cdot]$ stands for concatenation, $\sigma(\cdot)$ is the sigmoid function, and W and b are the weight and the bias. The stacked attention for the full sequence of LSTM outputs is then formed by applying the attention (4) sequentially with $j = 1, \ldots, N$. As can be seen in Figure 2, we have additionally utilized a skip connection from the LSTM output Z^1 to Z^3.

Word-level cross-entropy (XE) is used to pre-train the model, which is then fine-tuned via reinforcement learning. During the XE training, the model predictions are conditioned on the previous annotated words. Training with reinforcement learning employs the self-critical (SC) [23] training method. During the decoding, both greedy and stochastic samples of the output sequences are used at each time step. We employ the CIDEr-D [24] score as the reward of the SC reinforcement learning. The reward is baselined by a greedy sample rather than the mean of rewards. The gradient is then defined as

$$\nabla_\theta L(\theta) \;=\; -\frac{1}{M}\sum_{i=1}^{M}\left((r(w^i) - r(\hat{w}))\nabla_\theta \log p(w^i)\right) \;, \tag{6}$$

where w^i is the i-th stochastic sample in a batch, $\hat{w}$ is the greedy search sample and $r(\cdot)$ is the CIDEr-D reward function. When predicting, we perform greedy search and keep words with the highest predicted probabilities within the vocabulary.

A selection of slightly different captioning models were created with and without stacked attention by using the MS COCO [25], TGIF [26] and VATEX [27] datasets for training. These datasets contained 82,783, 125,713 and 41,250 images or videos, and 414,113, 125,713 and 825,000 captions, respectively. ResNet-152 [28] image features and I3D [29] video features were used as the visual inputs to all models. Both features are 2048-dimensional. Validation was performed with the TRECVID 2018 VTT task's available ground truth data and used to select the best architecture variants and hyperparameters for the allowed four final submissions.

Figure 3 shows the PicSOM team's four allowed submissions ("s1" to "s4") to the TRECVID 2020 VTT task, evaluated together with all other submissions to the task. Submission "s4" closely resembles the PicSOM team's best model in TRECVID 2019 that used the MS COCO and TGIF datasets for training a model with self-critical learning without a Transformer and attention model. Submission "s3" is similar to "s4", but stacked attention is used in the captioning model. Submissions "s1" and "s2" differ from the other two in their use of the VATEX dataset in addition to the MS COCO and TGIF datasets in the training. "s1" uses stacked attention whereas "s2" does not use it. Based on the results, it is evident that the introduction of the stacked attention model has notably improved the results, but even more important factor has been the use of the VATEX data. One can see that with respect to the CIDEr-D results, all of the PicSOM team's submissions are behind the results of the winner team, but ahead of all the other teams. On the other hand, in BLEU results, the PicSOM team's submission "s2" seems to be the third best one among the 19 submissions.

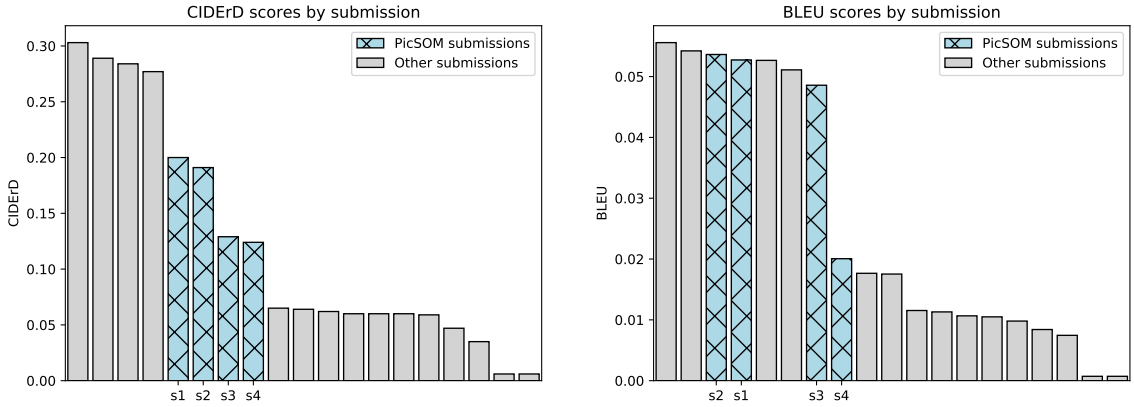


Figure 3: PicSOM team's CIDEr-D (left) and BLEU (right) results in the TRECVID 2020 VTT task.

3.3 Visual location classification

The SUN Scene Categorization Benchmark database contains 899 scene categories and 130,519 images.[8] Out of these categories, 397 have 100 or more annotated images and these form the SUN397 scene category set [30]. In total, SUN397 comprises 108,754 images, and the categories are divided into a three-level hierarchy. The first level of the hierarchy divides the categories into *indoor, outdoor natural*, and *outdoor man-made*. Examples of category division with all three levels include: 1. *indoor* → *home or hotel* → *alcove*, 2. *outdoor natural* → *water, ice, snow* → *bayou*, and 3. *outdoor man-made* → *transport* → *airfield*.

[8] https://vision.princeton.edu/projects/2010/SUN/

We used 2048-dimensional ResNet-152 [28] features and a feedforward network with one hidden layer and 397 SoftMax outputs to form a visual location classifier. The model was trained with the full SUN397 dataset and it can be applied for classifying the scene or location of shots with a simple flat vocabulary of 397 categories.

4 Multimodal test data

In our multimodal experiments, we have used a small subset consisting of eight MeMAD programs. This subset reflects different characteristics of content in terms of number of topics present, number of languages present, formal vs. colloquial speech, studio vs. field recording (both audio and video) and the amount of music present in the content. The dataset consists of programs broadcast by Yle and the details of the programs are summarised in Table 2. The characteristics of each program are:

- 1 *Uutiset Lounais-Suomi* is a local news program with six news topics presented by one studio journalist, very little music.
- 2 *Spotlight* is a technology documentary whose topic is in cyber security.
- 3 *Sohvasurffaajat* is a travel documentary with two hostesses, colloquial speech and field recordings
- 4 *Vallankumouksen lapset* is an old documentary about Cuba in the 1970's, only diegetic music.
- 5 *Kuningaskuluttaja* is a consumer magazine program containing six topics and four journalists.
- 6 *Strömsö* is a lifestyle magazíne program with ten topics and two primary journalists.
- 7 *Egenland* is a lifestyle magazíne program with four topics, two journalists and a large variety of languages.
- 8 *Mitt triathlon* is a sport-related interview between two people with some additional sports footage, very little music.

id	name	mm:ss	shots	known	faces	speakers	languages
1	*Uutiset Lounais-Suomi*	10:09	93	10	17	14	fi
2	*Spotlight*	28:09	325	11	16	18	sv en fi
3	*Sohvasurffaajat*	28:33	292	4	9	4	fi en
4	*Vallankumouksen lapset*	52:26	239	11	13	17	fi es
5	*Kuningaskuluttaja*	28:09	232	16	19	16	fi
6	*Strömsö*	28:35	336	8	9	7	sv fi
7	*Egenland*	28:51	311	15	16	16	fi sv en de fr
8	*Mitt triathlon*	7:17	65	2	3	2	sv fi
	total	212:09	1893	77	102	94	

Table 2: Yle programs used in the demonstrations and evaluation of the methods. The columns show the numerical id, program name and the duration of the program. Column "shots" shows the number of automatically detected shots, "known" is the number of known persons who have been recognized in the detected faces, "faces" shows the total number of individuals whose faces have been detected frequently, "speakers" shows the number of different human voices detected in speaker diarisation, and "languages" lists the languages spoken in the program, in the decreasing order of prevalence.

5 Multimodal processing and outputs

The ultimate goal of multimodal processing applied to media content description is to be able to generate *human-like* content descriptions, in which the persons, objects, actions and environments, both seen and heard, are described in a sophisticated way. In MeMAD Deliverable D5.3, the different levels of human understanding of media content were discussed and a hierarchy of five levels, ranging bottom-up from *Level 1: key elements* to *Level 5: audio description*, was identified. The work on automatic multimodal content description in MeMAD targeted in the direction of *Level 5* still comprehending that in practice we could only reach *Level 2: content descriptions* and even it with only imperfect results. In this section, we elaborate and evaluate a number of improvements we have made that combine unimodal key elements into multimodal content descriptions, thus raising from *Level 1* to *Level 2* in the hierarchy of Deliverable D5.3.

The multimodal processing stages have been implemented as a part of Aalto University's PicSOM media analysis framework, to which all unimodal inputs from each partner's processing pipelines are collected. All different variants of multimodal processing follow a common approach, where the visual caption produced by AALTO's *DeepCaption* video captioning program is used as a starting point. The speech recognition results have not been used in this processing, but they have been made available as such as subtitles. In addition, they have been used as inputs for, e.g., named entity recognition in WP3 and machine translation in WP4.

DeepCaption produces one caption for each visual shot of the programs. The breaks between the shots have been located with a simple shot boundary detection method as described in MeMAD Deliverable D2.1. The number of visual shots in each program can be seen in the "shots" column of Table 2 together with the length of the program in the "mm:ss" column. The average length of the shots in the whole material can be calculated to be approximately 6.7 seconds. The variation between the programs is otherwise minor, but the shots in the 1970's program 4 *Vallankumouksen lapset* are clearly longer than those in the more contemporary programs.

The information from the other modalities is combined with the produced captions in a number of sequential automatic postprocessing stages that will be described in the following sections. The changes caused by these postprocessing methods are by their nature either *corrections* or *enrichments* of the original captions. Corrections are such that, for example, the gender of a person mentioned in the original caption can be changed from "a man" to "a woman" or vice versa based on the gender of the person who has been recognized in the shot. For enrichment, the references to "a man" and "a woman" can be replaced with the actual names of the persons recognized by their faces, or additional information can be added in the sentences to specify the visible location or audible language, speaker or other sounds. Depending on the purpose for which corrections and enrichments are needed, any subset of the postprocessing stages can be applied.

The outputs of the postprocessing have been visualized with overlaid subtitle texts in the SubStation Alpha format[9] and with the ELAN media annotation and analysis tool[10] using its own XML format EAF.

5.1 Person naming by face

Naming of the person in the "raw" captions is based on using the face detections and recognitions generated by EURECOM's *FaceRec* software described in Section 3.1. Faces were detected

[9] http://moodub.free.fr/video/ass-specs.doc
[10] https://archive.mpi.nl/tla/elan

and facial image features were extracted in all frames of the eight test programs listed in Section 4. For each program, we then tried to identify some face image samples for those persons who have appeared most frequently in that program. This was accomplished by first clustering all faces of the program in 20 clusters based on their feature representations. By human inspection and expertise, those "clean" clusters that seemed to contain faces of only one person were then labeled with either the person's known name or a synthetic label like "Unknown man 601" where the number is unique across the whole test dataset. This annotation task was quite straightforward and did not require more than approximately 20 minutes per program. The effort of clustering and human annotation can be avoided for programs and persons for which one can obtain labeled face samples from the web or by any other means.

For each program, the numbers of persons whose faces were seen and names known can be found in the "known" column of Table 2. Similarly, the "faces" column shows the total number of persons whose faces have been detected and who have appeared in the programs frequently enough that their faces have formed a "clean" cluster in the facial feature space. We can see that in total there were 77 known and 25 unknown persons in the test material. For both the known and unknown individuals, we additionally assigned a metadata label to identify whether they are *male* or *female* by their gender. None of the persons in the material appeared in more than one program. Statistics of these annotations are shown in Table 3.

	male	**female**	**total**
known	44	33	77
unknown	15	10	25
total	59	43	102

Table 3: Statistics of known vs. unknown persons and their genders in the test dataset.

In the person naming algorithm, the identity labels – either a known person's name or an unknown person's unique number – are accumulated from frame-wise detections to entire visual shots by tracking the face bounding boxes and solving the most likely identities with voting. As an outcome of this process, we obtain a list of person names (possibly an unknown person with an unique number) and their genders, either a *male* or a *female*. The next step is a textual analysis of the raw caption to see if it contains references to person entities such as "a man", "a girl", "someone", etc. These references are then replaced with the available person names primarily so that the genders of the references and the persons match. If there is more than one possible match, for example two identified male faces, but only one reference to "a man" or "a boy", then the first matching face identity that is available in the inputs is selected. Similarly, if there are two references to female persons in the caption, but only one female name available from the faces, the first reference is replaced. After processing the gender-specific references, neutral references such as "someone", that allow both male and female name replacement, are handled in the similar manner. Only in the case when there are non-replaced identities and references left after these two matching phases, matches that violate the gender preservation rule are allowed. In that way, for example, a reference to "a man" can be replaced with a female name.

The raw captions contain structures like "with another man" and the postprocessing algorithm has been devised to remove the word "another" if the word "man" has been replaced with a name. Similarly, pronouns such as "he" or "her" are forced to match the gender of the preceding noun in the sentence.

Examples:

1. **input:** *A man* is smiling and talking to someone. **output:** *Jonathan Granbacka* is smiling and talking to someone. (8 *Mitt triathlon* 01:43)

Figure 4: A frame from 7 *Egenland* in which the raw caption referred to "a man" and "a woman", but postprocessing associated male identities for both. The resulting caption is: "*Ted Wallin* standing next to *Nicke Aldén* in a room."

2. **input:** *A woman* is talking to *a man* and *he* is smiling. **output:** *Pauliina Räsänen* is talking to *Slava Volkov* and *he* is smiling. (7 *Egenland* 09:55)

3. **input:** *A man* standing next to *a woman* in a room. **output:** *Ted Wallin* standing next to *Nicke Aldén* in a room. (7 *Egenland* 04:06, see Figure 4)

Examples 1 and 2 show simple cases where correct matches between the genders of the faces and the references have been found. Figure 4 shows a video frame that corresponds to example 3 and demonstrates how also an erroneous reference to "a woman" in the raw caption has been replaced with a male name.

Statistics of the person naming process in all test videos are shown in Table 4. The table shows for each program the total number of shots and in column "hits" the number of shots in which the caption has been modified to contain a person name. The "total" column shows the total number of person references substituted with a name. By comparing columns "shots" and "hits", we can see that at least one person has been named in approximately every third shot, but this fraction varies greatly between the programs. Faces are the most frequent in 7 *Mitt triathlon* with 62% and the least frequent in 4 *Vallankumouksen lapset* with 24% of the shots. The total number of namings shows that in most of the programs there have been also shots where more than one person has been named. These numbers are naturally dependent not only on the face recognitions, but also on the contents of the raw captions that need to refer to more than one person so that more than one replacement can take place.

The accuracy of face-based person naming was evaluated for all the eight test programs and they are shown in the right-most columns in Table 4. We can conclude that the face naming process failed in two programs, 2 *Spotlight* and 4 *Vallankumouksen lapset*, whereas the results for the other six programs can be considered as satisfactory. Inspection of those two programs reveals that they contain far more faces than the used number of 20 clusters, and most of the faces appear in the programs only for a very short time. Consequently, the creation of the face classifiers has, in the case of these programs, failed already in the clustering stage. Based on this small-scale evaluation, we can state that the face-based person naming procedure leads to more than 70% accuracy for 75% of the programs with our default parameter settings.

id	name	faces	named	shots	hits	total	corr	acc
1	*Uutiset Lounais-Suomi*	17	15	93	41	45	33	73%
2	*Spotlight*	16	16	325	145	160	84	53%
3	*Sohvasurffaajat*	9	6	292	81	82	64	78%
4	*Vallankumouksen lapset*	13	12	239	57	65	30	46%
5	*Kuningaskuluttaja*	19	17	232	84	84	74	88%
6	*Strömsö*	9	8	336	98	120	104	87%
7	*Egenland*	16	15	311	88	104	81	78%
8	*Mitt triathlon*	3	3	65	40	45	39	87%
	total	102	92	1893	634	705	509	72%

Table 4: Statistics related to facial person naming in captions. The first columns show the numerical id and the program name. Column "faces" shows how many individuals have been identified in the program and "named" how many of them were named in the captions as the result of the person naming process. Columns "shots" and "hits" tell the total number of shots and in how many of them one or more faces have been named, whereas column "total" tells the total number of faces named. The "corr" column shows the count of correct name replacements in the caption, followed by the corresponding accuracy.

5.2 Gender correction

In addition to the process described above of replacing gender references in the captions with person names, we have made it possible to just correct the gender mention in the caption if the facial recognition evidence supports that decision. Facial gender classifications have been available in MeMAD from INA's Face Gender classifier described in Deliverable D2.2 and from EURECOM's *FaceRec* library as a side product of the person name labeling. In these experiments we used EURECOM's *FaceRec*.

The approach for gender correction is a simplified version of the person naming algorithm described in the previous section. Instead of replacing a person reference in the raw caption with the person's name available from the recognised face, we just use the correct gender in the reference as it is available as metadata related to the person. In that way, "a man" and "a woman" or "a boy" and "a girl" can be interchanged in the caption.

Examples:

1. **input:** *A man* is sitting on a bench and talking. **output:** *A woman* is sitting on a bench and talking. (1 *Uutiset Lounais-Suomi* 04:03)

2. **input:** *A man* standing next to *a woman* in a room. **output:** *A man* standing next to *another man* in a room. (7 *Egenland* 04:06)

Example 1 is a simple case in which just the gender of the reference has been corrected, whereas example 2 (which is the same caption as example 3 of the previous section) shows how an erroneous reference to "a woman" has been replaced with "another man".

Face-based gender correction without name replacement was not regarded as a particularly useful type of multimodal postprocessing for the raw captions. Therefore, it was not experimented further nor evaluated after testing its feasibility with a small number of sample captions.

5.3 Human face and voice association

Associating a person's name and voice in a program has been implemented as another post-processing stage. Sufficient information for this purpose has been available from EURECOM's *FaceRec* and LLS/Lingsoft's speaker diarisation. The columns "faces" and "speakers" in Table 2 show the numbers of different faces and voices identified in each program in the test

dataset. We can observe that the numbers of identified faces and voices match quite well in all programs.

The process of finding matches between the faces and voices aims to associate person names or numerical identities to the voices found in speaker diarisation. It is assumed that the voices initially have no associated names. As a result of successful assignment of names to voices, we would be able to enrich the captions with information on the names of the persons whose voices can be heard.

The performance of the association process is dependent of the accuracies of the facial person identifications and the speaker voice diarisations. The algorithm assumes that most of the times a person's face is visible when his or her voice is heard. The mapping from voices to faces is not assumed to be one-to-one, but many-to-one so that multiple voices can be associated with one face and thus one name. It can be assumed that the same person's voice may well get split to multiple identities in diarisation depending on the background noises and other contextual factors.

The association process works as follows. For each identified voice in the program, the occurrences of that voice together with each of the identified faces are searched for, and the durations of those moments, expressed as numbers of video frames, are summed. The frame counts are then divided by the total numbers of frames where each voice has been heard. The identity of the most frequent face is associated with that voice if its fraction of frame matches exceeds a preset threshold, for which we have used the value 0.25. An example of these absolute and relative frame counts in the matching process is shown in Table 5 for the test program 5 *Kuningaskuluttaja*.

voice	**total**	**face**	**frames**	**frac**	**face**	**frames**	**frac**	**face**	**frames**	**frac**
#3	3041	Maarit Åström-K.	2028	0.667	Maaria Vatanen	623	0.205	Marko Rajamäki	221	0.073
#2	2663	Panu Vatanen	1349	0.507	Jan Karlsson	403	0.151	Pauli Salminen	160	0.060
#4	1493	Jukka Sassi	806	0.540	Marko Rajamäki	378	0.253	Maarit Åström-K.	309	0.207
#6	1426	Pauli Salminen	1118	0.784	Panu Vatanen	308	0.216			
#9	1210	Kaisu Nevasalmi	986	0.815	Pekka Sillanpää	137	0.113	Riikka Turunen	87	0.072
#13	1030	Pekka Sillanpää	703	0.683	Kaisu Nevasalmi	237	0.230	Jarkko Väänänen	51	0.050
#12	1021	Marko Rajamäki	942	0.923	Maarit Åström-K.	79	0.077			
#1	911	Maaria Vatanen	501	0.550	Maarit Åström-K.	402	0.441	Riikka Turunen	8	0.009
#14	607	Tanja Talvenheimo	607	1.000						
#15	576	Mitja Nylund	566	0.983	Jarkko Väänänen	10	0.017			
#16	514	Jarkko Väänänen	505	0.982	Mitja Nylund	9	0.018			
#10	507	Petteri Kolinen	507	1.000						
#7	476	Miikka Vartiainen	476	1.000						
#8	475	Riikka Turunen	475	1.000						
#11	254	Jan Karlsson	179	0.705	Tanja Talvenheimo	75	0.295			
#5	92	Kaisu Nevasalmi	24	0.261	Mitja Nylund	24	0.261	Jukka Sassi	21	0.228

Table 5: Matching of voices and faces in test program 5 *Kuningaskuluttaja*. The "total" column shows the total number of frames where the voice appears together with a recognized face and the voices have been ordered according to this value. The columns show the top three matched faces.

One can see in Table 5 that two persons, Maarit Åström-Kupsanen and Panu Vatanen, appear much more often than the other persons. This is easily explainable by the fact that they are journalists who appear in multiple scenes and host a number of guests in this magazine-type program. The third journalist, Kaisu Nevasalmi, appears only in one scene together with her interviewee Pekka Sillanpää. One can see that the faces and voices of these two persons, Nevasalmi and Sillanpää, have been mixed with each other as often as 11% and 23% of the time, but still the voices #9 and #13 have been correctly mapped to their names. It can also be seen that some voices were matched with only one face, whereas for some others the associations have been more mixed. In total, 14 out of the 16 voices, or 88%, have been

correctly mapped to a person's name so that more than half of the time when that voice has been detected it really has belonged to that named person. The two erroneous cases, voices #11 and #5, were also those that had the least total number of frames where they appeared together with any recognized face. In deeper inspection, voice #5 turned out to be a mixed cluster of many people's voices. On the the other hand, voice #11 belonged to the fourth journalist whose face was never visible in the program and his voice got erroneously associated with the name of one of the persons he interviewed.

The detailed analysis of 5 *Kuningaskuluttaja* thus showed that the association of person names from faces to their heard voices was successful for almost all voices and, more importantly, for those voices that were the most frequent in the program. The most severe shortcoming of the association was due to a "hidden" interviewer whose face did not appear in the broadcast. The face recognitions for this program were accurate enough so that the few misclassified faces did not cause any failure in naming the voices. On the other hand, the diarisation contained some errors, as shown in Table 1 in Section 2.1, which were also reflected as sporadic errors in the identification of the speakers.

5.4 Speaker naming

The caption enrichment with speaker name information can be performed alone or combined with face-based person naming. In either case, the speaker information has been appended to the captions with words "*while <person name> speaks*". If both person naming by faces and speaker naming are used and the recognized voice belongs to one of the visible persons, the speaker identification is not added in order to avoid naming the same person twice.

Examples:

1. **input:** A group of ducks are swimming in the water. **output:** A group of ducks are swimming in the water *while Leena Arvela-Hellén speaks*. (1 *Uutiset Lounais-Suomi* 04:14)

2. **input:** *A man* is driving a car and looking at something. **output:** *Maaria Vatanen* is driving a car and looking at something *while Maarit Åström-Kupsanen speaks*. (5 *Kuningaskuluttaja* 02:54)

3. **input:** *A woman* and *a man* preparing food in a kitchen. **output:** *Camilla Forsén-Ström* and *Paul Svensson* preparing food in a kitchen. (6 *Strömsö* 22:59)

Example 1 shows the simple case where the speaker's name has been appended to a caption that does not have any person's name in it. In example 2, a second person other than the speaker is visible and her name has been added based on face recognition, and so the names of the both get mentioned. Finally, example 3 presents the situation where a person, this time Camilla Forsén-Ström, is simultaneously both visible and audible, and for that reason her name is *not* appended in the role of a speaker.

Table 6 shows statistics of the speaker identification task for six programs among the test dataset. Two of the programs, 2 *Spotlight* and 4 *Vallankumouksen lapset* were not included in this experiments beceause, as could be seen in Table 5, the association between faces and voices had not been successful for these two programs and speaker naming could therefore not be expected to be successful either. We can see in Table 6 that speaker naming works best for the program 5 *Kuningaskuluttaja*, in which a total of 12 speakers are mentioned in the enriched captions. In that program, almost 80% of the mentions are correct, and almost all errors are related to the voice of one journalist whose face does not appear in any of shots and therefore neither was his voice associated with any name. For a similar reason, the process fails also for 1 *Uutiset Lounais-Suomi* where one of the journalists' voice cannot be matched with

id	name	faces	speakers	named	shots	hits	corr	acc
1	*Uutiset Lounais-Suomi*	17	14	8	93	49	27	55%
3	*Sohvasurffaajat*	9	4	3	292	134	70	52%
5	*Kuningaskuluttaja*	19	16	12	232	111	88	79%
6	*Strömsö*	9	7	3	336	97	69	71%
7	*Egenland*	16	16	11	311	141	66	47%
8	*Mitt triathlon*	3	2	1	65	20	16	80%
	total	73	59	38	1329	552	336	61%

Table 6: Statistics related to speaker naming in captions. The first columns show the numerical id and the program name. Column "faces" shows how many individuals have been identified in the program by faces and "speakers" tells how many distinct speakers were found in diarisation. The "named" column tells how many persons' names were included in the enriched captions as speakers. Columns "shots" and "hits" tell the total number of shots and in how many of them the speaker has been mentioned. The "corr" column shows the count of correct speaker identifications in the captions, followed by the corresponding accuracy.

her face and name. The low results for 3 *Sohvasurffaajat* and 7 *Egenland* follow from the fact that in both programs the background music is in many places rather loud. Therefore, both speech segmentation and speaker diarisation have performed clearly worse than for programs like 5 *Kuningaskuluttaja* in which there is background music, but it is not so emphasized.

The usefulness of the results for programs 3 *Sohvasurffaajat*, 6 *Strömsö* and 8 *Mitt triathlon* is further reduced by the fact that only a very small number of people appear in those programs as "narrators" whose voice can be heard even when they are not visible. This is of course a feature of the programs themselves and the speaker naming process cannot change that fact.

5.5 Location naming

The raw captions generated by the *DeepCaption* model do not typically contain references to locations, places or scenes that are the context of the event or happenings in the videos. This is mostly due to the MS COCO [25], TGIF [26] and VATEX [27] datasets used to train the caption generator models. In order to circumvent this shortcoming, we implemented a postprocessing mechanism that adds explicit location words in the captions if the SUN397 location classifier described in Section 3.3 produces a location classification score that exceeds a preset threshold. The model was applied to the middle frame of each shot and as the preset threshold we used the *ad hoc* value 0.5 in these experiments.

Information on the detected location class is added as an enrichment in the captions with the help of a set of simple rules that convert the SUN397 class name to the form of a prepositional phrase. Some example transformations are shown in Table 7. If the classification score exceeds the threshold, such a phrase is prepended to the caption.

SUN397 class	prepositional phrase
airplane_cabin	in an airplane cabin
bridge	on a bridge
casino_indoor	in a casino
cathedral_outdoor	outside a cathedral
desert_sand	in a sand desert
gazebo_exterior	outside a gazebo

Table 7: Examples of conversions from SUN397 class names to prepositional phrases.

Examples:

1. **input:** A group of people are sitting at a table. **output:** *In a lecture room* a group of people are sitting at a table. (4 *Vallankumouksen lapset* 09:18)

2. **input:** A car is driving down a road. **output:** *On a highway* a car is driving down a road. (5 *Kuningaskuluttaja* 12:44)

3. **input:** A man is standing in a small boat. **output:** *On a boat deck* a man is standing in a small boat. (7 *Egenland* 16:51)

Examples 1 and 2 show cases where the location enrichment has been visually correct and the prepended prepositional phrase makes well sense with the rest of the sentence. Example 3 demonstrates a quite rare case where the raw caption already contains some form of mention about the location and the added phrase is redundant.

Table 8 shows statistics and results of the location naming experiment. The validity of the identified location was assessed by one researcher in a Boolean fashion based on seeing the middle frame of the shot and remembering the contents of program. The judgements are naturally subjective and some tolerance was shown for accepting not fully accurate but related or visually possible locations. We can see that 1 *Strömsö* fails miserably, mostly because the vast majority of the shots are closeups of people or even their hands, in which the actual locations, such as a kitchen, remain visually ambiguous. Classification accuracies as high as those obtained for 3 *Sohvasurffaajat* and 1 *Uutiset Lounais-Suomi* are quite promising when we remember that the SUN397 dataset provides 397 scene classes.

id	name	shots	named	corr	acc
1	*Uutiset Lounais-Suomi*	93	22	13	59%
2	*Spotlight*	325	87	41	47%
3	*Sohvasurffaajat*	292	85	52	61%
4	*Vallankumouksen lapset*	239	59	33	56%
5	*Kuningaskuluttaja*	232	76	33	43%
6	*Strömsö*	336	93	8	9%
7	*Egenland*	311	83	26	31%
8	*Mitt triathlon*	65	12	5	42%
	total	1893	517	211	41%

Table 8: Statistics related to location naming in captions. The first columns show the numerical id and the program name. Columns "shots" and "named" tell the total number of shots and in how many of them the location has been mentioned. The "corr" column shows the count of correct location identifications in the captions, followed by the corresponding accuracy.

5.6 Audio background naming

The raw captions have been generated with a deep learning model that uses only visual features as inputs. However, the captions still contain such phrases as "a man is talking" or "a girl is singing", but these are based on the common sense or imagination of the crowdsourcing workers who have annotated the datasets used for training the model. All mentions about audible events can therefore be considered as totally coincidental. For obtaining really multimodal captions, we implemented a simple postprocessing technique that adds information about the audible part of the videos. This was based on using the Google AudioSet data [18] for training a 527-class audio classifier whose classification result can be appended to the caption if the classification score exceeds a preset threshold. For this threshold we used the *ad hoc* value 0.5.

Unlike the SUN397 class names that need some textual transformation for being integrated into the captions, the AudioSet class names can be used directly by just lowercasing the class names and changing underscores to spaces. The audio class is then suffixed to the caption in the form "*while <audio class> is heard*". If there already is the speaker's name appended,

id	name	shots	named	music	other	corr	acc
1	*Uutiset Lounais-Suomi*	93	14	13	1	0	0%
2	*Spotlight*	325	73	73	1	0	0%
3	*Sohvasurffaajat*	292	110	108	2	0	0%
4	*Vallankumouksen lapset*	239	33	30	3	1	33%
5	*Kuningaskuluttaja*	232	119	118	1	1	100%
6	*Strömsö*	336	140	136	4	1	25%
7	*Egenland*	311	168	168	0	—	—
8	*Mitt triathlon*	65	6	5	1	0	0%
	total	1893	663	650	13	3	23%

Table 9: Statistics related to audio background naming in captions. The first columns show the numerical id and the program name. Columns "shots" and "named" tell the total number of shots and in how many of them the audio background has been mentioned. The "music" and "other" columns show the counts of music and other identifications, respectively. The "corr" column shows the count of correct non-music identifications in the captions, followed by the corresponding accuracy.

then the combination of the speaker and audio background information is written in the form "*while <person name> speaks and <audio class> is heard*".

Audio classifications containing the word "speech" are ignored because we already have a postprocessing mechanism for incorporating information about the names of the speakers. After the removal of "speech" classes, the "music" class turned out to be by far the most frequent audio classification result in the test dataset. This could be expected as all of the programs contained either diegetic or incidental music. Table 9 shows statistics of the programs and their audio contents. We can see that audio content classification produced results other than "speech" whose score exceeded the threshold in approximately one shot out of every three shots. There is notable variability in that fraction between the programs. We can also see that, with the threshold we used, approximately only 2% of the shots contained some other audio than "speech" or "music". We may assume that also some of "music" classifications must be wrong, but more interesting is to study the accuracy of the other classifications.

All non-music audio background classifications:

1. bee (1 *Uutiset Lounais-Suomi* 07:00) wrong: speech
2. emergency vehicle (2 *Spotlight* 00:07) wrong: yelling crowd
3. insect (3 *Sohvasurffaajat* 04:03) wrong: speech
4. crushing (3 *Sohvasurffaajat* 23:36) wrong: music
5. livestock (4 *Vallankumouksen lapset* 09:15) wrong: clatter of fruits
6. harpsichord (4 *Vallankumouksen lapset* 18:38) almost correct: grand piano
7. clip-clop (4 *Vallankumouksen lapset* 23:34) wrong: jackhammer
8. drum (5 *Kuningaskuluttaja* 14:20) correct
9. chopping food (6 *Strömsö* 21:00) correct
10. humming (6 *Strömsö* 24:59) wrong: spoon hitting a bowl
11. raindrop (6 *Strömsö* 25:09) wrong: music
12. livestock (6 *Strömsö* 25:18) wrong: pepper grinder
13. gasp (8 *Mitt triathlon* 07:16) wrong: music like gasping

Overall, the results of this experiment are disappointing. First, all programs that were included in the test set turned out to consist mostly of human speech and music, both overlapping and alternating, and there was hardly any other audio content that could be recognized. Second, even in the rare cases when the audio classifier detected something other than speech or music, the actual content was most often either speech or music of some form.

5.7 Language identification

The identification of the languages that are spoken in the programs was brought into studies during the last year of the MeMAD project. It was considered as a necessary step before any language-dependent processing, such as automatic speech recognition, machine translation and media indexing, could be applied. Language identification was implemented as described in Section 2.2 and it was experimented with two programs in the test dataset. These programs 7 *Egenland* and 8 *Mitt triathlon* contained speech in Finnish, Finland Swedish, French, German and English.

Languages are identified on speech segments that result from speaker diarisation. Some of these segments actually contain no speech because the diarisation model misrecognises background music and sounds as speech. The performance of the back-end Naïve Bayes classifier trained on the five languages and samples without speech is reported in Table 10 for both programs.

Results are better for the 8 *Mitt triathlon* program, possibly because it has less background music than 7 *Egenland* and thus cleaner audio. Furthermore, in the former program, there are more long speaker turns and therefore long diarisation segments. These long segments can be easier to recognize because there is more input for the classifier. Even though many of the language identification errors appear to be resulting from problems in the audio input or in the segmentation of the speaker turns, there remains a lot of room for improvements in modeling and classification of the spoken language. One obvious direction would be to retrieve more and better training data with manually confirmed language labels. However, more experiments and error analysis would also be needed for detecting when the automatic language identification fails and how this could be improved.

language	segments	corr	acc
de	13	7	54%
en	17	13	77%
fi	119	35	29%
fr	3	0	0%
sv	44	14	32%
no language	64	34	53%
total	260	103	40%

(a) 7 *Egenland*

language	segments	corr	acc
de	0	0	—
en	0	0	—
fi	29	20	69%
fr	0	0	—
sv	19	10	53%
no language	8	2	25%
total	56	32	57%

(b) 8 *Mitt triathlon*

Table 10: Language identification results on diarisation segments for the test programs.

The boundaries of diarisation segments typically do not coincide with the boundaries of visual shots. This is problematic because the captions are generated and enriched on often very short visual shots, whereas speech and therefore also language segments are much longer and typically span multiple visual shots. Despite this mismatch, the language identifications are in our current implementation mapped to visual shots and combined with the enriched captions. Each visual shot that is inside an audible language segment receives information on that language. The identity of the speaker is in these cases always already included in the captions and the language information is appended to it to produce a phrase in the form "*while <person name> speaks in <language>*".

Statistics of the two test programs are shown in Table 11 together with the counts and percentages of the correctly identified languages in captions. By comparing Tables 10 and 11, we can see that the language namings that have ended in the enriched captions have been more accurate than language identifications overall in the same materials. The results can be considered promising, but further experiments will still be needed for validating the usability of the results.

id	name	shots	named	corr	acc
7	*Egenland*	311	74	44	59%
8	*Mitt triathlon*	65	17	15	88%
	total	376	91	59	65%

Table 11: Statistics related to language identification in captions. The first columns show the numerical id and the program name. Columns "shots" and "named" tell the total number of shots and in how many of them the spoken language has been mentioned. The "corr" column shows the count of correct language identifications in the captions, followed by the corresponding accuracy.

5.8 Combined results

If some combination of the previous postprocessing methods is used instead of a single method, the postprocessing algorithm possesses the following interdependencies between the enrichments:

1. The classified location is prepended to the caption.
2. The recognised speaker and language follow the caption.
3. The classified audio background is the last enrichment.
4. If a person's name has been inserted as a result of face recognition, it will not be inserted again as a result of voice recognition.
5. The identified language is mentioned only if the speaker has been identified.
6. Audio background recognition of any form of "speech" is always prevented.

The generic template of the location, speaker, language and audio background enrichments is of the form "*in/on/at/outside <a/an location>* <caption with person name replacements> *while <person name> speaks in <language> and <audio class> is heard*".

Examples with image samples of the shots in Figure 5:

1. **input:** *A woman* is walking down a street and talking. **output:** *In a crosswalk Maarit Åström-Kupsanen* is walking down a street and talking *while music is heard*. (5 *Kuningaskuluttaja* 00:46, see Figure 5a)
2. **input:** *A man* is holding a knife in *his* hand. **output:** *Jukka Torikka* is holding a knife in *his* hand *while Hannamari Hoikkala speaks in Finnish and music is heard*. (7 *Egenland* 16:38, see Figure 5b)
3. **input:** *A man* is talking to *a woman* and *she* is smiling. **output:** *In a subway Elin Skagersten-Ström* is talking to *Alicia Trezise-Segervall* and *she* is smiling. (6 *Strömsö* 6:57, see Figure 5c)
4. **input:** *A woman* is dancing and singing with *a man*. **output:** *In badlands Lotta Kaihua* is dancing and singing with *a man while music is heard*. (3 *Sohvasurffaajat* 20:45, see Figure 5d)

We can see that the caption postprocessing of example 1 has been fully successful. Example 2 is otherwise successful but the rope has been recognised as a knife in the caption. In example 3, the location has been erroneously classified as a subway. In the last example, the method has been unable to correct the erroneous reference to "a man" with a proper name because the other woman's face was not recognised. Overall, the results are reasonable and it is very likely that at least one of the enrichments is always correct, whereas they can only be very seldom correct all at the same time.

(a) (b)

(c) (d)

Figure 5: Video shot samples whose enriched captions are given in the text.

6 Discussion

In this final deliverable of Work Package WP2 *Automatic multimodal content analysis* of the MeMAD project, we have shown our latest developments of the uni- and multimodal media analysis techniques developed by various MeMAD partners in tight collaboration. Furthermore, we have demonstrated how the recognized key elements of media content can be combined together for obtaining enriched, more human-like descriptions of video contents. Referring to the hierarchy of different levels of human understanding of media contents, we have progressed from *Level 1: key elements* to *Level 2: content descriptions* while aiming towards the ultimate goal of *Level 5: audio description*.

As a novel contribution, we have studied the identification of the language that is spoken in the videos. Another important contribution was in the improvement of the clustering technique that allows to find and label face image samples also for persons whose names are not known in advance or for whom one cannot find samples with image search in the web. Notable improvement has been made also in the accuracy of speaker diarisation, which is necessary not only for its own sake, but also for the subsequent tasks of spoken language identification and human face and voice association.

A set of interrelated postprocessing techniques was presented for enriching raw visual captions generated with the *DeepCaption* program for the shots of the programs. The postprocessing is able to intelligently replace references to persons in the captions with their proper names if they are known based of facial person recognition. Furthermore, the algorithm is able to add mentions on the persons who can be recognised by their voice, together with the language they are speaking. Finally, the technique can enrich the captions also with classifications of both the visual environments and the audio background sounds of the shots.

The facial recognitions and name substitutions worked reasonably well. This was partly due to the type of the programs and captions where typically only one or two recognisable individuals were seen in the view and the substitutions were therefore quite straightforward. Two of the eight test programs were too difficult for this simple approach basically because there were clearly more than the assumed 20 people appearing in them and the straightforward clustering and labeling process for obtaining representative samples of face images fell short.

Naming the audible speakers was dependent on the success of associating people's faces and voices together. For those programs in which the face recognition had been successful, this mapping process managed to give names to the speakers well in slightly more than half of the cases. Consequently, we can state that for this small test dataset, in approximately half of programs the voices of the speakers could be named sufficiently well. Related to the speakers, the languages they spoke were recognised for two programs in the test dataset. The result of this small-scale experiment was promising, but further experiments will be needed to validate the applicability of the technique.

Accurate classification and naming of the locations seen in the video shots turned out to be quite difficult. Even if some classifications were quite tolerably regarded as acceptable, the average accuracy of annotations was slightly less than 50%. It has to be remembered, however, that the SUN397 ontology for scene recognition contains almost 400 scene categories. The results of the classification of the audio background sounds other than music in the test programs were disappointing. The outcome can be explained by the fact that there were only very few background sounds in the test material and those sounds that existed were frequently mixed with speech and music. Humans are good at perceiving background sounds even in such difficult content, but for a machine, the same task still seems to be too demanding.

The quality of the visual captions has continuously improved during the three years of the MeMAD project. This can be seen in the development of the yearly results of the TRECVID VTT evaluation task. However, it still seems that the accuracy of the raw visual captions is a delimiting factor for the quality of the final enriched captions that we can generate. Mostly this follows from the mismatch between the types, motifs and visual contents of the videos used for training the *DeepCaption* generator model and the contents of the television broadcast programs studied in the MeMAD project. Also, the quality of the human-written captions for the videos available for model training is in many datasets quite low. Maybe in the future we could obtain training data for video captioning with materials that are closer to broadcast programs. The currently available caption datasets have been collected from social network sites and as such do not match MeMAD's target domain. Continuous improvements for the feature extraction methods, attention techniques and language models used in the contemporary automatic captioning models are of course also called for.

In the MeMAD project's original work plan, we envisaged paying special attention not only to the humans but also to the objects visible in the broadcast programs. During the course of the project, the need for precise description of objects was found to be less important than better recognition of the scene or visual context of the shots. Similarly, identifying the spoken language was found to be a more important research topic than explicit and accurate object naming. In the future, objects and actions that humans perform with them are very likely to become an even more important research topic in multimedia content analysis.

Overall, in MeMAD's WP2 the work on visual content description remained on the granularity level of visual shots. Shots are natural atomic building blocks for captions and therefore also a good starting point for multimodal content description, but for more fluent, dense and human-like narrative one will need to consider media units longer than just visual shots. When this will become reality, in terms of MeMAD Deliverable D5.3's hierarchy, we will be able to climb to *Level 3: cohesive ties and establishing relevance* and gradually even further. An approach in this direction will still be demonstrated in Deliverables D6.8 and D6.9 where higher-level semantic shot segmentation will be implemented and tested in the Limecraft Flow environment.

7 Summary of the MeMAD multimodal analysis software

name	**st**	**provider**	**license**	**code**	**description**
PicSOM	U	AALTO	Apache 2	C++	multimedia content analysis framework
DeepCaption	U	AALTO	Apache 2	Python3	image and video captioning
visual-storytelling	O	AALTO	Apache 2	Python3	visual storytelling
AALTO ASR	O	AALTO	MIT		speech recognition scripts using Kaldi
Speaker-aware training	O	AALTO	MIT	Python3	speaker-aware training of end-to-end ASR using Espnet
SphereDiar	U	AALTO	MIT	Python3	tools for overlapping speaker detection, speaker verification and speaker diarisation
avsr	O	AALTO	MIT	Python3	multimodal ASR
AudioTagger	U	AALTO	Apache 2	Python3	audio event classification
LIDBOX	N	AALTO	MIT	Python3	spoken language identifications
Image Caption Translation	U	AALTO	MIT	Python3	multi-modal image caption translation for WP4
statistical-tools	O	AALTO	MIT	Python3	tools for creating dataset statistics for WP5
FaceRec	U	EURECOM	Apache 2	Python3	tools for detecting, aligning and recognising faces in video
inaSpeechSegmenter	U	INA	MIT	Python3	speech, music and noise segmentation; speaker gender detection
inaFaceGender	O	INA	proprietary	Python3	face detection, tracking and gender classification
Flow Shot Cut Detector	O	Limecraft	proprietary	C	subprogram of broadcast video production system
Lingsoft Speech Service	U	Lingsoft	proprietary	Python3, C++, JavaScript	automatic speech recognition service via an API

Table 12: Software components of MeMAD related to multimodal content analysis. Column "st" shows the status symbols standing for "O" = old version of MeMAD D2.2, "U" = updated version from D2.2 to D2.3, and "N" = new component in D2.3.

Table 12 contains a summary of the software components used in the MeMAD project for multimodal content analysis and available for the project's members. Software components that have proprietary license are available for the MeMAD partners as software or as a service. Those that have been identified to have a liberal licensing scheme, such as MIT or Apache 2, are publicly available as source code in MeMAD's GitHub page located at:

`https://github.com/MeMAD-project`

The liberally licensed software components discussed in this report have been specifically collected for ease of installation in a repository named `mmca`:

`https://github.com/MeMAD-project/mmca`

Some of the modules in `mmca` are physically located outside of the MeMAD GitHub project, but the Git submodule mechanism facilitates their seamless availability from their true locations. All of the software packages can be obtained with a single operation:

```
git clone https://github.com/MeMAD-project/mmca.git --recursive
```

Each of the subdirectories created inside the `mmca` directory contains its own further installation and use instructions. Specifically, each package will have up-to-date instructions for installation and usage in a file called `README.md` in the corresponding directory. The licensing information of each submodule is available in a file named `LICENSE`.

8 References

[1] Matias Lindgren, Tommi Jauhiainen, and Mikko Kurimo. Releasing a toolkit and comparing the performance of language embeddings across various spoken language identification datasets. In *Proceedings of Interspeech 2020*, 2020.

[2] Abhilash Jain, Aku Rouhe, Stig-Arne Grönroos, and Mikko Kurimo. Finnish ASR with deep transformer models. In *Proceedings of Interspeech 2020*, 2020.

[3] Jorma Laaksonen and Zixin Guo. PicSOM experiments in TRECVID 2020. In *Proceedings of the TRECVID 2020 Workshop*, Gaithersburg, MD, USA, December 2020.

[4] T. J. Park, K. J. Han, M. Kumar, and S. Narayanan. Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap. *IEEE Signal Processing Letters*, 27:381–385, 2020.

[5] Zhiyuan Tang, Dong Wang, and Liming Song. AP19-OLR Challenge: Three Tasks and Their Baselines. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1917–1921, 11 2019.

[6] Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. Automatic Dialect Detection in Arabic Broadcast Speech. In *Proc. Interspeech 2016*, pages 2934–2938, 2016.

[7] Lukas Mateju, Petr Cerva, Jindrich Zdansky, and Radek Safarik. Using Deep Neural Networks for Identification of Slavic Languages from Acoustic Signal. In *Proc. Interspeech 2018*, pages 1803–1807, 9 2018.

[8] David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. Spoken Language Recognition using X-vectors. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 105–111, 6 2018.

[9] Suwon Shon, Ahmed Ali, and James Glass. Convolutional Neural Network and Language Embeddings for End-to-End Dialect Recognition. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 98–104, 6 2018.

[10] Tuomas Kaseva, Aku Rouhe, and Mikko Kurimo. Spherediar - an efficient speaker diarization system for meeting data. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[11] György Kovács, László Tóth, Dirk Van Compernolle, and Sriram Ganapathy. Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout. *Pattern Recognition Letters*, 100:44 – 50, 2017.

[12] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, François Grondin, et al. The JHU-MIT system description for NIST SRE18. *Johns Hopkins University, Baltimore, MD, Tech. Rep*, 2018.

[13] Xiaoxiao Miao, Ian McLoughlin, and Yonghong Yan. A New Time-Frequency Attention Mechanism for TDNN and CNN-LSTM-TDNN, with Application to Language Identification. In *Proc. Interspeech 2019*, pages 4080–4084, 9 2019.

[14] Matias Lindgren. Deep learning for spoken language identification. Master's thesis, Aalto University, 2020.

[15] Peter Smit, Sami Virpioja, and Mikko Kurimo. Advances in subword-based HMM-DNN speech recognition across languages. *Computer Speech and Language*, 66, March 2021. — openaire: EC/H2020/780069/EU//MeMAD.

[16] Abhilash Jain. Finnish language modeling and asr with deep transformer models. Master's thesis, Aalto University, 2020.

[17] Zhicun Xu, Peter Smit, and Mikko Kurimo. The Aalto system based on fine-tuned audioset features for DCASE2018 task2 - general purpose audio tagging. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Surrey, UK, November 2018.

[18] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 776–780. IEEE, 2017.

[19] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[23] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[26] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. TGIF: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2016.

[27] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research, 2019.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[30] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

A Appendices

A.1 Abstracts of Master's Theses

The following pages contain abstracts of the Master's Theses whose full contents can be accessed through the links below:

- Matias Lindgren: *Deep learning for spoken language identification*. Master's Thesis, Aalto University, 2020. [14]
- Abdilash Jain: *Finnish language modeling and ASR with Deep Transformer Models*. Master's Thesis, Aalto University, 2020. [16]

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Matias Erik Lindgren

Title
Deep learning for spoken language identification

School School of Science

Master's programme Computer, Communication and Information Sciences

Major Computer Science **Code** SCI3042

Supervisor Professor Mikko Kurimo

Advisor PhD Tommi Jauhiainen

Level Master's thesis **Date** 26th April 2020 **Pages** 10+91 **Language** English

Abstract

This thesis applies deep learning based classification techniques to identify natural languages from speech. The primary motivation behind this thesis is to implement accurate techniques for segmenting multimedia materials by the languages spoken in them.

Several existing state-of-the-art, deep learning based approaches are discussed and a subset of the discussed approaches are selected for quantitative experimentation. The selected model architectures are trained on several well-known spoken language identification datasets containing several different languages. Segmentation granularity varies between models, some supporting input audio lengths of 0.2 seconds, while others require 10 second long input to make a language decision.

Results from the thesis experiments show that an unsupervised representation of acoustic units, produced by a deep sequence-to-sequence autoencoder, cannot reach the language identification performance of a supervised representation, produced by a multilingual phoneme recognizer. Contrary to most existing results, in this thesis, acoustic-phonetic language classifiers trained on labeled spectral representations outperform phonotactic classifiers trained on bottleneck features of a multilingual phoneme recognizer. More work is required, using transcribed datasets and automatic speech recognition techniques, to investigate why phoneme embeddings did not outperform simple, labeled spectral features.

While an accurate online language segmentation tool for multimedia materials could not be constructed, the work completed in this thesis provides several insights for building feasible, modern spoken language identification systems. As a side-product of the experiments performed during this thesis, a free open source spoken language identification software library called "lidbox" was developed, allowing future experiments to begin where the experiments of this thesis end.

Keywords language identification, machine learning, deep neural networks, speech analysis

Aalto University
School of Science

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Abhilash Jain		
Title:	Finnish language modeling and ASR with Deep Transformer Models		
Date:	July 30, 2020	**Pages:**	74
Major:	Computer Science	**Code:**	SCI3042
Supervisor:	Professor Mikko Kurimo		
Advisors:	Aku Rouhe M.Sc. (Tech.) Stig-Arne Grönroos M.Sc. (Tech.)		

Transformers have taken the centre stage for most NLP applications after LSTM's were established as advanced approaches to sequence modeling and transduction problems such as language modeling and speech recognition. Recently, BERT and Transformer-XL based architectures demonstrated the efficacy of Transformer models as a language model pre-trained on large corpora.

It is important to understand that these strides have mostly been made with the English language for which abundant resources already exist. It is also morphologically a simpler language in comparisons to an agglutinative language like Finnish. An important question then arises on the usage and performances of these Transformer models for Finnish.

In this thesis, we take an important Natural Language Processing task like Automatic Speech Recognition and compare different architectures mentioned above for Finnish. First, we perform an intrinsic evaluation task like language modeling to understand how well suited the Transformer architectures are for Finnish. Next for an extrinsic evaluation, we employ these models as a language model in an ASR system and analyze their performance.

Our Transformer models performed exceptionally well both in the language modeling and the ASR task. Transformer-XL achieves 29% (absolute) better perplexity and 3% (absolute) better WER in ASR than our previous best LSTM-based approach. We also introduce a novel three-pass decoding scheme which improves the ASR performance by 8%. In this thesis, we also achieve the following (i) to formulate an alpha smoothing framework to use the non-autoregressive BERT language model for an ASR task, and (ii) to explore sub-word units with Transformer-XL for an agglutinative language like Finnish.

Keywords:	language modeling, automatic speech recognition, transformer, BERT, transformer-xl
Language:	English

A.2 AALTO's language identification paper in Interspeech 2020 conference [1]

This paper describes AALTO's toolkit and experiments in spoken language identification. The toolkit is applied to implement three baseline models that have performed well in three language identification challenges and reproduce their results to verify the code and hyperparameters. A few modifications and AALTO's own method are evaluated on the same datasets.

Releasing a toolkit and comparing the performance of language embeddings across various spoken language identification datasets

Matias Lindgren[1]*, Tommi Jauhiainen*[2]*, Mikko Kurimo*[1]

[1]Department of Signal Processing and Acoustics, Aalto University, Finland
[2]Department of Digital Humanities, University of Helsinki, Finland

matias.lindgren@iki.fi, tommi.jauhiainen@helsinki.fi, mikko.kurimo@aalto.fi

Abstract

In this paper, we propose a software toolkit for easier end-to-end training of deep learning based spoken language identification models across several speech datasets. We apply our toolkit to implement three baseline models, one speaker recognition model, and three x-vector architecture variations, which are trained on three datasets previously used in spoken language identification experiments. All models are trained separately on each dataset (closed task) and on a combination of all datasets (open task), after which we compare if the open task training yields better language embeddings. We begin by training all models end-to-end as discriminative classifiers of spectral features, labeled by language. Then, we extract language embedding vectors from the trained end-to-end models, train separate Gaussian Naive Bayes classifiers on the vectors, and compare which model provides best language embeddings for the back-end classifier. Our experiments show that the open task condition leads to improved language identification performance on only one of the datasets. In addition, we discovered that increasing x-vector model robustness with random frequency channel dropout significantly reduces its end-to-end classification performance on the test set, while not affecting back-end classification performance of its embeddings. Finally, we note that two baseline models consistently outperformed all other models.

Index Terms: spoken language identification, deep learning, x-vector, language embedding, TensorFlow

1. Introduction

Spoken Language Identification (SLI)[1] is the task of identifying the language of a spoken utterance. For a thorough introduction to SLI, see [1]. Automating the comparison of several different SLI models can be challenging if each model uses its own data pipeline, making it difficult to ensure that a particular comparison is not affected by unknown variability of the underlying implementations. Although several approaches to deep learning based end-to-end SLI have been proposed, there is no easy, unified way to train and compare several SLI models. We seek to remedy this situation by proposing an easy to use toolkit, built on the popular TensorFlow deep learning framework [2]. We use the toolkit to implement one baseline x-vector model, three variations of it, and three additional SLI architectures. These architectures are then trained on three SLI datasets discussed in Section 3.

X-vector SLI X-vector based SLI has in the past two years shown to be a viable alternative to i-vector based SLI [3, 4, 5, 6, 7, 8, 9], although x-vectors were originally proposed for speaker recognition [10]. Some extensions that have been proposed to the x-vector architecture include 2-dimensional (2D) convolutional neural network (CNN) feature extractor front-ends, attention mechanisms and long short-term memory (LSTM) layers [6], as well as a larger time-delayed deep neural network (TDNN) structure with residual, skip connections [11]. In the fourth oriental language recognition (AP19-OLR) challenge, an x-vector based SLI model was given as the baseline [12]. In addition to supporting end-to-end SLI, the x-vector architecture can also be used to learn a fixed-length language embedding representation for variable length utterances [4]. An alternative way of discovering embedding spaces is to explicitly map the embedded vectors onto a hypersphere by L_2-normalization, where the angular distance of embedding vectors imply class similarity. This approach has outperformed i-vector based systems both in SLI [13] and speaker recognition [14].

Contributions of this paper We publish a new, end-to-end SLI toolkit for running multiple SLI experiments on multiple datasets, implement seven existing SLI architectures on our toolkit, and run experiments on three SLI datasets. We implement the SphereSpeaker speaker recognition architecture [14] on our toolkit and apply it to SLI for the first time. We release our toolkit online as free open source software[2]. In addition, we publish[3] the configuration files, dataset metadata, and scripts for all experiments discussed in this paper to improve reproducibility of our results.

2. End-to-end deep learning SLI toolkit

Several frameworks and toolkits supporting end-to-end, automatic speech recognition (ASR) have been proposed [15, 16, 17, 18]. Applying ASR methods to SLI, e.g. by training language classifiers on phoneme embeddings extracted from a phoneme recognizer, has shown to work very well [19, 20, 21, 22]. While end-to-end SLI performed directly on labeled speech features is usually outperformed by models that utilize phoneme level information, it is sometimes possible to reach good performance also with end-to-end models [6, 23]. Our toolkit focuses only on the latter, simpler task of end-to-end SLI from spectral or cepstral features. This allows the toolkit to remain more lightweight compared to the larger frameworks that focus on ASR tasks such as sequence annotation based on connectionist temporal classification [24]. However, one trade-off is that there is no support for training multilingual bottleneck features. These must be acquired using other methods if one wishes to use them as input features. Nevertheless, our toolkit provides an easy starting point for training SLI models on an several speech datasets, therefore making the comparison of different methods significantly easier. Similar to librosa [25] providing an easy Python programming language interface for

[1]SLI is also known as Spoken Language Recognition (SLR).

[2]https://github.com/matiaslindgren/lidbox

[3]https://github.com/matiaslindgren/interspeech-2020-lidbox

audio analysis, we hope our toolkit can provide an easy way to get started with SLI experiments.

The core of our toolkit has been implemented with TensorFlow 2, which supports signal processing and model training on the GPU. Simple signal processing techniques based on the open source implementations of librosa [25] and Kaldi [15] have been implemented into our toolkit to support high performance, parallel feature extraction. Note that all feature extraction is performed with TensorFlow, librosa or Kaldi is not required. We apply dataset iterators[4] from the TensorFlow data module to construct parallelized data pipelines that support datasets with unbounded amounts of samples. All data processing steps from reading the acoustic data from disk to training a SLI model on spectral features is done in batches, allowing the user to control memory usage regardless of dataset size. Intermediate pipeline state can be cached to disk into a single binary file. This allows the user to perform all high latency operations, such as random disk access of several utterances, in a single pre-processing pass. The toolkit also supports extraction of language embedding vectors from trained end-to-end SLI models and a simple back-end training module for the language vectors. Lastly, we claim that the toolkit could also be used for end-to-end classification of speech signals beyond SLI, since the toolkit does not make any assumptions on what the provided signal labels encode.

3. Datasets

In this paper, we use three different datasets for training and testing. We did not have access to the NIST LRE datasets.

AP19-OLR Oriental Language Recognition challenge 2019 (AP19-OLR), contains speech in 10 languages mainly spoken in Asia and one out-of-set (OOS) mixture of European languages [12]. The dataset includes 261 hours of training data and 5 hours of test data. For testing, we use the AP19-OLR short-utterance task (AP19-OLR task 1), where all test utterances have a duration of exactly 1 second. For validation, we use the AP19-OLR short-utterance validation set, which contains 6 hours of exactly 1 second long utterances. The OOS mixture does not have any samples in the test or validation set.

MGB-3 3rd Multi-Genre Broadcast challenge (MGB-3), contains speech in 5 regional Arabic dialects [26]. The dataset includes 53 hours of training data and 10 hours of test data. We follow the approach of [27], who augmented the training set with randomly chosen validation set utterances. In this approach, we choose uniformly at random 90% validation set utterances separately for each of the 5 classes, create 4 new copies of each utterance, and include this 5-fold augmented validation set into the training set. This brings the amount of training data to 99 hours. The remaining 10% is used as a held-out validation set. The test set contains test utterances of varying length, with the median duration at 15 seconds.

DoSL Dataset of Slavic Languages (DoSL), contains speech in 11 Slavic languages [19]. The dataset includes 220 hours of training data and 8 hours of test data, where test utterances are almost uniformly distributed between 5 and 6 seconds. DoSL does not provide a validation set so we created our own held-out set from the training set. We choose uniformly at random 500 utterances for each of the 11 languages from the training set and remove these utterances from the training set. This results in 5500 validation set utterances (8 hours), with a median duration of 4.8 seconds.

[4] https://www.tensorflow.org/versions/r2.2/api_docs/python/tf/data/Dataset

Closed and open tasks All seven models described in Section 4 are first trained in a "closed task" manner, separately on every training set of every dataset, using the validation set of each dataset to monitor training progress. When the best weights for each model has been discovered, measured against the validation set, we evaluate the models as end-to-end language predictors on each of the test sets. Then, we use the same, trained models as feature extractors to generate fixed-length language vectors from the training and test sets of every dataset. The language vector training sets are then used for training a back-end classifier, which is evaluated as a language predictor on language vectors extracted from each test set, using each end-to-end model.

In addition, we train all models in an "open task" manner, where each model is first trained on the union of all three training sets, using the union of all three validation sets to monitor training progress. After discovering the best weights, we extract language vectors in a closed task manner, separately for each dataset and train the back-end classifiers. By doing this, we compare if end-to-end SLI models as language vector extractors benefit from training on a larger amount of data, while still extracting language vectors from a smaller amount of data.

Note that the union of all three training sets does not include the OOS mixture from AP19-OLR (label "unknown") since we could not confirm whether this mixture contains languages from MGB-3 or DoSL. In this case, including it would present multiple labels for the same language, which could have a detrimental effect for model performance in the open task.

4. End-to-end experiments

Models We use our toolkit to implement three different baseline models for the three datasets described in Section 3, one speaker recognition model, and three different x-vector variations. All models are based on existing architectures. The choice of baselines for MGB-3 and DoSL were motivated by the success previously reported for these two datasets. The baseline architecture for AP19-OLR was defined as the competition baseline by [12] and we choose the same baseline. We enumerate the model architectures used in this paper as follows:

1. Regular x-vector [4], but TDNN layers replaced with temporal convolution layers as in [27]. See Table 2 for our configuration. This is our baseline model for AP19-OLR. In addition, this is the baseline x-vector architecture for creating the variations, i.e. models 5, 6, and 7.
2. 1D CNNs with average pooling over the time dimension and three FC layers [27]. This is our baseline model for MGB-3.
3. Two bi-directional gated recurrent units (BGRU) and three FC layers [19]. This is our baseline model for DoSL.
4. SphereSpeaker architecture, that has recently been successful for speaker recognition [14], now applied to SLI.
5. Model 1 with increased robustness by applying channel dropout on input during training, using probability 0.5 [28]. See Figure 1 for an example on channel dropout applied on FBANK input.
6. Model 1 extended with additional layers before the statistics pooling layer as in [11, Table 3]. Initially, we used FC layers to extend the model but noticed that this caused unstable training progress, leading to non-finite weight values. Therefore, we use only temporal convolution layers before the statistics pooling layer, as in Model 1.

Model	10^6 params	D	mean min/epoch
1	4.5	512	5.1
2	9.0	1500	6.2
3	8.6	1024	30.9
4	5.1	1000	33.0
5	4.5	512	5.0
6	6.4	512	6.7
7	4.6	512	13.4

Table 1: *Amount of parameters in millions and the language embedding dimension D, representing the number of features, for each model. We also measured the average amount of minutes per epoch required to train each model architecture using a Tesla V100-PCIE-32GB on the open task of all three datasets.*

	Layer	Output shape
0	Input $\mathbf{X}$	198×40
1	Conv1D $512 \times 5 \times 1$	198×512
2	Conv1D $512 \times 3 \times 2$	99×512
3	Conv1D $512 \times 3 \times 3$	33×512
4	Conv1D $512 \times 1 \times 1$	33×512
5	Conv1D $1500 \times 1 \times 1$	33×1500
6	Reduce mean and stddev.	3000
7	FC ReLU 512	512
8	FC ReLU 512	512
9	FC log-softmax N	N

Table 2: *Implementation of model 1. Notation for convolution layers is "filters×kernel width×stride". All layers except 0, 6, and 9 are ReLU activated and batch normalized. Layers 1–5 are batch normalized over the time axis to avoid diluting information over different frequency channels. X-vectors are extracted as outputs of layer 7 before ReLU activation and batch normalization.*

7. Model 1 with a small 2D CNN front-end for gathering frequency information [6], see Table 3 for our 2D CNN configuration.

Model input and output All acoustic data consists of single-channel waveform signals of varying lengths at 16 kHz sample rate. Audio files with invalid headers are dropped and channels of multi-channel signals are merged by averaging. We assume the ratio of invalid files is negligible compared to all files. The list of training set audio files is shuffled before reading, separately for all three training sets. We note that energy-based voice activity detection (VAD) is used in 4 out of 7 cases in the reference experiments [4, 27, 6, 11], partially used in 2 out of 7 cases [19, 14], and not used in [28]. Therefore, we decided to use a simple energy-based VAD in all our experiments. Our VAD approach is based on comparing the root-mean-square (RMS) values of non-overlapping 10 ms windows to the mean RMS over all 10 ms windows within each signal. We require non-speech segments to contain at least 100 ms of continuous non-speech decisions before they are dropped.

	Layer	Output shape
0	Input $\mathbf{X}$	198×40
1	Add channel dimension	$198 \times 40 \times 1$
2	Conv2D $256 \times (1,5) \times (1,1)$	$198 \times 36 \times 256$
3	Conv2D $128 \times (1,3) \times (1,2)$	$198 \times 17 \times 128$
4	Conv2D $64 \times (1,3) \times (1,3)$	$198 \times 5 \times 64$
5	Conv2D $32 \times (1,3) \times (1,3)$	$198 \times 1 \times 32$
6	Flatten channels	198×32

Table 3: *2D CNN front-end of an x-vector model [6], which is prepended to model 1 to produce model 7. Notation for convolution layers is "filters×kernel size×strides". I.e. we use unit width and stride over time steps but larger height and stride over frequency channels. Outputs of layers 2–5 are ReLU activated and batch normalized.*

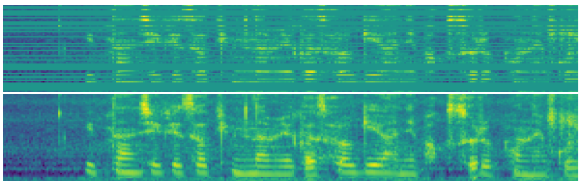


Figure 1: *Log-scale Mel-spectrogram of a randomly chosen 3 second training utterance from AP19-OLR with (top) and without (bottom) channel dropout with probability 0.5.*

After VAD, every signal that is shorter than 2 seconds is repeatedly appended to itself until its duration is at least 2 seconds. Then, all signals are divided into utterances of exactly 2 seconds, with 0.5 second overlap. A similar repeating method was used by [29], who suggested that repeating the feature sequence extracted from a short utterance is an effective way of providing more information about the utterance to the language classifier. However, we choose to repeat the short utterances already in the time domain, because using fixed length samples allows our toolkit to store all acoustic data more effectively into a single file, containing batches of 2 second utterances. In addition, using a fixed utterance length provides more comparable SLI results between different model architectures. We store all 2 second utterances into 9 files, each containing the training, validation, and test sets for all three datasets. From these 2 second utterances, log-scale Mel-spectra $\mathbf{X} \in \mathbb{R}^{198\times40}$ (FBANK) is extracted with a 512-point FFT from 25 ms windows using 10 ms offset, warped into 40 Mel-frequency bins. Finally, each channel is centered to zero mean within each $\mathbf{X}$. The same procedure is applied also to the validation and test data.

Each model outputs non-positive log-softmax language scores $\mathbf{y} \in \mathbb{R}^N$ where N is the amount of languages in the target set. Final language scores for a variable length test utterance are produced by averaging over the outputs on all its 2 second chunks. In case a model fails to produce predictions for a test utterance, smallest possible (worst case) log-softmax language scores are generated for all language classes for that test utterance.

Training and testing All seven models are trained using TensorFlow version 2.1 with the Adam optimizer [30] using learning rate 0.0001. All other optimizer parameters are left to their default values. Training samples $\mathbf{X}$ are shuffled within a buffer containing $2 \cdot 10^4$ samples, from which training batches containing 64 samples are produced. One exception is model 3, for which we apply an additional preparation step before shuffling, where each FBANK $\mathbf{X}$ is divided into non-overlapping chunks $\mathbf{X}' \in \mathbb{R}^{30\times40}$. We chose a time context of 30 time steps based on the results by [19]. In addition, the shuffle buffer size for training model 3 is $\lfloor 198/30 \rfloor \cdot 2 \cdot 10^4 = 1.2 \cdot 10^5$, to make sure the amount of information is approximately same as in the shuffle buffers of all other models. Otherwise model 3 is trained exactly as all other models. For all models, early stopping is applied with the condition that multi-class cross-entropy loss [31, Eq. 4.108] has not improved within 20 epochs from its lowest value, measured on the validation set. After early stopping, model weights are reset to the best weights, chosen from the

Model	AP19-OLR	MGB-3	DoSL	Avg
1	**0.125**	0.260	**0.019**	0.135
2	0.126	**0.236**	0.024	**0.129**
3	0.128	0.263	0.037	0.143
4	0.146	0.238	0.024	0.136
5	0.222	0.334	0.166	0.241
6	**0.125**	0.285	0.026	0.145
7	0.139	0.325	0.030	0.165
Avg	0.144	0.277	0.047	

Table 4: *C_{avg} closed task, end-to-end.*

epoch when validation loss was its lowest value.

After training, language scores $\mathbf{y}$ are predicted for each test set utterance $\mathbf{X}$. For model 3, language scores are first predicted for all $\mathbf{X}'$, which are then averaged to produce language scores for the original 2 second utterance chunk $\mathbf{X}$, from which all $\mathbf{X}'$ were partitioned from. For simplicity, we place equal weight on each chunk. Then, we use our toolkit to compute the average detection cost (C_{avg}) as defined in NIST LRE2017 [32, Eq. 6], with parameters $C_{Miss} = C_{FA} = 1$ and $P_{Target} = 0.5$, on the predicted language scores.

Results From Table 4, we compare the C_{avg} results of models 1, 2, and 3 to the respective baseline results of AP19-OLR (0.126) [12], MGB-3 (0.218) [27], and DoSL (0.013) [19], and note that our results are within 0.1, 1.8, and 2.4 percentage points. We note that model 1 outperforms other models on AP19-OLR and DoSL, while model 2 is best on its reference dataset MGB-3 and the best overall model on average. Also, model 5 clearly produces worse results compared to all other models.

5. Back-end classifiers

We choose Gaussian Naive Bayes[5] (GNB) for back-end classification because we found it to be stable and fast to train. While it is currently the only back-end classifier supported by our toolkit, we believe new classifiers are relatively easy to add, especially if they conform to the scikit-learn classifier interface[6]. The GNB models are trained on fixed-length language vectors $\mathbf{x} \in \mathbb{R}^D$ (see Table 1), which are extracted from end-to-end models trained on the closed and open tasks.

Language vector extraction For all models (except 4), $\mathbf{x}$ is extracted from a fully-connected (FC) layer, without activations and batch normalization. For models 1, 5, 6, and 7, $\mathbf{x}$ is an x-vector, i.e. the outputs of the first FC layer after the statistics pooling layer [4]. For model 2, we choose $\mathbf{x}$ as the output of the first FC layer after the average pooling layer. For model 3, we choose $\mathbf{x}$ as the output of the first FC layer after the second BGRU layer. For model 4, $\mathbf{x}$ is the L_2-normalized output of the SphereSpeaker embedding layer [14].

Classification For all three training and test sets, we feed $\mathbf{X}$ to the seven different, trained end-to-end models and collect new training and test sets of language vectors $\mathbf{x}$. All D features of each $\mathbf{x}$ are scaled to zero mean and unit variance using statistics computed separately on each training set. Dimensionality is reduced to $N-1$ by probabilistic linear discriminant analysis[7] [33] and L_2-normalization is applied on the reduced vectors. Then, GNB is fitted on the reduced $\mathbf{x}' \in \mathbb{R}^{N-1}$ training set vectors. After training, we predict log-likelihoods on the language vectors extracted from the test set. Finally, we compute C_{avg} values from these log-likelihoods using the same approach as with the log-softmax scores and report the minimum C_{avg} value as the final result. We have included this back-end training pipeline into our toolkit.

[5] https://scikit-learn.org/0.23/modules/generated/sklearn.naive_bayes.GaussianNB.html

[6] https://scikit-learn.org/stable/supervised_learning.html

[7] https://github.com/RaviSoji/plda

Model	AP19-OLR	MGB-3	DoSL	Avg
1	**0.135**	0.218	0.030	**0.128**
2	0.153	**0.211**	**0.028**	0.130
3	0.147	0.248	0.056	0.151
4	0.167	0.213	0.032	0.137
5	0.136	0.248	0.041	0.142
6	0.143	0.243	0.029	0.138
7	0.149	0.255	0.040	0.148
Avg	0.147	0.234	0.036	

Table 5: *C_{avg} closed task, GNB on embeddings.*

Model	AP19-OLR	MGB-3	DoSL	Avg
1	0.162	0.193	0.030	**0.128**
2	0.173	**0.181**	**0.028**	**0.128**
3	0.169	0.226	0.064	0.153
4	0.202	0.193	0.035	0.143
5	**0.150**	0.200	0.041	0.130
6	0.173	0.209	0.038	0.140
7	0.167	0.202	0.041	0.137
Avg	0.171	0.201	0.040	

Table 6: *C_{avg} open task, GNB on embeddings.*

Results From Table 5 we see that on MGB-3, all models except 5 are better as language vector extractors than end-to-end classifiers, but not on AP19-OLR or DoSL. We also see that frequency channel dropout (model 5) significantly weakens end-to-end SLI performance (Table 4), without affecting language embedding quality (Table 5). This is in contrast with the common assumption that speaker embedding quality is proportional to the end-to-end classification performance of the speaker recognition model used for extracting the embeddings [34]. Regarding the open task training condition, we note by comparing Tables 6 and 5 that increasing the dataset size improves the quality of language embeddings on MGB-3 even further, but at the same time reduces the results on both AP19-OLR and DoSL.

6. Conclusions

We proposed a new toolkit for easier end-to-end SLI and applied the toolkit for comparing SLI performance of different deep learning models both end-to-end and using back-end classifiers on language embedding vectors. We noticed that language embeddings on the Arabic dialect dataset MGB-3 are easier to classify with GNB when we allow an open task approach where language embedding model is trained on all available, three training sets. However, this was not beneficial for the two other datasets. In addition, we noticed that poor end-to-end SLI performance of frequency channel dropout [28] did not imply poor back-end classification performance.

7. Acknowledgements

This work was supported by EU's Horizon 2020 research and innovation programme via the project MeMAD (GA 780069). Computational resources were provided by Aalto Science-IT.

8. References

[1] H. Li, B. Ma, and K. A. Lee, "Spoken Language Recognition: From Fundamentals to Practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 5 2013.

[2] Martín Abadi et al. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from https://tensorflow.org.

[3] A. Mccree, D. Snyder, G. Sell, and D. Garcia-Romero, "Language recognition for telephone and video speech: The jhu hltcoe submission for nist lre17." in *Odyssey*, 2018, pp. 68–73.

[4] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken Language Recognition using X-vectors," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 6 2018, pp. 105–111.

[5] A. Hanani and R. Naser, "Spoken arabic dialect recognition using x-vectors," *Natural Language Engineering*, pp. 1–10, 5 2020.

[6] X. Miao, I. McLoughlin, and Y. Yan, "A New Time-Frequency Attention Mechanism for TDNN and CNN-LSTM-TDNN, with Application to Language Identification," in *Proc. Interspeech 2019*, 9 2019, pp. 4080–4084.

[7] H. Wu, W. Cai, M. Li, J. Gao, S. Zhang, Z. Lyu, and S. Huang, "Dku-tencent submission to oriental language recognition ap18-olr challenge," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 1646–1651.

[8] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly data loader and utterance-level aggregation for speaker and language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1038–1051, 2020.

[9] P. Jain, K. Gurugubelli, and A. K. Vuppala, "Study on the effect of emotional speech on language identification," in *2020 National Conference on Communications (NCC)*. IEEE, 2020, pp. 1–6.

[10] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4 2018, pp. 5329–5333.

[11] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin *et al.*, "The JHU-MIT system description for NIST SRE18," *Johns Hopkins University, Baltimore, MD, Tech. Rep*, 2018.

[12] Z. Tang, D. Wang, and L. Song, "AP19-OLR Challenge: Three Tasks and Their Baselines," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 11 2019, pp. 1917–1921.

[13] G. Gelly and J. Gauvain, "Spoken Language Identification Using LSTM-Based Angular Proximity," in *Proc. Interspeech 2017*, 8 2017, pp. 2566–2570.

[14] T. Kaseva, A. Rouhe, and M. Kurimo, "Spherediar: An effective speaker diarization system for meeting data," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 12 2019, pp. 373–380.

[15] D. Povey, 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," *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 2011, iEEE Catalog No.: CFP11SRW-USB.

[16] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 12 2015, pp. 167–174.

[17] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech 2018*, 9 2018, pp. 2207–2211.

[18] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6465–6469.

[19] L. Mateju, P. Cerva, J. Zdansky, and R. Safarik, "Using Deep Neural Networks for Identification of Slavic Languages from Acoustic Signal," in *Proc. Interspeech 2018*, 9 2018, pp. 1803–1807.

[20] Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic Temporal Neural Model for Language Identification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 134–144, 1 2018.

[21] Z. Ren, G. Yang, and S. Xu, "Two-Stage Training for Chinese Dialect Recognition," in *Proc. Interspeech 2019*, 9 2019, pp. 4050–4054.

[22] M. Zhao, R. Li, S. Yan, Z. Li, H. Lu, S. Xia, Q. Hong, and L. Li, "Phone-aware multi-task learning and length expanding for short-duration language recognition," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 433–437.

[23] L. Wan, P. Sridhar, Y. Yu, Q. Wang, and I. L. Moreno, "Tuple-max Loss for Language Identification," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5 2019, pp. 5976–5980.

[24] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. ACM, 2006, pp. 369–376.

[25] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[26] A. Ali, N. Dehak, P. Cardinal, S. Khurana, S. H. Yella, J. Glass, P. Bell, and S. Renals, "Automatic Dialect Detection in Arabic Broadcast Speech," in *Proc. Interspeech 2016*, 2016, pp. 2934–2938.

[27] S. Shon, A. Ali, and J. Glass, "Convolutional Neural Network and Language Embeddings for End-to-End Dialect Recognition," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 6 2018, pp. 98–104.

[28] G. Kovács, L. Tóth, D. V. Compernolle, and S. Ganapathy, "Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout," *Pattern Recognition Letters*, vol. 100, pp. 44 – 50, 2017.

[29] Z. Ma, H. Yu, W. Chen, and J. Guo, "Short Utterance Based Speech Language Identification in Intelligent Vehicles With Time-Scale Modifications and Deep Bottleneck Features," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 121–128, 1 2019.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, 5 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[32] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. Greenberg, D. Reynolds, E. Singer, L. Mason, and J. Hernandez-Cordero, "The 2017 nist language recognition evaluation," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 82–89.

[33] S. Ioffe, "Probabilistic linear discriminant analysis," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 531–542.

[34] S. Wang, Y. Qian, and K. Yu, "What Does the Speaker Embedding Encode?" in *Proc. Interspeech 2017*, 2017, pp. 1497–1501.

A.3 AALTO's speech recognition paper in Interspeech 2020 conference [2]

This paper describes AALTO's experiments in Finnish speech recognition. Deep transformer architectures, BERT and Transformer-XL, are shown to provide competitive results to our previous best LSTM-based approach [15]. We propose to use an alpha smoothing method to modify the BERT for better ASR performance. The best results are obtained with Transformer-XL trained on our morpheme-based subword units and decoded by a novel three-pass hypothesis rescoring scheme.

Finnish ASR with Deep Transformer Models

Abhilash Jain, Aku Rouhe, Stig-Arne Grönroos, Mikko Kurimo

Department of Signal Processing and Acoustics, Aalto University

firstname.lastname@aalto.fi

Abstract

Recently, BERT and Transformer-XL based architectures have achieved strong results in a range of NLP applications. In this paper, we explore Transformer architectures—BERT and Transformer-XL—as a language model for a Finnish ASR task with different rescoring schemes.

We achieve strong results in both an intrinsic and an extrinsic task with Transformer-XL. Achieving 29% better perplexity and 3% better WER than our previous best LSTM-based approach. We also introduce a novel three-pass decoding scheme which improves the ASR performance by 8%. To the best of our knowledge, this is also the first work (i) to formulate an alpha smoothing framework to use the non-autoregressive BERT language model for an ASR task, and (ii) to explore sub-word units with Transformer-XL for an agglutinative language like Finnish.

Index Terms: speech recognition, language modeling, Transformers, BERT, Transformer-XL

1. Introduction

Language modeling has the most important applications in Natural Language Processing (NLP) especially in downstreaming tasks like Automatic Speech Recognition (ASR). Recurrent Neural Networks (RNN) especially Long Short Term Memory (LSTM) networks [1] have been the typical architecture to language modeling which do achieve strong results. In spite of these results, their fundamental sequential computation constraint has restricted their use in the modeling of long-term dependencies in sequential data. To address these issues the Transformer architecture was introduced [2]. The Transformer relies completely on an attention mechanism to form global dependencies between input and output. It also offers more parallelization and has achieved the state-of-the-art (SOTA) results in language modeling outperforming LSTM models [2].

In recent years, there has been a lot of development based on basic Transformer models particularly on unsupervised pretraining [3, 4, 5, 6, 7, 8] which have set state-of-the art results on multiple NLP benchmarks. One such model architecture has been the Bidirectional Encoder Representations from Transformers (BERT) [4] model which uses a deep bidirectional Transformer architecture. Another architecture of interest is the Transformer-XL (T-XL) [5], which introduces the concept of recurrence in a self-attention model and a novel relative positional embedding scheme.

The choice of language model (LM) in ASR for the last five years has commonly been LSTM based models [9, 10, 11]. Recently, the adaptation of Transformer based models as a LM [12, 13, 14] has been proven to be very successful on improving on the earlier results. The focus though has been mostly on the English language for which abundant data is present. It is interesting to see the performance of these models for an agglutinative language like Finnish, which is morphologically richer when compared to English. In this work, we explore the implementation of two Transformer-based models—BERT and T-XL—as a LM in Finnish ASR.

Firstly, we conduct text-based experiments to see how they perform in word prediction. We take inspiration from recent works [15, 16] in investigating Deep Transformers. A sub-word based approach for both T-XL and BERT is implemented as Finnish has a very large vocabulary. With smaller units, the modeled sequences are longer, and we expect that the recursive XL architecture can allow us to still model long term effects, but avoid the Transformer's memory issues (grows quadratically with size). To the best of our knowledge, this is the first work to use subword units with T-XL for an agglutinative language like Finnish.

Secondly, we perform the Finnish ASR task on the YLE news dataset. The ASR setup is the same as in [17] which is considered the previous best. The same training data as the previous best LSTM is used to train both BERT and T-XL. We compare the Word Error Rate (WER) of all models using N-Best list rescoring. The above is done to ensure fair comparisons among different model architectures. We experiment with a novel rescoring technique which accounts for BERT's bi-directionality and the sharper predicted word probability distribution. This is one of the first works using a BERT rescoring technique with α smoothing.

We are able to successfully apply these models for ASR. T-XL obtained strong results when compared to the interpolation of LSTM+N-Gram used in [17]. Rescoring with BERT did improve the accuracy of the 1st pass ASR system, but was still behind rescoring with the LSTM model. We also develop a three-pass decoding technique where we rescore a short N-best list generated by the 2^{nd} pass LSTM+N-Gram. This technique also achieved strong results when compared to the LSTM+N-Gram and an interpolation of T-XL and LSTM+N-Gram.

2. Methods

2.1. Language Modeling - Perplexity

The goal of a LM is to assign meaningful probabilities to a sequence of words. Given a set of tokens $\mathbf{X} = (x_1, ..., x_T)$, where T is the length of a sequence, our task is to estimate the joint conditional probability $P(\mathbf{X})$ which is

$$P(\mathbf{X}) = \prod_{i=1}^{T} p\left(x_i | x_1, \ldots, x_{i-1}\right), \tag{1}$$

where $(x_1, \ldots, x_{i-1})$ is the context. An intrinsic evaluation of the performance of a LM is perplexity (PPL) which is defined as the inverse probability of the set of the tokens and taking the T^{th} root were T is the number of tokens

$$\text{PPL}(\mathbf{X}) = P(\mathbf{X})^{-1/T}. \tag{2}$$

Calculating the auto-regressive $P(\mathbf{X})$ for the T-XL is quite straight-forward as the model is unidirectional. Due to BERT's non-auto-regressive nature, the joint probability does not factorize the same way.

BERT's bi-directional context poses a challenge for us to calculate an auto-regressive joint probability. A simple workaround could be that we mask all the tokens $\mathbf{x}_{>i}$ and calculate the conditional factors as we do for an unidirectional model. By doing so though, we lose the advantage of bi-directional context the BERT model enables. We use an approximation of the joint probability as,

$$P(\mathbf{X}) \approx \prod_{i=1}^{T} p\left(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T\right). \quad (3)$$

Eq.3 is defined as a pseudo-perplexity (pseudo-PPL) score. This pseudo-PPL is used in our language modeling experiments with BERT. This type of approximations has been previously explored with Bi-directional RNN LM's [18] but not for deep Transformer models. The one drawback [18] with this approach is higher probabilities assigned to each word and one way to address this issue is to use a tunable parameter α to smooth the probability distribution. Then the word probability is,

$$p\left(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T\right) = \frac{\exp\left(\alpha y_i\right)}{\sum_j^T \exp\left(\alpha y_j\right)} \quad (4)$$

where y_i is the activation before the softmax function in the output layer. We set α to 0.6 in all our tasks after optimizing it over the development set. We apply this α smoothing technique with our BERT models during inference. This is advantageous as this technique can be applied with already existing pre-trained BERT models.

2.2. LM training with BERT and Transformer-XL

The original BERT has two training objectives: Masked language modeling (MLM), in which you mask input tokens randomly and then predict the masked tokens using the left and right context. Next, there is the 'next sentence prediction' task that jointly trains text-pair representations. We will drop this objective as it was intended for downstream tasks like 'Question & Answering' and offers little gains for other tasks [19].

T-XL is a unidirectional deep Transformer architecture, therefore the PPL can be calculated as (Eq 2). The only change is in the input format, where we use sub-word units rather than whole word units.

To remain consistent with experiments performed with the previous best LSTM [17] we use Morfessor 2.0 [20, 21] for the subword tokenization in Finnish for both BERT and T-XL. We also apply the same boundary markers- left+right-marked (+m+) markings which was the best performing sub-word marking in [17]. We use the basic unsupervised Morfessor Baseline algorithm [22] with a corpus weight parameter of 0.001. This parameter choice achieved the best result in [17].

2.3. Rescoring with BERT and Transformer-XL

To fairly compare the performances of our BERT, T-XL and LSTM models, we perform ASR experiments. Lattice rescoring is a difficult task for Bi-directional models when compared to uni-directional models [23] and therefore to simplify our approach we will be rescoring on a N-Best list (50-Best in our case). The N-Best candidates are gathered from the first-pass decoding with the same ASR system used by [17]. The N-Best candidates are then further reranked with BERT and T-XL models using the following score:

$$\text{score} = \lambda \cdot \text{score}_{\text{LM}} + \text{score}_{\text{AM}} \quad (5)$$

where score_{LM} and score_{AM} are the score of each hypothesis from the LM and AM, respectively. λ is a LM weight parameter empirically determined from a development set.

While training BERT with MLM, both the left and right context is available. More context can be advantageous to rescore utterances. In our approach, instances are created of the target utterance with one word replaced by the mask token at a time. For example, if the utterance has 4 tokens, we would create 4 instances as in Table 1.

Table 1: *Masked instances for a Finnish word like 'verkko+ +sivu+ +illa+ +an' which translates to 'on their website'*

Instance No.	Label	Masked Input
1	verkko+	[MASK] +sivu+ +illa+ +an
2	+sivu+	verkko+ [MASK] +illa+ +an
3	+illa+	verkko+ +sivu+ [MASK] +an
4	+an	verkko+ +sivu+ +illa+ [MASK]

Next, the BERT LM computes the log-likelihood using Eq.3 of the original label in the masked position. For the α smoothing BERT computation is done using the Eq.4. All the log-likelihoods of each instance is summed up and this total is defined as the score of each sentence. Even though this is not the same as the sentence probability generated by a uni-directional model, this can be used to compute score_{LM} to rescore N-best lists. Next, Eq.5 is used to rerank the N-Best list. This task is parallelized as each instance calculation is independent of each other. Also this rescoring task does not require further training and thus can be directly used with the pre-trained model. Similar approach has been used for a English BERT task [24] except they don't smooth the word probabilities as Eq.4 and they further train the model with task-specific data.

For T-XL, due to its uni-directionality the sentence probabilities are calculated using Eq.1, Next Eq.2 is used to calculate score_{LM}. Eq.5 is used to rerank the N-Best candidates. interpolation of two LM's - LM1 and LM2 is done by modifying the Eq.5 as,

$$\text{score} = \lambda 1 \cdot \text{score}_{\text{LM1}} + \lambda 2 \cdot \text{score}_{\text{LM2}} + \text{score}_{\text{AM}} \quad (6)$$

where $\lambda 1$ and $\lambda 2$ are optimized for each LM.

T-XL model is used to develop a three-pass decoding scheme. This is a pipeline strategy, where the first pass is the same as [17]. In the second pass, LSTM+N-Gram is used to rescore lattices from the first-pass to generate a shorter N-Best list. The rescoring of lattices is the same as [17]. Lastly in the third pass, T-XL is used to rescore the short N-Best list using Eq.5. The second pass prunes a majority of the less likely candidates and the heavy LM does not rescore all the possible N-best candidates in the third pass, potentially saving time and resources. Fig. 1 represents the three-pass decoding scheme.

3. Data

We use 1500 hours of speech containing manually transcribed sessions from the Finnish parliament and read speech from large number of speakers was utilized to train the Acoustic Model

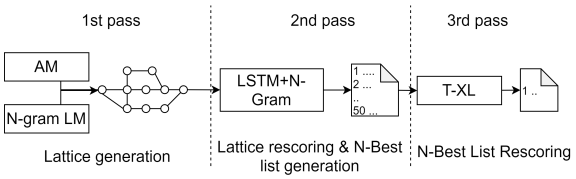


Figure 1: *Schematic diagram of three-pass decoding scheme*

(AM) as described in [17]. The Finnish text data used for all the language modeling experiments is provided by Kielipankki - the Language Bank of Finland [25]. The dataset consists mainly of newspapers and books of around 144 million word tokens and 4.2 million unique types of tokens. The data is preprocessed to remove any punctuation marks, special characters. We also convert digits to numerals. The data is randomly divided into a training and a validation dataset. The validation set contains 10K sentences. In the training data, the average token length per sentence is 21 and the maximum length is 300 per sentence. We have a total of 234 million tokens in 12.8 million sentences with a vocabulary of 34K subword tokens. The input is one sentence per line and we shuffle the sentences at each epoch. For ASR evaluation, The development and test set dataset consists of 2850 and 3006 utterances of Finnish transcribed broadcast news obtained from the Finnish national broadcaster YLE. This test set has been previously used to evaluate Finnish ASR systems in [26, 27, 17]

4. Language Modeling Experiments

All BERT and T-XL models in the experiments are trained for a maximum of 1.2 million steps and an early stopping technique is applied if the training loss keeps increasing. Adam is used as an optimizer for both BERT and T-XL experiment sets. All the experiments are trained on a single NVIDIA Tesla V100 32 GB graphic card unlike the multi-gpu setup used by the English models [4, 5]. Only the models fitting on a single -gpu were selected to keep resource demands low. The previous best was an interpolated LSTM+N-Gram. The LSTM model is taken from [17], it is a 'deep' architecture which starts from a projection and a LSTM layer that has four pairs of dropout and highway layers. The large N-Gram is also taken from [17] and contains 50-80 million n-gram contexts. Hence-forth, we call this interpolation as **LSTM+N-Gram**.

4.1. BERT

BERT models are very resource intensive to train, and therefore our hyperparameter search space is influenced by the original BERT [4]. We also tried to increase the depth rather than width as suggested in [15]. All the BERT models are pre-trained from scratch and the pseudo-perplexities are calculated on the development set. Different configurations using the development set are tried out, finally settling on the configurations in Table 2.

Even though the BERT model pseudo-PPL cannot be directly compared with the T-XL PPL, the pseudo-PPL can still be used to intrinsically evaluate between different BERT models. The 20 layer BERT model from Table 2 performs the best among all the BERT models after α smoothing.

Table 2: *Pseudo-perplexities calculated on the test set for different configurations of the BERT model with layers (L), feed-forward layer size (FF), hidden Transformer size (H) and attention heads (A)*

L	FF	H	A	Pseudo PPL W/o α	Pseudo PPL With α
10	1024	360	8	62.11	952.2
12	3072	768	12	18.86	49.29
4				20.35	45.1
6				19.74	44.31
8	3584	896	16	11.56	28.51
10				**11.48**	28.33
20				11.84	**28.27**

Table 3: *Perplexities calculated on the test set with different model configurations with layers (L), feed-forward layer size (FF), hidden Transformer size (H)*

LM	L	FF	H	seg-mem	PPL
LSTM+N-Gram					93.2
Transformer	18	4096	1024		78.7
	3	2048	512	150-150	89.6
	4	2048	512	150-150	82.3
	4	4096	1024	32-32	75.1
T-XL	8				85.4
	16				75.3
	32	1024	256	32-32	68.7
	64				67.1
	72				**66.3**

4.2. Transformer-XL

Because T-XL utilizes previous context to calculate its attention as explained in [5] the selection of contexts is an important hyperparameter. They are seg-length(seg) and mem-length(mem) as explained in [5] As training T-XL is very resource and time intensive, we focus on comparing a wider context but a shallower model against a narrower context but a deeper model.

The same cosine annealing learning rate scheduler and relative positional embeddings as [5] are used in all of the T-XL experiments. A baseline Transformer model is trained with the same self-attention as BERT and no previous context, the hyperparameters resemble the one's in [5]. All T-XL models are trained from scratch. The perplexities are calculated on the development set and the results are in Table 3.

From Table 3 the T-XL with 72 layers achieved a PPL score of 66.3 achieving a 29% better score than LSTM+N-Gram.

5. ASR Experiments

5.1. Two-pass decoding

First a conventional rescoring scheme is a applied to use the LMs trained in the previous section to rescore a 50-best list in the YLE news dataset. In preliminary experiments we found 50 to be a suitable compromise between speed and accuracy for all LMs. The AM is a TDNN-BLSTM trained with lattice-free MMI and the small first-pass N-gram LM contains approximately 5M N-grams and both the models are from [17]. Table 4

Table 4: *WERs (%) on different LMs obtained by 50-best rescoring (except the 5M N-gram LM)*

LM	WER	
	dev	test
5M N-gram	17.56	25.41
LSTM+N-Gram	14.19	20.57
Baseline Transformer	13.93	20.23
BERT (without α smoothing)		
L-10 FF-3584 H-896 A-16	16.35	25.62
L-20 FF-3584 H-896 A-16	16.25	25.72
BERT (with α smoothing)		
L-10 FF-3584 H-896 A-16	15.39	25.21
L-20 FF-3584 H-896 A-16	15.34	25.26
T-XL		
L-4 FF-4096 H-1024	14.63	21.23
L-8 FF-1024 H-256	14.11	20.54
L-16 FF-1024 H-256	13.81	20.24
L-32 FF-1024 H-256	13.78	**20.04**
L-48 FF-1024 H-256	13.74	20.16
L-64 FF-1024 H-256	13.72	**20.04**
L-72 FF-1024 H-256	**13.67**	20.12

shows that the best T-XL outperforms the LSTM+N-Gram by 5% on the dev set and 3% on the test set. The BERT models do outperform the 5M N-gram but still cannot get a better WER than LSTM+N-Gram. There is an incremental gain in WER by applying the α smoothing. The pseudo-PPL with α and WER results appear to correlate. Therefore, we can potentially use the pseudo-PPL with α to faster optimize the hyperparameters without using the model on ASR.

5.2. Three–pass decoding scheme

A three-pass decoding scheme was tested for the best performing T-XL's from the Sec. 5.1. On the first pass of the ASR, A TDNN-BLSTM AM and a 5M N-gram LM same as Sec.5.1 is used. In the second pass, LSTM+N-gram rescores first pass lattices generating a new 50-best list. In the third pass the 50-best list is rescored using T-XL. We also perform interpolating experiments by using the LM scores from LSTM+N-gram and T-XL after each of them have rescored a 1000-best list generated from the first pass. In preliminary experiments we found that 1000 is a good compromise between speed and accuracy. With T-XL as LM1 and LSTM+N-gram as LM2 in Eq.6, optimal λ_1=2λ_2. This signifies that T-XL is twice more impactful than LSTM+N-gram in the interpolation results. From Table 5, our best three-pass decoding scheme outperforms the Interpolated LSTM+N-gram+T-XL by 4%.

6. Discussion

BERT-based architectures [28, 29] have set very good benchmarks in a variety of tasks like part of speech tagging, named entity recognition, Question & Answering. However, they have not attracted as much attention in ASR, where their results have been much behind the best [24]. This may be due to the fact that BERT has a broader objective like sentence encoding due to the masked LM training and ASR requires the LM to have a auto-regressive training component. Nonetheless, this could be leveraged as during rescoring we do have the entire utterance at hand. Thus, a more efficient rescoring scheme could be developed in the future to take advantage of both T-XL and BERT.

Table 5: *WERs (%) of the following decoding models:*
[1]*rescoring a 1000-best,*
[2]*interpolation after each rescoring 1000-best,* $\lambda_{T\text{-}XL} = 2\lambda_{LSTM}$
[3]*three-pass decoding with LSTM+N-gram rescoring a 1000-best list followed by T-XL rescoring a 50-best, and*
[4]*three-pass decoding with LSTM+N-Gram rescoring a lattice followed by T-XL rescoring a 50-best list*

LM	WER	
	dev	test
[1]**LSTM+N-Gram**	13.68	19.3
[1]**T-XL**		
L-32 FF-1024 H-256	13.10	18.65
L-64 FF-1024 H-256	13.00	18.50
L-72 FF-1024 H-256	12.95	18.63
[2]**T-XL and LSTM+N-Gram (Interpolated)**		
L-32 FF-1024 H-256	12.76	18.54
L-64 FF-1024 H-256	12.78	18.38
L-72 FF-1024 H-256	12.77	18.40
[3]**1000-best LSTM+N-Gram + 50-best T-XL**		
L-32 FF-1024 H-256	13.07	18.51
L-64 FF-1024 H-256	13.02	18.42
L-72 FF-1024 H-256	12.85	18.56
[4]**Lattice LSTM+N-Gram + 50-best T-XL**		
L-32 FF-1024 H-256	12.76	18.05
L-64 FF-1024 H-256	12.68	17.72
L-72 FF-1024 H-256	**12.59**	**17.71**

In our experiments, T-XL performs better when compared to the LSTM+N-Gram both in the intrinsic and extrinsic tasks. T-XL's encapsulation of the recurrence mechanism is advantageous for long sequences of subword units. LSTM+N-Gram and T-XL are also used in the three-pass decoding scheme outperforming both the individual models and the interpolated models on the N-Best lists. In comparison to the interpolated models, the three-pass decoding scheme runs the heavy LM rescoring on a pruned set of hypothesis saving time and space.

7. Conclusion

We apply BERT and T-XL LMs for speech recognition. We show that Transformer-XL outperforms LSTM+N-Gram on perplexity by 29% and ASR by 3% with the same data. We propose a three-pass decoding scheme which successfully approximates the interpolation of LSTM+N-Gram + T-XL and avoids running the large slow T-XL language model for the full 1000-best hypothesis as in the interpolation. We propose a framework for pre-trained BERT models along with α smoothing which can be used for ASR. We believe it is possible to improve the performance of these models by using more data, applying more regularization techniques and scaling them up.

8. Acknowledgements

This work was supported by the Academy of Finland (grant 329267) and EU's Horizon 2020 research and innovation programme via the project MeMAD (GA 780069). The computational resources were provided by Aalto ScienceIT.

9. References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[3] A. Radford, "Improving language understanding by generative pre-training," 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[5] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *ACL*, 2019.

[6] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 5753–5763. [Online]. Available: http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

[7] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[8] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.

[9] K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane, "The CAPIO 2017 Conversational Speech Recognition System," *arXiv e-prints*, p. arXiv:1801.00059, Dec 2017.

[10] G. Saon, T. Sercu, S. Rennie, and H.-K. J. Kuo, "The ibm 2016 english conversational telephone speech recognition system," *Interspeech 2016*, Sep 2016. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2016-1460

[11] G. Kurata, B. Ramabhadran, G. Saon, and A. Sethy, "Language modeling with highway lstm," *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017. [Online]. Available: http://dx.doi.org/10.1109/ASRU.2017.8268942

[12] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention," *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1780

[13] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, and M. L. Seltzer, "Transformer-based acoustic modeling for hybrid speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6874–6878.

[14] I. Medennikov, Y. Khokhlov, A. Romanenko, I. Sorokin, A. Mitrofanov, V. Bataev, A. Andrusenko, T. Prisyach, M. Korenevskaya, O. Petrov, and A. Zatvornitskiy, "The STC ASR System for the VOiCES from a Distance Challenge 2019," in *Proc. Interspeech 2019*, 2019.

[15] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Proc. Interspeech 2019*, 2019, pp. 3905–3909. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2225

[16] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An End-to-End Convolutional Neural Acoustic Model," in *Proc. Interspeech 2019*, 2019, pp. 71–75. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1819

[17] P. Smit, "Modern subword-based models for automatic speech recognition," ser. Aalto University publication series DOCTORAL DISSERTATIONS; 97/2019. Aalto University; Aaltoyliopisto, 2019, G5 Artikkeliväitöskirja, pp. 62 + app. 136. [Online]. Available: http://urn.fi/URN:ISBN:978-952-60-8566-1

[18] X. Chen, A. Ragni, X. Liu, and M. Gales, "Investigating bidirectional recurrent neural network language models for speech recognition," 08 2017, pp. 269–273.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[20] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," in *ACL-02 Workshop on Morphological and Phonological Learning*, ser. MPL '02, vol. 6. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 21–30. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1118647.1118650

[21] S. Virpioja, P. Smit, S.-A. Grönroos, and M. Kurimo, "Morfessor 2.0: Python implementation and extensions for morfessor baseline," Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland, Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, 2013.

[22] M. Creutz and K. Lagus, "Unsupervised models for morpheme segmentation and morphology learning," *ACM Trans. Speech Lang. Process.*, vol. 4, no. 1, Feb. 2007. [Online]. Available: https://doi.org/10.1145/1187415.1187418

[23] X. Liu, Y. Wang, X. Chen, M. J. F. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4908–4912.

[24] J. Shin, Y. Lee, and K. Jung, "Effective sentence scoring method using bert for speech recognition," in *Proceedings of The Eleventh Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, W. S. Lee and T. Suzuki, Eds., vol. 101. Nagoya, Japan: PMLR, 17–19 Nov 2019, pp. 1081–1093.

[25] CSC - IT Center for Science, "The helsinki korp version of the Finnish text collection, url: http://urn.fi/urn:nbn:fi:lb-2016050207," 1998. [Online]. Available: http://urn.fi/urn:nbn:fi:lb-2016050207

[26] A. Mansikkaniemi, P. Smit, and M. Kurimo, "Automatic construction of the finnish parliament speech corpus," in *Proc. Interspeech 2017*, 2017, pp. 3762–3766. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-1115

[27] M. Varjokallio, S. Virpioja, and M. Kurimo, "First-pass techniques for very large vocabulary speech recognition of morphologically rich languages," in *2018 IEEE Spoken Language Technology Workshop, December 18-21, 2018, Athens, Greece*. United States: IEEE, 2018, pp. 227–234.

[28] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[29] A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo, "Multilingual is not enough: Bert for finnish," *ArXiv*, vol. abs/1912.07076, 2019.

A.4 AALTO's PicSOM team's submission to TRECVID 2020 [3]

This paper describes the participation of AALTO's PicSOM team in TRECVID 2020 VTT video caption generation task. A new stacked attention architecture is proposed and the results of the evaluation show that it generates captions that are better than those produced by a plain LSTM model.

PicSOM Experiments in TRECVID 2020

Pre-workshop draft – Revision: 1.5

Jorma Laaksonen and Zixin Guo

Department of Computer Science
Aalto University School of Science
P.O.Box 15400, FI-00076 Aalto, Finland
firstname.lastname@aalto.fi

Abstract

This year, the PicSOM team participated only in the Video to Text Description (VTT), Description Generation subtask. In total, the PicSOM team submitted four runs. We had two goals in our submissions, first, to study the performance of our recent developments in the architectures of the captioning model, and second, to see the effect of using the VATEX dataset in model training. The submitted four runs are as follows:

- PicSOM.1.PRIMARY: Our latest and best stacked attention model, trained with three datasets.
- PicSOM.2: Model architecture similar to our best VTT 2019 submission, trained with three datasets.
- PicSOM.3: Another well-performing stacked attention model, trained with two datasets.
- PicSOM.4: Model architecture similar to our best VTT 2019 submission, trained with two datasets.

The runs aim at comparing different implementations of stacked attention on the visual features and the benefit from using the VATEX dataset. Based on our results we can conclude that the use of the VATEX dataset had more effect on the improvement of the results than the stacked attention, which also produced small but noticeable improvement. Based on the results of the runs, it seems that our latest attention model combined with self-critical reinforcement learning was the best approach.

I. Introduction

In this notebook paper, we describe the PicSOM team's experiments for the TRECVID 2020 evaluation [1]. We participated only in the Video to Text Description (VTT) subtask Description Generation. Our approaches are variations of the "Show and tell" model [2], augmented with a richer set of contextual features [3] and self-critical training [4]. The captioning models are described in more detail in Section II and their used training loss functions in Section III. Then, we describe the features in Section IV and the datasets used for training in Section V. In Section VI we intoduce our stacked attention model. Our experiments, submitted runs and results are discussed in Section VII and conclusions are drawn in Section VIII.

II. DeepCaption neural captioning model

The PicSOM team's LSTM [5] model has been implemented in PyTorch and is available as open source.[1] The features are translated to the hidden size of the LSTM by using a fully connected layer. We apply dropout and batch normalization [6] at this layer. As the loss function, we similarly use cross entropy, in addition to Reinforcement Learning with self-critical loss function [4] in order to fine-tune a well-performing model. The fine-tuning is implemented either by switching to the self-critical loss in training time or by specifying a pre-trained model to load and fine-tune.

III. Training loss functions

In order to train the architecture so that its output distribution approximates the target distribution at each decoding step t, several optimisation objectives are used. Recent progress on sequence training enables new optimisation paradigms, which are applied and compared in this work.

[1] https://github.com/aalto-cbir/DeepCaption

A. Cross-entropy

Traditionally, the teacher forcing algorithm [7] is the most common method to maximise the log-likelihood of a model output X to match the ground truth $y = \{y_1, y_2, \cdots, y_T\}$. It minimises the cross-entropy objective

$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} \log p_\theta \left(y_t \mid y_{t-1}, \mathbf{h}_{t-1}, X \right), \qquad (1)$$

where $\mathbf{h}_{t-1}$ is the hidden state of the RNN from the previous step and p_θ the probability of an output parametrized by θ. In the inference time, the output can be produced simply by greedy sampling of the sequence being generated.

B. Self-critical

Lately, Reinforcement Learning ideas have been used to optimise a captioning system based on recurrent neural network language models. Such a system can be seen as an agent taking actions according to a policy π_θ and outputting a word $\hat{y}_t$ as an action.

One proposed approach is the self-critical algorithm [4], where the output at inference time of the model $\hat{y}^g_{i,t}$ is used, normally applying greedy search. The sequences are scored using a reward function r. Thanks to the properties of this optimisation, NLP metrics can be used as reward to affect

the actual loss. In our case, CIDErD [8] is used. The final objective reads

$$\mathcal{L}_\theta = \frac{1}{N}\sum_{i=1}^{N}\sum_t \log \pi_\theta\left(\hat{y}_{i,t} \mid \hat{y}_{i,t-1}, \mathbf{s}_{i,t}, \mathbf{h}_{i,t-1}\right) \cdot \left(r(\hat{y}_{i,1}, \cdots, \hat{y}_{i,T}) - r(\hat{y}^g_{i,1}, \cdots, \hat{y}^g_{i,T})\right) . \quad (2)$$

IV. Features

Table I summarizes the features used in our experiments and their dimensionalities.

TABLE I
SUMMARY OF THE FEATURES USED IN OUR EXPERIMENTS.

abbr.	feature	dim.	modality
rn	ResNet-152	2048	image
i3d	I3D	2048	video
fake-i3d	Fake I3D	2048	image

A. ResNet-152

We are using pre-trained CNN features from ResNet-152 so that the 2048-dimensional features from the pool5 layer averaged across five crops from the original and horizontally flipped images. When applied to a video object, we have used the middlemost frame of the video.

B. I3D

To encode video features, we adopted Inflated 3D Convolutional Network (I3D) [9]. It builds upon already competent image recognition models (2D) and inflates the filters and kernels to 3D, thus creating an additional temporal dimension. Concretely, the base network used is ImageNet-pretrained Inception-V1 [10] using two streams [11]. The videos were first resampled to 25 frames per second as in the original I3D paper and 128 frames were taken from the center. For DeepCaption, the extractor is applied convolutionally over the whole video and the output is average-pooled in order to produce a 2048-dimensional feature vector.

C. Fake I3D

When we used still images of the COCO dataset for training a captioning model, we naturally were not able to extract and use I3D features for those images. Therefore we had calculated the average value of the I3D feature vectors in the TGIF dataset and used that vector as a "fake I3D" feature for all COCO images.

V. Training data

Table II gives a summary of the databases and the features we have extracted for them. In Tables II and III, we have shortened the dataset names with one letter abbreviations.

A. COCO

The *Microsoft Common Objects in COntext (MS COCO)* dataset [12] has 2,500,000 labeled instances in 328,000 images, consisting on 80 object categories. COCO is focused on non-iconic views (or non-canonical perspectives) of objects, contextual reasoning between objects, and precise 2D localization of objects.

TABLE II
SUMMARY OF THE TRAINING DATASETS USED IN OUR EXPERIMENTS.

	dataset	items	captions	features
C	COCO	82,783 img	414,113	rn fake-i3d
T	TGIF	125,713 vid	125,713	rn i3d
V	VATEX	41,250 vid	825,000	rn i3d

B. TGIF

The *Tumblr GIF (TGIF)* dataset [13] contains 100,000 animated GIFs and 120,000 natural language sentences. This dataset aims to provide motion information involved between image sequences (or frames).

C. VATEX

As an addition to the training datasets we have used earlier, we have now started to use the new VATEX video captioning dataset [14]. VATEX contains over 41,250 videos and 825,000 captions in both English and Chinese.

VI. Stacked attention

Figure 1 shows the overall architecture of DeepCaption's new caption generation model. The visual features and captions are treated as inputs to the encoding and decoding layers, and the last output of the decoding layers are processed by an LSTM. All the outputs from the decoding layers are collected and used to attend the representations generated by the recurrent language model to produce the output words.

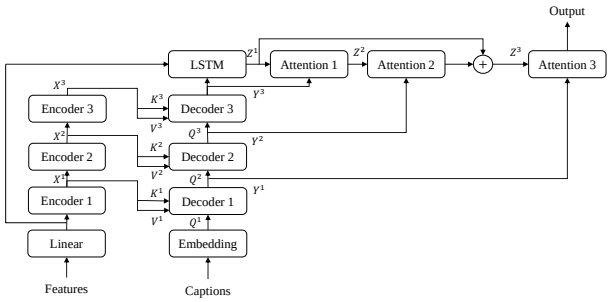


Fig. 1. The architecture of DeepCaption's new stacked attention model.

The stacked attention model is based on the Transformer model [15], in which the intra- and cross-relations between the visual and the text features are calculated via scaled dot-product attention. The attention function receives three sequential sets with length s, and d_{model} dimensions, denoted as queries Q, keys K, and values V. The attention function is defined as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V \ , \quad (3)$$

where $Q \in \mathbb{R}^{s \times d_{model}}$ is a matrix of query vectors and K and $V \in \mathbb{R}^{s \times d_{model}}$ are matrices of key and value vectors. Given a set of features from videos, intra-modality attention is obtained in the encoder with self-attention on the different

feature inputs. Cross-modality dependencies are modeled in the decoder via cross-modal attention operations between the visual and textual features. Multihead attention is employed for improving the feature representation and with k heads it is formulated as

$$Multihead(Q, K, V) = concat(h_1, \ldots, h_k)W^O \quad (4)$$

$$h_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (5)$$
$$i = 1, \ldots, k \ ,$$

with matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_{model}/k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_{model}/k}$ and $W_i^V \in \mathbb{R}^{d_{model} \times d_{model}/k}$ used in each of the k attention heads, and $W^O \in \mathbb{R}^{d_{model} \times d_{model}}$.

In our stacked attention model, we have used the depth of $N = 3$ layers, as seen in Figure 1. Both the encoding and decoding layers are stacked sequentially. The stack of N encoding layers generates multi-level outputs $X = (X^1, \ldots, X^N)$ which are used as the key and value inputs, K and V, of the cross-modal attention in each corresponding decoder. The query inputs Q come there from the word embeddings of the caption. The multi-level cross-modal relations of visual and textual features provide refined inputs for the attention on the recurrent language model. The decoding layers depend on the visual features and the previously generated words. We collect them and exploit the outputs level by level.

The stacked attention mechanism always uses the decoder output Y^{N-j+1} to attend the attention-stacked LSTM output Z^j. First, with $j = 1$ and given the decoder output Y^N and the LSTM output Z^1, the stacked attention mechanism concatenates Y^N and Z^1 and transforms them linearly to the same dimension with Z. The stacked attention is then defined as element-wise or Hadamard product

$$StackedAttention(Y^{N-j+1}, Z^j) = \alpha(Y, Z) \odot Z \ , \quad (6)$$

where we have dropped the superscripts on the right for clarity and $\alpha(\cdot, \cdot)$ is a function that generates a element-wise multiplication matrix which has the same dimensions as Z. The function $\alpha(\cdot, \cdot)$ is defined as

$$\alpha(Y, Z) = \sigma(W \, [Y, Z] + b) \ , \quad (7)$$

where we have dropped the superscripts on the right for clarity and $\alpha(\cdot, \cdot)$ is a function that generates a element-wise multiplication matrix which has the same dimensions as Z. The function $\alpha(\cdot, \cdot)$ is defined as

$$\alpha(Y, Z) = \sigma(W \, [Y, Z] + b) \ , \quad (8)$$

where $[\cdot, \cdot]$ stands for concatenation, $\sigma(\cdot)$ is the sigmoid function, and W and b are the weight and the bias. The stacked attention for the full sequence of LSTM outputs is then formed by applying the attention (6) sequentially with $j = 1, \ldots, N$. As can be seen in Figure 1, we have additionally utilized a skip connection from the LSTM output Z^1 to Z^3.

Word-level cross-entropy (XE) is used to pre-train the model, which is then fine-tuned via reinforcement learning. During the XE training, the model predictions are conditioned on the previous annotated words. Training with reinforcement learning employs the self-critical (SC) [16] training method. During the decoding, both greedy and stochastic samples of the output sequences are used at each time step. We employ the CIDEr-D [17] score as the reward of the SC reinforcement learning. The reward is baselined by a greedy sample rather than the mean of rewards. The gradient is then defined as

$$\nabla_\theta L(\theta) = -\frac{1}{M} \sum_{i=1}^{M} \left((r(w^i) - r(\hat{w})) \nabla_\theta \log p(w^i) \right) \ , \quad (9)$$

where w^i is the i-th stochastic sample in a batch, $\hat{w}$ is the greedy search sample and $r(\cdot)$ is the CIDEr-D reward function. When predicting, we perform greedy search and keep words with the highest predicted probabilities within the vocabulary.

VII. Experiments and results

During the development stage, we mostly kept the selection of the trainig datasets and features fixed and concentrated on selecting the best model architecture and training straregy. We evaluated our results using the previously released ground truth of TRECVID VTT 2018 test set. The four runs submitted are identified as "s1" to "s4" in Table III.

Our four runs were mutually different fromeach other in the following:

- s1: Our latest and best stacked attention model, trained with all three dtasets: COCO, TGIF and VATEX.
- s2: Model architecture similar to our best VTT 2019 submission, trained with all three datasets.
- s3: Another well-performing stacked attention model, trained with two datasets: COCO and TGIF.
- s4: Model architecture similar to our best VTT 2019 submission, trained with two datasets: COCO and TGIF.

Based on evaluation on the TRECVID VTT 2018 and 2019 test sets, we ended up using a 2-layer LSTM for DeepCaption with an embedding vector size of 512, and 1024 for the hidden state dimensionality in all PicSOM team's runs. Both in the input translation layer and in the LSTM we applied a dropout of 0.5. We used Adam optimiser [18] for the self-critical stage with a learning rate of 5×10^{-5} and no weight decay. Additionally, gradient clipping is performed when a range $[-0.1, 0.1]$ is exceeded. The models were pretrained using centered RMSprop [19] with a learning rate of 0.001 and weight decay (L2 penalty) of 10^{-6}.

Our results compared to those of the other submitted runs are visualized with bar charts for each automatic performance measure in Figures 2–7.

VIII. Conclusions

There were two main research questions in the PicSOM team's set of four submissions. We wanted to see the benefit we could get from our novel stacked attention model in our DeepCaption captioning system. The results with the stacked attention model were quite systematically better than those without it, but the advantage was quite limited. We also studied, how much the intriduction of the VATEX dataset as an additional training data to COCO and TGIF improved the results. In this case the positive effect was clear and the

TABLE III
RESULTS OF OUR SUBMISSIONS S1,. . . ,4.

	2020					
id	**METEOR**	**CIDEr**	**CIDErD**	**BLEU**	**SPICE**	**STS**
s1	**0.2617**	**0.319**	**0.200**	*0.0527*	**0.079**	0.4406
s2	*0.2556*	*0.312*	*0.191*	**0.0536**	*0.076*	*0.4293*
s3	0.2414	0.278	0.129	0.0485	0.069	**0.4581**
s4	0.2323	0.278	0.124	0.0201	0.067	0.4458

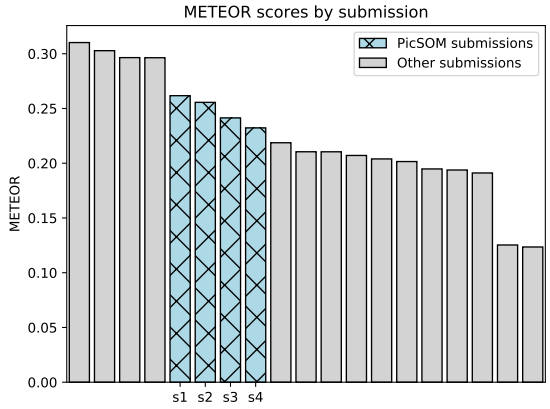


Fig. 2. METEOR results of the PicSOM team and others.

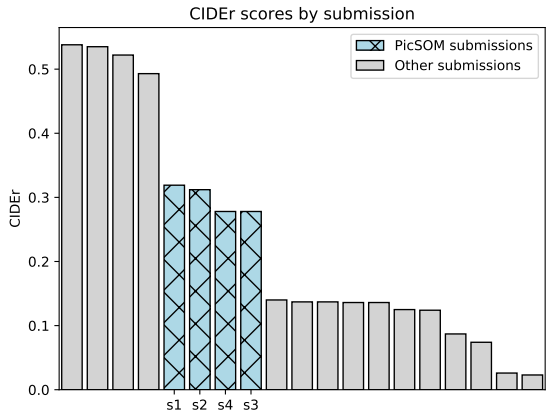


Fig. 3. CIDEr results of the PicSOM team and others.

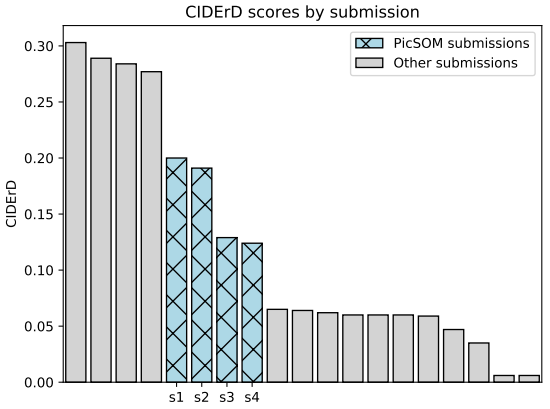


Fig. 4. CIDErD results of the PicSOM team and others.

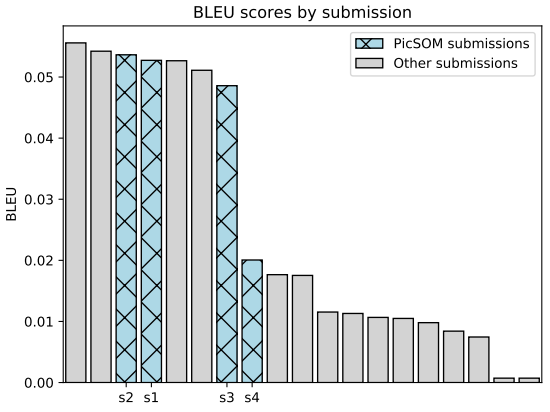


Fig. 5. BLEU results of the PicSOM team and others.

our results were clearly improved from the performance level where we were in the last year's submissions.

ACKNOWLEDGMENTS

This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780069 *Methods for Managing Audiovisual Data: Combining Automatic Efficiency with Human Accuracy* (MeMAD). This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence, FCAI. The calculations were performed using computer resources provided by the Aalto University's *Aalto Science IT* project and CSC – IT Center for Science Ltd. We also acknowledge the support of NVIDIA Corporation with the donation of TITAN X and Quadro P6000 GPUs used for parts of this research.

REFERENCES

[1] George Awad, Asad A. Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Jesse Zhang, Eliot Godard, Lukas Diduch, Jeffrey Liu, Alan F. Smeaton, Yvette Graham, Gareth J. F. Jones, Wessel Kraaij, and Georges Quénot. Trecvid 2020: comprehensive campaign for evaluating video retrieval tasks across multiple application domains. In *Proceedings of TRECVID 2020*. NIST, USA, 2020.

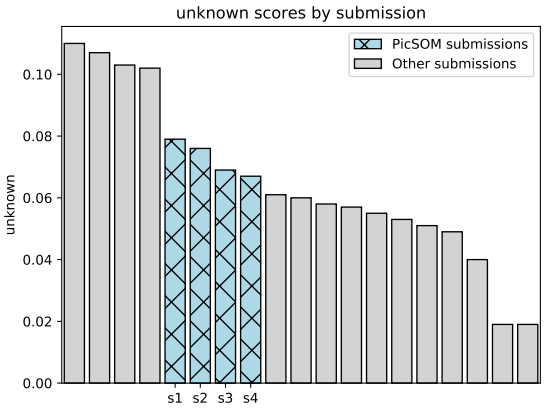


Fig. 6. SPICE results of the PicSOM team and others.

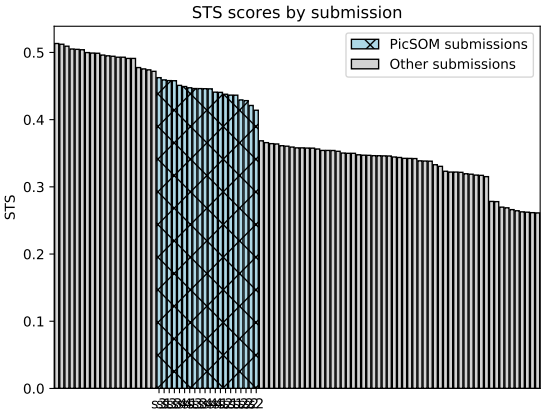


Fig. 7. STS results of the PicSOM team and others.

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[3] Rakshith Shetty, Hamed R.-Tavakoli, and Jorma Laaksonen. Image and video captioning with augmented neural architectures. *IEEE MultiMedia*, 25(2):34–46, April 2018.

[4] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[7] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015.

[8] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[9] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[11] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573, 2016.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[13] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. TGIF: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2016.

[14] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research, 2019.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[16] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.