Punctuation Prediction for Unsegmented Transcript Based on Word Vector

Xiaoyin Che, Cheng Wang, Haojin Yang, Christoph Meinel
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
xiaoyin.che@hpi.de, cheng.wang@hpi.de, haojin.yang@hpi.de, christoph.meinel@hpi.de

Abstract

In this paper we propose an approach to predict punctuation marks for unsegmented speech transcript. The approach is purely lexical, with pre-trained Word Vectors as the only input. A training model of Deep Neural Network (DNN) or Convolutional Neural Network (CNN) is applied to classify whether a punctuation mark should be inserted after the third word of a 5-words sequence and which kind of punctuation mark the inserted one should be. TED talks within IWSLT dataset are used in both training and evaluation phases. The proposed approach shows its effectiveness by achieving better result than the state-of-the-art lexical solution which works with same type of data, especially when predicting puncuation position only.

Keywords: Punctuation Prediction, Word Vector, Deep Learning, Convolutional Neural Network

1. Introduction

Generally the text transcript generated by automatic speech recognition (ASR) systems is unpunctuated and unsegmented. However, the readability of the transcript can be largely improved by the existence of punctuation marks, and the segmentation of the transcript based on punctuation positions will also increase the efficiency of many downstream natural languages processing (NLP) tasks, such as semantic parsing, question answering or machine translation(Matusov et al., 2007; Wang et al., 2010). It may also act as effective pre-processing to facilitate some other tasks like subtitling(Tiedemann, 2007).

Therefore many efforts, some of which may focus on similiar topic "sentence boundary detection", have been made to restore or predict the punctuation marks. Most of the recent works can be divided into three categories: prosodic or acoustic feature based (Levy et al., 2012; Xie et al., 2012; Sinclair et al., 2014), lexical feature based (Gravano et al., 2009; Lu and Ng, 2010; Ueffing et al., 2013) and the hybrid of previous two (Wang et al., 2012; Xu et al., 2014; Hasan et al., 2014). In this paper, we focus on the lexical aspect only.

A pure lexical model is understandably less powerful than a hybrid model, or even a pure acoustic model. But it is able to cope with the situation either with or without the audio files. Another fact is that the training data of a hybrid model, which take both lexical and acoustic features as input, must be the standardized ASR transcripts, but a pure lexical model can take any kind of textual material into consideration, which is almost endless and extremely easy to get. However, the posterior probabilities of the independent lexical model can be freely used in late fusion with acoustic features(Tilk and Alumäe, 2015; Khomitsevich et al., 2015). Considering all these facts, we believe a good lexical model is widely applicable.

In previous research efforts, frequently used lexical features include the language model (LM) score, token, part-of-speech (POS) tag, chunk tag and so on. However, we would like attempt another possibility by using the Word Vector. Word vectors are learned through neural language models (Bengio et al., 2003). Each word will be represented by a comparatively low-dimentional real-valued vector and the lexical similarity between words can be evaluated by the Euclidean distance between their correspondent vectors (Mikolov et al., 2013; Pennington et al., 2014). Thus, word vectors can be used as lexical features in various NLP applications and have achieved many positive results (Collobert et al., 2011; Maas et al., 2011).

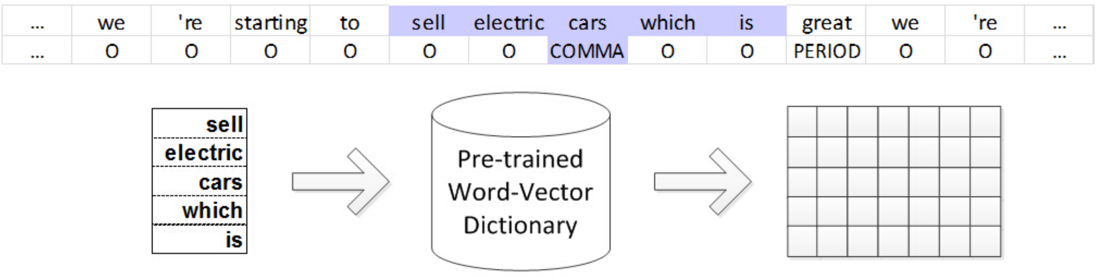
When training, some state-of-the-art punctuation prediction approaches chose Conditional Random Fields (CRFs) as the classifier, but only taking traditional types of lexical features as input. Meanwhile, some recent experiments have shown the effectiveness of the combination of word vector and convolutional neural network (CNN) in the task of sentence classification or sentiment analysis (Kim, 2014; Kalchbrenner et al., 2014). Inspired by these experiments, we would like to attempt this combination in the task of punctuation prediction in this paper.

2. Model Description

2.1. Input Preparation

In our approach, all the data are collected from text files, which will be first transformed into a long word sequence. We treat the classification problem as whether a word in the sequence is followed by a punctuation mark. We define 4 classes in total: O (*means no punctuation mark followed*), COMMA, PERIOD and QUESTION. Exclamation marks or semicolons are classified as PERIOD, while colons or dashes are classified as COMMA. There are no brackets in our data and all the other punctuation marks, such as quotation marks, are just ignored.

For the word w_i in the sequence, we apply an continuous m-words subsequence $<w_{i-(m-1)/2}$,..., w_{i-1}, w_i, w_{i+1},..., $w_{i+(m-1)/2}>$ to represent the context. The n-dimensional word vectors of these words can be extracted from a pre-trained "dictionary", and a default vector will be used as the replacement for all words which do not exist in the dictionary. Then we obtain an $m \times n$ feature matrix as the training model input, just as shown in Fig. 1. During the training process, the value of word vectors will be kept static.



...	we	're	starting	to	sell	electric	cars	which	is	great	we	're	...
...	O	O	O	O	O	O	COMMA	O	O	PERIOD	O	O	...

Figure 1: Input Preparation

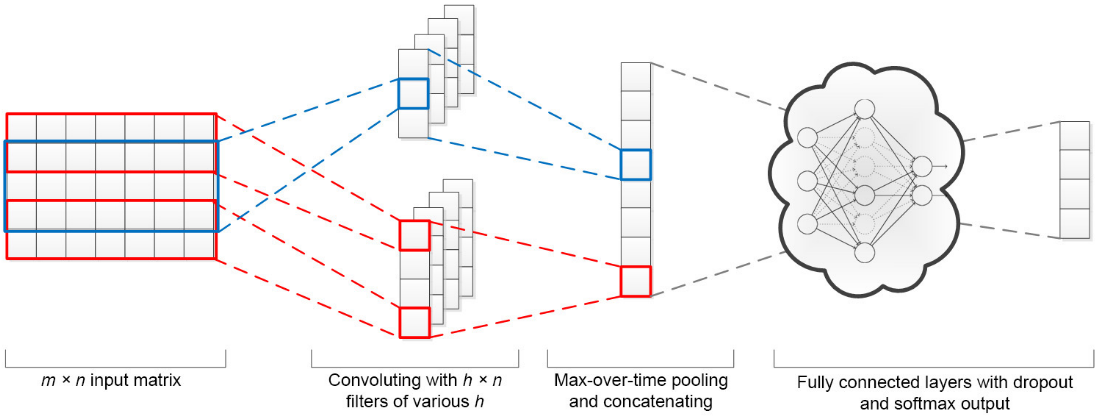

m × n input matrix
Convoluting with h × n filters of various h
Max-over-time pooling and concatenating
Fully connected layers with dropout and softmax output

Figure 2: Model Architecture of CNN-1

2.2. Training Models

We attempt three different models for the training process. The first model is a typical deep neural network with 3 hidden fully connected layers. We address it as DNN in our experiments. The $m \times n$ input matrix will be reshaped into a long vector, which equals to concatenate the m word vectors together, and this long vector acts as the input layer of the neural network. Then each training sample can be represented as $\{X_i, Y_i\}$, $i = 1, 2, \ldots, N$. $X_i \in \mathbb{R}^{m \times n}$, $X_i = \{x_1, x_2, \ldots, x_{m \times n}\}$, while $Y_i \in \mathbb{R}^K$, $y_i = \{y_1, y_2, \ldots, y_K\}$. The goal is to find the optimal weights W and biases b within cost function C:

$$\arg\min_{W,b} C = \frac{1}{2N} \sum_{i=1}^{N} \|\hat{Y}_i - Y_i\|^2 + \frac{\lambda}{2} \sum_{l=1}^{L-1} \|W_l\|_F^2 \qquad (1)$$

where $\hat{Y}_i$ is the predicted output, while the second term in (1) is the weight decay. It is implemented to prevent overfitting, with the weight decay parameter λ which is set to 0.0005 in our case. Sigmoid function as (2) is applied to all hidden layers and softmax function as (3) to the output layer.

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2)$$

$$S_k(y) = \frac{e^{y_k}}{\sum_{j=1}^{K} e^{y_j}}, k = 1, 2, \ldots, K \qquad (3)$$

In order to prevent co-adaptation between neurons in the hidden layers, we implement "dropout" on fully connected layers, which randomly "hide" some neurons along with their connections to reduce overfitting (Srivastava et al., 2014). Dropout is also applied in the fully connected layers in other two models.

The architecture of second model (*Fig. 2*) is similar as described by Kim (2014). Taking $m \times n$ matrix as input, a convolutional filter with the kernel size of $h \times n$ will be applied, in which $1 \leq h < m$. Because the value of n is fixed, the filter acts as a sliding window which moves only vertically and works with every h continuous words within the sample. As a result, a feature map with $m - h + 1$ elements will be generated. Then a max-over-time pooling operation will extract the maximum value as the final feature of this filter. By implementing a group of such filters, with same or different value of h, multiple features can be achieved and then concatenated together, which will be further fed into two fully connected layers with a softmax output. In this model, the integrity of word representation is guaranteed during the process of convolution. We will address this model as CNN-1 in later sections.

The third model (*Fig. 3*) is also a variant of CNN, but we will treat the $m \times n$ feature matrix no longer as m complete word vectors. Instead, each element in the matrix will be treated independently, just like a pixel in an image with a resolution of $m \times n$. There are 3 typical convolution layers and 1 pooling layer, followed also by 2 fully connected layers and softmax output. This model will be addressed as CNN-2.

2.3. Implementation

We choose the 50-dimensional pre-trained word vectors from GloVe for our experiments[1]. Compared with

[1] http://nlp.stanford.edu/projects/glove/

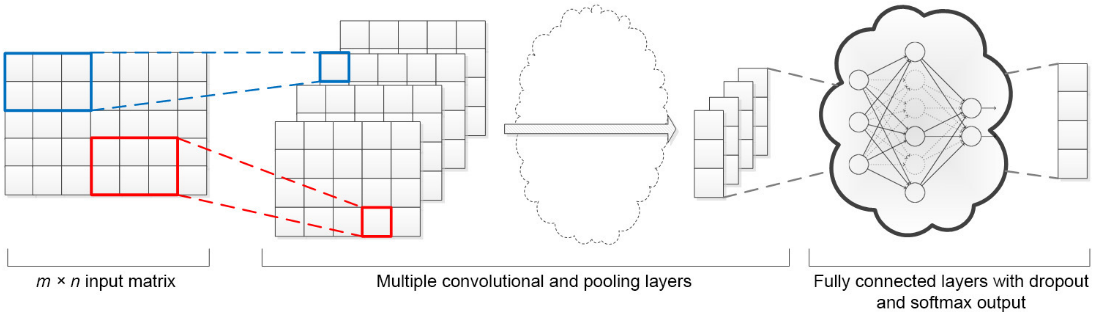

m × *n* input matrix
Multiple convolutional and pooling layers
Fully connected layers with dropout and softmax output

Figure 3: Model Architecture of CNN-2

	Dataset	Words	COMMA	PERIOD	QUESTION
Ref	Ueffing et al. (2013)	17207	1096	925	84
	Ours	12626	830	808	46
ASR	Ueffing et al. (2013)	17344	1084	904	82
	Ours	12822	798	810	35

Table 1: Test Sets Details

other publicly available word vector packs, such as 300-dimensional word2vec[2], GloVe-50d is a much smaller set, not only dimensional, but also on the total words involved. But for our purpose, GloVe-50d is already sufficient. And in all our experiments, we will use 5-words subsequence as a sample. All above models are trained on CAFFE framework (Jia et al., 2014), taking HDF5 as input format and with $m = 5, n = 50$.

In DNN model, we set the total number of neurons for each layer to 2048, 4096 and 2048. In CNN-1 model, we apply 128 "convolutional + pooling" filter pairs for each window size from 1-word to 4-words, and the two following fully connected layers have 4096 and 2048 neurons respectively. In CNN-2 model, the numbers of filters of three sequential convolutional layers, with pooling layer after the first one, are set to 64, 128 and 128, followed by the same fully connected layers of CNN-1. When running, the resources occupied by 3 models can be ordered by: DNN < CNN-1 < CNN-2.

3. Evaluation

3.1. Datasets

We test our models on the IWSLT datasets, which were originally used to evaluate ASR or SLT output. IWSLT datasets consist of TED talks which are openly available online[3]. There are both punctuated reference transcripts and unpunctuated but segmented ASR transcripts provided. Ueffing et al. (2013) reported the performance of their approach on IWSLT2011 test set, which we would like to compare with. However, we found that the IWSLT2011 dataset we achieved[4] is smaller than claimed by Ueffing et al. (2013). The detail about the test set can be found in Table 1.

We divide the training data of machine translation track in IWSLT2012, which also consists of TED talks, into two parts as our training and development sets. Based on TED talk ID, we make sure that there is no overlapping between datasets. The training and development sets contain 2.1M and 296k samples respectively.

Several reports have indicated that the performance of punctuation prediction is largely influenced by the average number of punctuation marks per utterance in the dataset (Wang et al., 2012). It is quite understandable that there will always be a period or question mark at the end of the utterance. In our evaluation, we remove all the segmentation from both the reference and ASR transcripts of IWSLT data files and there is only one single utterance existing in each dataset.

3.2. Metrics

If we define a word sequence between two punctuation marks as a minimal sentence unit, the average length of such minimal sentence units of our training dataset is 7.8, which means there are over 85% of the words are not followed by any punctuation mark, in other words, they are the "O" samples. The majority of these "O" samples can be successfully classified by our model and make the general accuracy of the classification beyond 90%. But in purpose of predicting punctuation marks, we care much more about the performance on the samples belonging to the other three "punctuated" classes. Therefore the correctly classified "O" samples will be ignored and we evaluate the performance by:

$$Precision = \frac{\#\, Correctly\ predicted\ punctuation\ marks}{\#\, All\ predicted\ punctuation\ marks} \tag{4}$$

$$Recall = \frac{\#\, Correctly\ predicted\ punctuation\ marks}{\#\, All\ expected\ punctuation\ marks} \tag{5}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

[2] https://code.google.com/p/word2vec/

[3] https://www.ted.com/talks

[4] http://hltc.cs.ust.hk/iwslt/index.php/evaluation-campaign/ted-task.html

	Model	4-Classes			3-Classes			2-Classes		
		Pre.	Rec.	F_1	Pre.	Rec.	F_1	Pre.	Rec.	F_1
Ref	CRF Best	49.8	**58**	53.5	–	–	–	–	–	75.8
	DNN	**60.3**	48.6	53.8	**62.1**	50.1	55.5	**86.0**	69.2	76.7
	CNN-1	55.2	46.4	50.4	57.0	47.9	52.1	83.0	69.8	75.8
	CNN-2	57.8	49.9	53.5	59.6	51.5	55.3	83.6	72.3	77.5
	DNN-A	54.8	53.6	**54.2**	56.4	55.2	55.8	79.0	77.3	78.2
	CNN-2A	53.4	55.0	**54.2**	55.1	**56.7**	**55.9**	77.6	**79.9**	**78.8**
ASR	CRF Best	47.8	**54.8**	**51**	–	–	–	–	–	64
	DNN	**54.4**	45.6	49.6	**56.0**	46.9	51.0	**77.5**	64.9	70.7
	CNN-1	49.9	45.2	47.5	51.4	46.6	48.9	73.9	66.9	70.2
	CNN-2	51.0	46.8	48.8	52.3	48.1	50.1	74.2	68.1	71.0
	DNN-A	49.2	51.6	50.4	50.5	53.0	**51.6**	70.6	74.1	**72.3**
	CNN-2A	46.4	51.9	49.1	47.6	**53.2**	50.2	68.2	**76.3**	72.1

Table 2: Experimental Results on General Accuracy

	Model	COMMA			PERIOD			QUESTION		
		Pre.	Rec.	F_1	Pre.	Rec.	F_1	Pre.	Rec.	F_1
Ref	DNN	**58.2**	35.7	44.2	**61.6**	64.8	63.2	0	0	–
	CNN-1	48.3	36.3	41.4	60.6	59.6	60.1	0	0	–
	CNN-2	53.4	37.6	44.1	60.7	65.4	63.0	0	0	–
	DNN-A	48.6	42.4	45.3	59.7	68.3	**63.7**	0	0	–
	CNN-2A	48.1	**44.5**	**46.2**	57.6	**69.0**	62.8	0	0	–
ASR	DNN	**47.2**	32.0	38.1	**59.0**	60.9	60	0	0	–
	CNN-1	40.2	33.5	36.5	57.8	58.8	58.3	0	0	–
	CNN-2	42.1	33.6	37.4	57.5	61.9	59.6	0	0	–
	DNN-A	41.0	**40.9**	**40.9**	56.2	64.5	**60.1**	0	0	–
	CNN-2A	37.3	40.5	38.8	54.6	**65.5**	59.6	0	0	–

Table 3: Experimental Results per Class

The original test is addressed as 4-Classes. Then we combine the QUESTION and PERIOD together as "full-stop" class to form a 3-Classes test. Furthermore, the "full-stop" and COMMA classes can be combined again as "punctuated" in a 2-Classes test. By this configuration, we predict punctuation position only, regardless of the punctuation type. We implement 3-Classes and 2-Classes tests because in some applications, such as semi-automated subtitle generation, pre-segmenting the transcript by accurately detected punctuation position can save a lot of time for the human staff, but whether it is a comma or a period at the end of the segment is less important.

Besides the general accuracy evaluation introduced above, we are also interested in "per-class" performance. The statistics about comma, period and question mark will be collected separately. The general accuracy and per-class performance can be found in Table 2 and Table 3 respectively.

3.3. Results

As shown in Table 2, the general accuracy of the experiments on both reference transcripts and ASR output can be found. "CRF best" represent the best model performance reported by Ueffing et al. (2013). "DNN", "CNN-1" and "CNN-2" are corresponding to the 3 models proposed in this work. Furthermore, since we found out that all three proposed models achieved much higher precision than recall, we did one additional adjustment to our better performers: half the value of softmax output for class "O". By this effort (*addressed as DNN-A & CNN-2A*), more border cases would be classified as "punctuated", which balanced the precision and recall.

From the stats we can find out that in 2-Classes evaluation, all proposed models perform better than the benchmark, with CNN-2A and DNN-A as best performers on reference transcripts and ASR output respectively. CNN-2A is also the best performer for 4-Classes evaluation on reference. But in ASR 4-Classes evaluation, none of proposed models can outperform the benchmark. These facts show that our approach is good at predicting the punctuation positions, but needs to improve on distinguishing puctuation types.

Table 3 shows the per-class performances. Obviously, our proposed models failed to predict even one question mark in the test dataset. We believe there are two reason for this phenomenon: the first is that the QUESTION samples in the training set are not sufficient; the second, which we consider as the more crucial one, is that the sample we apply, a 5-word sequence, does not cover the sentence beginning, where the typical question pattern locates, such as "*what do you ...*" or "*how can I ...*". Therefore, the incompetence of proposed model in detecting question mark is understandable, also still disappointing.

Among comma and period, all proposed model works much better with period. We think it is normal because the grammatical ambiguity of a pause is generally higher than a full-stop, especially in less formal text like the transcript of TED talks. Another interesting finding is that our CNN-1 model, which keeps the integrity of word representation during the convolutional process, achieved worse performance than the other two. It means, at least in punctuation prediction task, the digits in the word vectors could be considered as independent features.

4. Conclusion

In this paper we attempted to use word vectors as the features in predicting punctuation marks for unsegmented transcript in a pure lexical approach. The pre-trained word vectors are fed into proposed models based on DNN or CNN for training. The evaluation shows that the proposed approach can achieve quite promising result. In the future we intend to first improve the input word vectors, by applying larger pre-trained set or training our own word vectors, and use longer word sequence as samples. Then we will also attempt to involve prosodic feature.

5. References

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Gravano, A., Jansche, M., and Bacchiani, M. (2009). Restoring punctuation and capitalization in transcribed speech. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4741–4744. IEEE.

Hasan, M., Doddipatla, R., and Hain, T. (2014). Multi-pass sentence-end detection of lecture speech. In *Proc. Interspeech*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Khomitsevich, O., Chistikov, P., Krivosheeva, T., Epimakhova, N., and Chernykh, I. (2015). Combining prosodic and lexical classifiers for two-pass punctuation detection in a russian asr system. In *Speech and Computer*, pages 161–169. Springer.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Levy, T., Silber-Varod, V., and Moyal, A. (2012). The effect of pitch, intensity and pause duration in punctuation detection. In *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*, pages 1–4. IEEE.

Lu, W. and Ng, H. T. (2010). Better punctuation prediction with dynamic conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 177–186. Association for Computational Linguistics.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Matusov, E., Hillard, D., Magimai-Doss, M., Hakkani-Tür, D. Z., Ostendorf, M., and Ney, H. (2007). Improving speech translation with automatic boundary prediction. In *INTERSPEECH*, volume 7, pages 2449–2452.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Sinclair, M., Bell, P., Birch, A., and McInnes, F. (2014). A semi-markov model for speech segmentation with an utterance-break prior. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tiedemann, J. (2007). Improved sentence alignment for movie subtitles. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, volume 7.

Tilk, O. and Alumäe, T. (2015). Lstm for punctuation restoration in speech transcripts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Ueffing, N., Bisani, M., and Vozila, P. (2013). Improved models for automatic punctuation prediction for spoken and written text. In *INTERSPEECH*, pages 3097–3101.

Wang, K., Ming, Z.-Y., Hu, X., and Chua, T.-S. (2010). Segmentation of multi-sentence questions: towards effective question retrieval in cqa services. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 387–394. ACM.

Wang, X., Ng, H. T., and Sim, K. C. (2012). Dynamic conditional random fields for joint sentence boundary and punctuation prediction. In *INTERSPEECH*.

Xie, L., Xu, C., and Wang, X. (2012). Prosody-based sentence boundary detection in chinese broadcast news. In *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium on*, pages 261–265. IEEE.

Xu, C., Xie, L., Huang, G., Xiao, X., Chng, E. S., and Li, H. (2014). A deep neural network approach for sentence boundary detection in broadcast news. In *Proc. Interspeech*.