PUNCTUATED TRANSCRIPTION OF MULTI-GENRE BROADCASTS USING ACOUSTIC AND LEXICAL APPROACHES

Ondřej Klejch, Peter Bell, Steve Renals

Centre for Speech Technology Research, University of Edinburgh, Edinburgh EH8 9AB, UK
o.klejch@sms.ed.ac.uk, {peter.bell,s.renals}@ed.ac.uk

ABSTRACT

In this paper we investigate the punctuated transcription of multi-genre broadcast media. We examine four systems, three of which are based on lexical features, the fourth of which uses acoustic features by integrating punctuation into the speech recognition acoustic models. We also explore the combination of these component systems using voting and log-linear interpolation. We performed experiments on the English language MGB Challenge data, which comprises about 1,600h of BBC television recordings. Our results indicate that a lexical system, based on a neural machine translation approach is significantly better than other systems achieving an F-Measure of 62.6% on reference text, with a relative degradation of 19% on ASR output. Our analysis of the results in terms of specific punctuation indicated that using longer context improves the prediction of question marks and acoustic information improves prediction of exclamation marks. Finally, we show that even though the systems are complementary, their straightforward combination does not yield better F-measures than a single system using neural machine translation.

***Index Terms*—** punctuation, speech recognition, neural machine translation, rich transcription

1. INTRODUCTION

Automatic speech recognition (ASR) systems typically process a stream of audio to produce transcripts which are uncased and without punctuation marks or sentence boundaries. However, casing and punctuation are very important for transcript readability, as well as for further natural language processing such as summarization, information extraction or machine translation. In this paper we explore a number of approaches for the punctuated transcription of multi-genre broadcast recordings, using broadcast media data obtained from a wide range of BBC television programmes that was used in the first MGB Challenge [1].

We addressed three principal research questions:

1. Is acoustic context necessary to predict punctuation?
2. Does the provision of longer context information in a punctuation prediction model improve the recall or precision by enabling long distance dependencies to be modelled? (For instance modelling the relation of sentence-initial "Wh" words with [?]? And if so, how can this context be effectively modelled?
3. Can we improve recall and precision through the automatic combination of different systems for punctuation prescription?

In this paper we investigate four methods of punctuation prediction using both lexical and acoustic features. The lexically-based methods are based on statistical language modelling using n-grams, word-based punctuation labelling using a recurrent neural network, and a novel approach derived from neural machine translation (NMT). To incorporate acoustic data we have explored an ASR-based approach in which punctuation marks are appended to the lexicon and "consume" audio. We also combine these approaches using voting and log-linear interpolation. We performed experiments on both reference and ASR-generated transcripts, analysing system performance across the five different punctuation marks that we predict. Our results indicate that the NMT-based approach results in the best performing systems (based on the F-Measure combination of precision and recall).

2. RELATED WORK

Automatic sentence boundary detection and punctuation have now been explored for over 20 years. Stolcke and Shriberg published some pioneering papers using n-gram language models to predict sentence boundaries in speech transcripts [2, 3]; a similar language modelling approach was used by Beeferman et al [4] to insert commas in pre-segmented sentences. Chen [5] presented an approach in which punctuation prediction was integrated into the speech recognition decoder, combining both acoustic and lexical features to predict a range of punctuation marks.

Since then there has been a wide range of approaches to the problem which may be categorized in terms of the way punctuation marks are modelled, and the features used for the prediction.

2.1. Statistical punctuation models

There are three main approaches to modelling punctuation marks. First, punctuation marks may be treated as inter-word events, predicted using statistical language models [2, 3, 4, 6, 7] or by finite state or hidden Markov models [8, 9]. In both cases these are trained from a corpus of punctuated text, and the Viterbi algorithm is used to obtain the most probable sequence of words and punctuation marks. Second, transcript punctuation can be viewed as a word labeling task, in which a punctuation label is assigned to each word. A variety of methods have been used for this approach, in which punctuation is treated as a classification task, including maximum entropy modeling [10], boosting [11, 12], conditional random fields [13, 14] and neural networks [9, 15, 16, 17].

Finally, transcript punctuation can be viewed as a monolingual machine translation problem in which the source is unpunctuated text and the target is punctuated text. Phrase-based statistical machine translation approaches have been used in this way [18, 19, 20].

2.2. Features for automatic punctuation

All the above methods can potentially use different types of features for punctuation: most published approaches either use purely lexical features or attempt to combine acoustic and lexical features. The most commonly used lexical features for punctuation prediction are n-gram statistics obtained from the language model, but part-of-speech tags [15] and syntax information from a sentence parse tree [14] have also been used. Acoustic features for transcript punctuation are usually suprasegmental prosodic features, often pause duration [9], but also features relating to phoneme duration, fundamental frequency, and energy [3, 6, 9, 11, 21].

3. SYSTEMS

We have implemented and evaluated four systems for punctuated transcription based on (1) n-gram language modelling, (2) word labeling using recurrent networks, (3) a variant of neural machine translation, and (4) automatic speech recognition treating punctuation marks as additional words.

3.1. Language modeling

To use an n-gram statistical language model trained on a large corpus for punctuation annotation [3, 4, 6, 7], we constructed a hyper-string weighted finite state acceptor (WFSA) which is a compact representation of the search space of all possible punctuation marks (see Fig. 1 for an example). The hyper-string WFSA was composed with a weighted finite state transducer (WFST) representing a statistical language model, which adds weights to all possible sequences of words and punctuation marks. The best punctuation is then obtained by selecting the lowest cost path in the composed transducer [7].

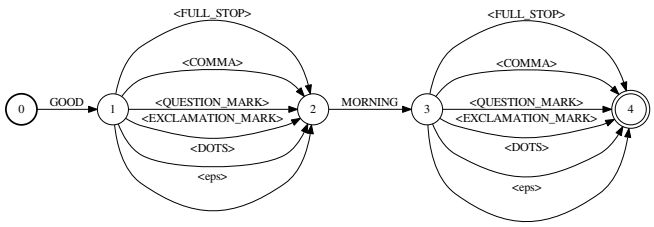

Fig. 1. Example of hyper-string WFSA for string "GOOD MORNING"

Since this method uses finite context it cannot model those cases in which longer context is needed for the prediction of punctuation marks – for instance when indicative words for an end of sentence punctuation mark appear at the beginning of the sentence. Experiments have indicated that simply increasing the n-gram context does not increase the punctuation accuracy [7]. Additionally, this method is vulnerable to data sparsity – smoothed language models do not predict well punctuation marks for sequences that were not seen in the training data.

3.2. Word labeling

The word labeling approach is based on a classifier which maps lexical input to a punctuation mark label for each word. If a recurrent network is used for classification then it is possible to incorporate arbitrary context. Tilk and Alümae [16, 17] developed word labeling systems using unidirectional and bidirectional long short-term memory (LSTM) neural networks [22]. In this paper we used the Punctuator 2 system [17], which uses a bidirectional LSTM network with an attention layer to predict the punctuation mark (or no punctuation) for each word. Although this method is able to model long distant lexical context, it assumes that the generated punctuation marks are independent – i.e. there is no prior sequence model of punctuation labels.

3.3. Neural Machine Translation

The LSTM systems discussed above map an input sequence to a label, using the hidden state of the recurrent network to learn the context. To address the problem of label independence, recurrent neural networks can also be used for sequence-to-sequence mapping using an encoder-decoder architecture [23]. This architecture, typically with an attention layer, has been used successfully for machine translation [24, 25], where it is often referred to as neural machine translation (NMT). We apply NMT for punctuation by mapping from a high dimension input sequence (the sequence of words) to a low dimension label sequence (the sequence of punctuation marks – including blank). This is in contrast to previous applications of statistical phrase based machine translation to punctuation [18, 19, 20], in which unpunctuated

text is mapped to punctuated text. Mapping to the punctuation marks only is much more efficient, because the output layer of the neural network is smaller by several orders of magnitude and the punctuation process cannot introduce new lexical errors.

3.4. Automatic Speech Recognition

The previously described systems are purely text based and thus cannot model acoustic information or speech prosody, which has a strong relation to punctuation, for instance when predicting [!]. Therefore we implemented an automatic speech recognition system which treats punctuation marks as additional words, with a one "phoneme" pronunciation, with a unique phoneme assigned to each type of punctuation mark. The resultant punctuation aware speech recognition system may be used in two ways: first, it may be used directly to decode a stream of words including punctuation marks from the recorded audio; second, it may be used indirectly to reprocess an initial transcription generated by a non-punctuation aware ASR system. In the indirect case, punctuation marks can be included in a transcription by decoding the corresponding audio with an HCLG[1] decoding graph constructed by composing a language model with a hyper-string WFSA constructed using the initial decoding.

4. SYSTEM COMBINATION

The systems described above are potentially complementary, hence we also evaluated two methods of system combination: voting and log-linear interpolation.

4.1. Voting

The most straightforward method of system combination is voting. We used per token voting in which a hyper-string WFSA is created from all hypotheses and weights are assigned to arcs according to the agreement between the systems. The resulting sequence of punctuation marks corresponds to the path with the highest per token agreement. We considered several voting schemes: uniform weight, precision per system weight and precision per system per token weight. We used the last one, because it yielded the best results (based on F-Measure).

4.2. Log-linear interpolation

Since the language modeling and word labeling approaches can be represented as WFSTs, and an n-best list from the NMT approach can be also represented as a WFST, a log-linear interpolation of them corresponds to a composition of the corresponding transducers. Even though an n-best list from NMT significantly prunes the search space, it is not a limiting factor for log-linear interpolation, because the oracle per segment F-Measure of the NMT n-best list is on the same level as the oracle per segment F-Measure of the combination of all single systems. Therefore, we can use this composition for ASR in the same way as above (Section 3.4).

[1] HCLG denotes the composition of the following WFSTs: acoustic HMM (H), context (C), lexicon (L), and language model (G).

	LM	train	dev
full stop [.]	62 694 524	669 807	7 204
comma [,]	39 095 545	418 883	5 222
question mark [?]	14 378 702	111 622	1 549
exclamation mark [!]	10 116 835	74 135	1 102
three dots [...]	4 622 695	8 288	553
All Tokens	775 338 349	8 767 781	92 622

Table 1. Number of occurrences of punctuation marks in the LM training data, and in the transcripts of the acoustic training and dev sets..

5. EXPERIMENTS

We performed experiments using the MGB Challenge data [1] which contains 1 600 hours of recorded television programmes for acoustic model training. In addition to transcripts for lightly-supervised acoustic model training, the MGB training data also includes segment time marks and word matching error rate (MER) measurements [1]. Because the transcripts, which were obtained from broadcast subtitles, contain considerable insertion, deletion, and substitution errors, the MER indicates how well the transcript corresponds to the audio (given a trained acoustic model). The MGB Challenge data also contains 650 millions words of BBC subtitles for language model training, to which we refer as LM data.

Since the provided preprocessed subtitles do not contain punctuation marks we used the raw subtitles, which were normalized using our internal text normalization pipeline and aligned to the preprocessed subtitles in order to obtain the correct time boundaries and MER for each segment. In these experiments we predicted five punctuation mark types: *full stop* [.], *comma* [,], *question mark* [?], *exclamation mark* [!] and *three dots* [...]. Table 1 summarizes the number of occurrences of punctuation marks in the different datasets. We repeated the same process with the MGB development dataset (referred to as dev). As this dataset contains human verbatim transcriptions, we aligned the normalized raw subtitles with the verbatime transcriptions. Since the raw subtitles do not always match the verbatim transcription, we used only a subset of the dev set where they matched perfectly for evaluation, referred to as sdev.

We performed experiments on both reference text and ASR output with a word error rate (WER) of 31.6% on the dev set of the MGB challenge, and 29.0% on our subset. We

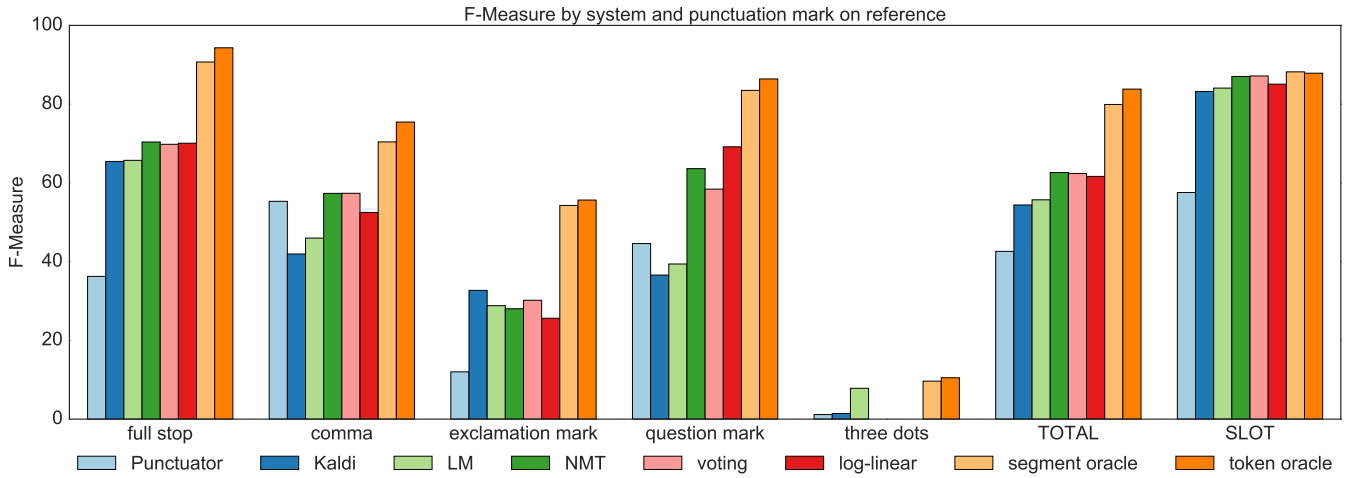

Fig. 2. F-Measure of punctuation marks on reference.

	PPL	P	R	F1
3-gram	101.97	55.55	55.86	55.71
4-gram	103.36	56.57	55.94	56.25
5-gram	101.97	56.59	55.92	56.25

Table 2. Impact of the n-gram length to perplexity (PPL), precision (P), recall(R) and f-measure (F1).

used the oracle segmentation in all the following experiments. Below we describe the training procedure of the punctuation systems.

Statistical language model: We trained an n-gram language model with Kneser-Ney interpolation [26] on the provided LM data and interpolated it with an n-gram language model trained on the transcripts of the acoustic training data. We observed that using a context beyond trigram did not significantly help with the transcript punctuation accuracy [7] (see Table 2 for comparison), therefore we used a trigram language model in all following experiments. We refer to this system as LM in the results.

Word labeling: We used Punctuator 2 [17] (`https://github.com/ottokart/punctuator2`) as a labeling approach. The system uses gated recurrent unit networks [27] and an attention layer [24]. We trained the system on the LM data using the suggested setting of hidden layer size 256 and learning rate 0.02. We monitored the system's performance after each epoch on dev data, keeping the best performing system for later use. We refer to this system as Punctuator in the results.

Neural machine translation: We used the NMT example from blocks.examples (`https://github.com/mila-udem/blocks-examples`) for our NMT approach. The hidden layer size was 512 and we trained using AdaDelta [28] with dropout [29]. We monitored F-Measure on the dev set every 5 000 iterations, keeping the best performing model for later use. We used a beam search with a beam size of 6 for decoding. Although the length of the output is known in advance, we were not able to improve the performance of the system by constraining it to output a sequence of punctuation marks of the given length. Therefore, we allowed the system to output a sequence of punctuation marks of arbitrary length, padding it with spaces or trimming it when necessary. We refer to this system as NMT in the results.

Automatic speech recognition: We trained an ASR system using the Kaldi speech recognition toolkit [30] using the training data from MGB challenge with $MER < 10\%$ (216 hours) to ensure that the inserted punctuation marks actually appeared in the audio. We assigned a different pronunciation to each punctuation mark and ensured that the phonemes corresponding to different punctuation marks were not tied to the same state.

We trained a GMM model using fbank+pitch features using the baseline recipe from the MGB Challenge (`http://mgb-challenge.org`). We then trained a DNN model using 6 hidden layers each with 2048 sigmoid units, which were initialized using RBM pre-training [31] and trained using the frame-wise cross-entropy objective function. Additionally, trained an *Adaptive softmax* system by reusing the hidden layers from a previously trained unpunctuated DNN on the MGB Challenge data with $MER \leq 30\%$ without punctuation (538 hours), and then retrained only a punctuation-aware softmax layer using cross-entropy, followed by sequence training using the sMBR objective function [32]. Results on the dev set, our subset when evaluated without punctuation marks, and on our subset when evaluated with punctuation marks are reported in Table 3. Since Adapted softmax + sMBR achieved the best results, we used this system in all further experiments. We refer to this system as Kaldi in the results.

We also explored the impact of using punctuation marks in acoustic modeling. We trained a GMM AM, DNN AM and Adapted softmax AM ($MER \leq 10$) with and without punc-

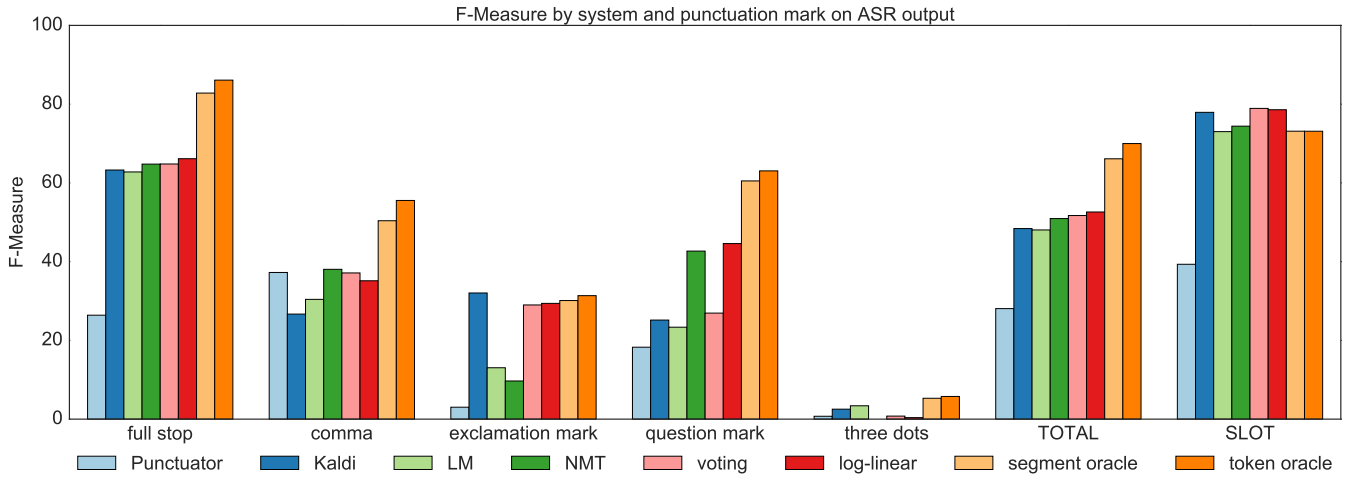

Fig. 3. F-Measure of punctuation marks on ASR output.

	dev	sdev-	sdev+
GMM baseline	48.59	46.33	51.93
DNN baseline	36.74	33.32	41.36
Adapted softmax	34.72	31.44	39.75
Adapted softmax + sMBR	33.43	30.44	39.14

Table 3. WER without punctuation on the official dev set, our dev subset and WER with punctuation on our dev subset.

trained with punctuation?	**yes**	**no**
GMM baseline	48.59	48.93
DNN baseline	36.74	36.39
Adapted softmax	34.72	34.62

Table 4. Comparison of systems when trained with or without punctuation marks.

tuation marks. From the results in Table 4. We may observe that a GMM AM trained with punctuation marks achieves a reduced WER of 0.3% absolute, but both DNN systems trained with punctuation marks had higher WERs (0.1–0.3% absolute).

6. RESULTS

In this section we report results using F-Measure on the reference transcript (Figure 2) and on the ASR output (Figure 3). The results are summarized in Table 5. Furthermore, we use paired bootstrap resampling for significance testing [33].

Single systems: Overall, NMT is significantly the best system with F-Measure of 63% followed by LM with F-Measure of 56%, Kaldi with F-Measure of 54% and Punctuator with F-Measure of 43% on reference transcriptions. On ASR output, NMT is also performs best with an F-Measure of 51% followed by Kaldi and LM with F-Measures of 48% LM, and Punctuator with F-Measure of 28%. The biggest problem of Punctuator system is very low recall – over three-quarters of the errors are deletions. The biggest problem of the remaining systems is disambiguation between punctuation marks – each of three systems has a substitution error of 45–50%. Except for the [. . .], which was very difficult to predict for all systems, we observe that most insertion errors were caused by full stops, deletion errors by commas, and substitution errors by confusing other punctuation marks with full stop. When we consider only the binary classification of whether a punctuation mark is present (called SLOT in Figure 2 and Figure 3), NMT achieves F-Measure of 87%, LM achieves F-Measure of 84%, Kaldi achieves F-Measure of 83% and Punctuator achieves F-Measure 58%.

Acoustic information improves the prediction of [!]. We compared Kaldi with the systems using only lexical features: using paired bootstrap resampling we show that Kaldi is significantly better in the prediction of exclamation marks than others system with $p = 0.982$ for NMT, $p = 0.963$ for LM and $p > 0.999$ for Punctuator. All systems can accurately detect a slot for [!], but systems using statistical language models are more likely to confuse [!] with [.].

To show that using longer context improves the prediction of question marks we compare the systems using LSTMs – NMT and Punctuator – with systems using statistical language models – LM and Kaldi. Using paired bootstrap resampling we show that both systems using LSTMs are significantly better in the prediction of question marks with $p > 0.998$. All systems are good in finding a slot for question mark, but systems using statistical language models are more likely to confuse question mark with full stop. However, using longer context harms the performance of the systems more than using finite context when there is a mismatch between reference and ASR output because there is a higher chance of error in a longer context. The relative degradation of systems on ASR output is 34% for Punctuator, 19% for NMT, 14% for LM and 11% for Kaldi.

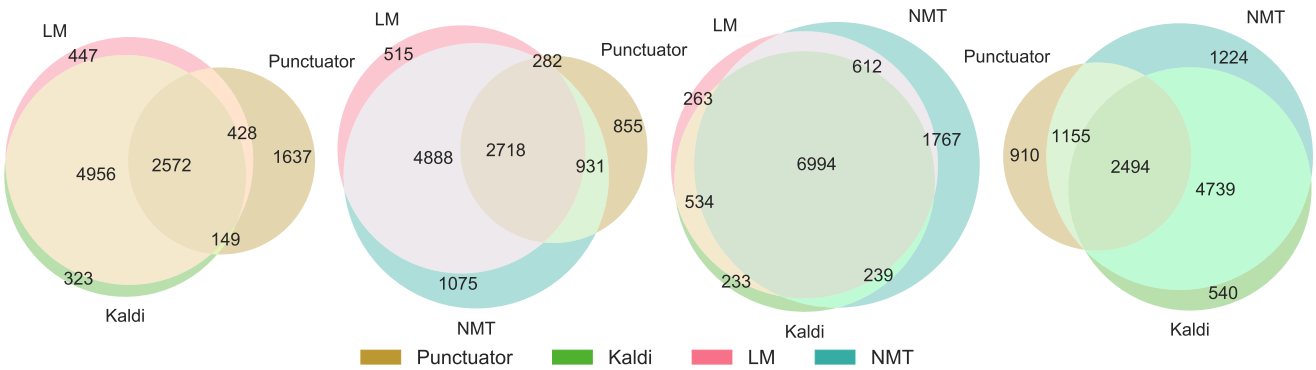

Fig. 4. Venn diagrams showing mutual agreement of systems on correctly predicted punctuation marks. Each diagram shows the relations between 3 out of 4 systems.

	Punctuator		Kaldi		LM		NMT		voting		log-linear	
	REF	ASR	REF	ASR	REF	ASR	REF	ASR	REF	ASR	REF	ASR
full stop	36.26	26.40	65.45	63.27	65.73	62.78	70.38	64.77	69.80	64.80	70.08	66.14
comma	55.32	37.25	41.94	26.68	45.98	30.42	57.34	38.05	57.36	37.13	52.50	35.14
exclamation mark	12.01	3.03	32.70	32.04	28.82	13.05	28.03	9.69	30.19	29.00	25.62	29.40
question mark	44.59	18.27	36.59	25.16	39.40	23.35	63.62	42.68	58.42	26.93	69.16	44.59
three dots	1.16	0.73	1.44	2.53	7.83	3.39	-	-	-	0.77	-	0.38
TOTAL	42.61	28.07	54.39	48.40	55.70	48.05	62.63	50.94	62.40	51.71	61.65	52.61
SLOT	57.56	39.33	83.21	77.92	84.08	73.02	87.03	74.41	87.16	78.92	85.08	78.58

Table 5. Results of the systems on reference (REF) and ASR output (ASR).

System Combination: To explore the limits of system combination, we computed the segment oracle and token oracle F-Measure of the combined systems. Note, that the computation of oracle is ambiguous depending on whether substitution or deletion errors are preferred. Since deletion errors have less effect on the readability of the transcript, we prefer them in this paper. Even though oracle experiments showed that the systems are complementary, voting and log-linear interpolation of systems using reference transcripts did not outperform NMT. This might be explained by the low mutual agreement of the systems on correctly predicted punctuation marks (see Figure 4), which causes the correct system to be outvoted by other systems. This is due to the simplicity of the used system combination methods and we want to evaluate more complex system combination methods in the future (e.g., [34]). On the other hand, log-linear interpolation on the ASR output is significantly better than NMT system with ($p = 0.999$), which we hypothesize is due to the ability of the Kaldi system to recover from errors in the ASR output.

7. CONCLUSIONS

In this paper we evaluated four systems for punctuated transcription, three of which were based on lexical features, the fourth of which used acoustic features by integrating punctuation into the speech recognition acoustic models. We also explored the combination of these component systems using voting and log-linear interpolation. In contrast to previous works, which had usually focused on full stop, comma and question mark, we also predicted exclamation marks and three dots in order to capture expressivity of broadcast speech. Among the single systems, an NMT-based system achieved the significantly best F-Measure of 63% on reference with 19% degradation on ASR output. We observed that using acoustic information significantly improves the prediction of exclamation marks and longer context significantly improves the prediction of question marks. However, we observed that systems using longer context harms the prediction more than finite context on ASR output. Finally, we observed that even though the systems are complementary, straightforward combination using voting or log-linear interpolation does not yield any improvement, because there is low mutual agreement between systems on correctly predicted punctuation marks.

In the future work we plan to extend the NMT based system for transcript punctuation to incorporates acoustic input features, and to explore the robustness of these approaches across domains and languages.

8. ACKNOWLEDGMENTS

This work was supported by the H2020 project SUMMA, under grant agreement 688139.

9. REFERENCES

[1] Peter Bell, MJF Gales, Thomas Hain, Jonathan Kilgour, Pierre Lanchantin, Xunying Liu, Andrew McParland, Steve Renals, Oscar Saz, Mirjam Wester, and Philip C Woodland, "The MGB challenge: Evaluating multi-genre broadcast media recognition," in *ASRU*, 2015.

[2] A. Stolcke and E. Shriberg, "Automatic linguistic segmentation of conversational speech," in *ICSLP*, 1996, pp. 1005–1008.

[3] Andreas Stolcke, Elizabeth Shriberg, Rebecca A Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gökhan Tür, and Yu Lu, "Automatic detection of sentence boundaries and disfluencies based on recognized words.," in *ICSLP*, 1998.

[4] Doug Beeferman, Adam Berger, and John Lafferty, "Cyberpunc: A lightweight punctuation annotation system for speech," in *ICASSP*, 1998.

[5] C Julian Chen, "Speech recognition with automatic punctuation.," in *Eurospeech*, 1999.

[6] Ji-Hwan Kim and Philip C Woodland, "The use of prosody in a combined system for punctuation generation and speech recognition.," in *Interspeech*, 2001.

[7] Agustin Gravano, Martin Jansche, and Michiel Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *ICASSP*, 2009.

[8] Yoshihiko Gotoh and Steve Renals, "Sentence boundary detection in broadcast speech transcripts," 2000.

[9] Heidi Christensen, Yoshihiko Gotoh, and Steve Renals, "Punctuation annotation using statistical prosody models," in *ISCA Workshop on Prosody in Speech Recognition and Understanding*, 2001.

[10] Jing Huang and Geoffrey Zweig, "Maximum entropy model for punctuation annotation from speech," in *INTERSPEECH*, 2002.

[11] Jáchym Kolář, Jan Švec, and Josef Psutka, "Automatic punctuation annotation in Czech broadcast news speech," *SPECOM*, 2004.

[12] Jáchym Kolář and Lori Lamel, "Development and evaluation of automatic punctuation for French and English speech-to-text.," in *Interspeech*, 2012.

[13] Wei Lu and Hwee Tou Ng, "Better punctuation prediction with dynamic conditional random fields," in *EMNLP*, 2010.

[14] Nicola Ueffing, Maximilian Bisani, and Paul Vozila, "Improved models for automatic punctuation prediction for spoken and written text.," in *Interspeech*, 2013.

[15] Eunah Cho, Kevin Kilgour, Jan Niehues, and Alex Waibel, "Combination of NN and CRF models for joint detection of punctuation and disfluencies," in *Interspeech*, 2015.

[16] Ottokar Tilk and Tanel Alumäe, "LSTM for punctuation restoration in speech transcripts," in *Interspeech*, 2015.

[17] Ottokar Tilk and Tanel Alumäe, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *Interspeech*, 2016.

[18] Stephan Peitz, Markus Freitag, Arne Mauser, and Hermann Ney, "Modeling punctuation prediction as machine translation.," in *IWSLT*, 2011.

[19] Eunah Cho, Jan Niehues, and Alex Waibel, "Segmentation and punctuation prediction in speech language translation using a monolingual translation system.," in *IWSLT*, 2012.

[20] Joris Driesen, Alexandra Birch, Simon Grimsey, Saeid Safarfashandi, Juliet Gauthier, Matt Simpson, and Steve Renals, "Automated production of true-cased punctuated subtitles for weather and news broadcasts," in *Interspeech*, 2014.

[21] Madina Hasan, Rama Doddipatla, and Thomas Hain, "Noise-matched training of CRF based sentence end detection models," in *Interspeech*, 2015.

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

[23] Ilya Sutskever, Oriol Vinyals, and Quoc Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[25] Rico Sennrich and Barry Haddow, "Linguistic input features improve neural machine translation," *arXiv preprint arXiv:1606.02892*, 2016.

[26] Reinhard Kneser and Hermann Ney, "Improved backing-off for m-gram language modeling," in *ICASSP*, 1995.

[27] Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio, "Gated feedback recurrent neural networks," *arXiv preprint arXiv:1502.02367*, 2015.

[28] Matthew D Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[29] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý, "The Kaldi speech recognition toolkit," in *ASRU*, 2011.

[31] Geoffrey Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, no. 1, pp. 926, 2010.

[32] Karel Vesely, Arnab Ghoshal, Lukas Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Interspeech*, 2013.

[33] Philipp Koehn, "Statistical significance tests for machine translation evaluation.," in *EMNLP*, 2004.

[34] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.