Punctuation Prediction with Transition-based Parsing

Dongdong Zhang[1], Shuangzhi Wu[2], Nan Yang[3], Mu Li[1]

[1]Microsoft Research Asia, Beijing, China
[2]Harbin Institute of Technology, Harbin, China
[3]University of Science and Technology of China, Hefei, China
{dozhang,v-shuawu,v-nayang,muli}@microsoft.com

Abstract

Punctuations are not available in automatic speech recognition outputs, which could create barriers to many subsequent text processing tasks. This paper proposes a novel method to predict punctuation symbols for the stream of words in transcribed speech texts. Our method jointly performs parsing and punctuation prediction by integrating a rich set of syntactic features when processing words from left to right. It can exploit a global view to capture long-range dependencies for punctuation prediction with linear complexity. The experimental results on the test data sets of IWSLT and TDT4 show that our method can achieve high-level performance in punctuation prediction over the stream of words in transcribed speech text.

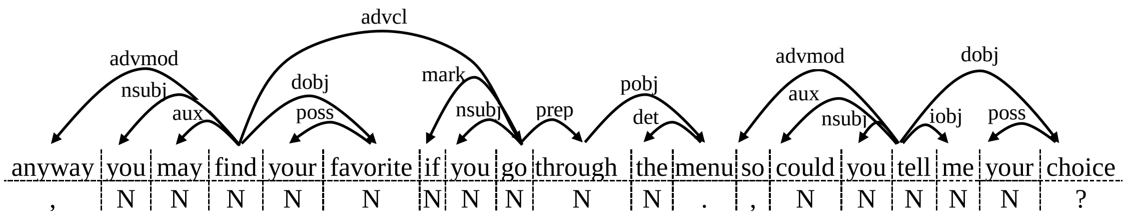
1 Introduction

Standard automatic speech recognizers output unstructured streams of words. They neither perform a proper segmentation of the output into sentences, nor predict punctuation symbols. The unavailable punctuations and sentence boundaries in transcribed speech texts create barriers to many subsequent processing tasks, such as summarization, information extraction, question answering and machine translation. Thus, the segmentation of long texts is necessary in many real applications. For example, in speech-to-speech translation, continuously transcribed speech texts need to be segmented before being fed into subsequent machine translation systems (Takezawa et al., 1998; Nakamura, 2009). This is because current machine translation (MT) systems perform the translation at the sentence level, where various models used in MT are trained over segmented sentences and many algorithms inside MT have an exponential complexity with regard to the length of inputs.

The punctuation prediction problem has attracted research interest in both the speech processing community and the natural language processing community. Most previous work primarily exploits local features in their statistical models such as lexicons, prosodic cues and hidden event language model (HELM) (Liu et al., 2005; Matusov et al., 2006; Huang and Zweig, 2002; Stolcke and Shriberg, 1996). The word-level models integrating local features have narrow views about the input and could not achieve satisfied performance due to the limited context information access (Favre et al., 2008). Naturally, global contexts are required to model the punctuation prediction, especially for long-range dependencies. For instance, in English question sentences, the ending question mark is long-range dependent on the initial phrases (Lu and Ng, 2010), such as "*could you*" in Figure 1. There has been some work trying to incorporate syntactic features to broaden the view of hypotheses in the punctuation prediction models (Roark et al., 2006; Favre et al., 2008). In their methods, the punctuation prediction is treated as a separated post-procedure of parsing, which may suffer from the problem of error propagation. In addition, these approaches are not able to incrementally process inputs and are not efficient for very long inputs, especially in the cases of long transcribed speech texts from presentations where the number of streaming words could be larger than hundreds or thousands.

In this paper, we propose jointly performing punctuation prediction and transition-based dependency parsing over transcribed speech text. When the transition-based parsing consumes the stream of words left to right with the shift-reduce decoding algorithm, punctuation symbols are predicted for each word based on the contexts of the parsing tree. Two models are proposed to cause the punctuation prediction to interact with the transition actions in parsing. One is to conduct transition actions of parsing followed by punctuation predictions in a cascaded way. The other is to associate the conventional transition actions of parsing with punctuation perditions, so that predicted punctuations are directly inferred from the anyway you may find your favorite if you go through the menu so could you tell me your choice

(a). The transcribed speech text without punctuations


advcl
advmod
nsubj
aux
dobj
poss
mark
nsubj
prep
pobj
det
advmod
aux
nsubj
dobj
iobj
poss
anyway you may find your favorite if you go through the menu so could you tell me your choice
, N N N N N N N N N N . , N N N N N ?

(b). Transition-based parsing trees and predicted punctuations over transcribed text

anyway, you may find your favorite if you go through the menu. so, could you tell me your choice?

(c). Two segmentations are formed when inserting the predicted punctuation symbols into the transcribed text

Figure 1. An example of punctuation prediction.

parsing tree. Our models have linear complexity and are capable of handling streams of words with any length. In addition, the computation of models use a rich set of syntactic features, which can improve the complicated punctuation predictions from a global view, especially for the long range dependencies.

Figure 1 shows an example of how parsing helps punctuation prediction over the transcribed speech text. As illustrated in Figure 1(b), two commas are predicted when their preceding words act as the adverbial modifiers (advmod) during parsing. The period after the word "*menu*" is predicted when the parsing of an adverbial clause modifier (advcl) is completed. The question mark at the end of the input is determined when a direct object modifier (dobj) is identified, together with the long range clue that the auxiliary word occurs before the nominal subject (nsubj). Eventually, two segmentations are formed according to the punctuation prediction results, shown in Figure 1(c).

The training data used for our models is adapted from Treebank data by excluding all punctuations but keeping the punctuation contexts, so that it can simulate the unavailable annotated transcribed speech texts. In decoding, beam search is used to get optimal punctuation prediction results. We conduct experiments on both IWSLT data and TDT4 test data sets. The experimental results show that our method can achieve higher performance than the CRF-based baseline method.

The paper is structured as follows: Section 2 conducts a survey of related work. The transition-based dependency parsing is introduced in Section 3. We explain our approach to predicting punctuations for transcribed speech texts in Section 4. Section 5 gives the results of our experiment. The conclusion and future work are given in Section 6.

2 Related Work

Sentence boundary detection and punctuation prediction have been extensively studied in the speech processing field and have attracted research interest in the natural language processing field as well. Most previous work exploits local features for the task. Kim and Woodland (2001), Huang and Zweig (2002), Christensen et al. (2001), and Liu et al. (2005) integrate both prosodic features (pitch, pause duration, etc.) and lexical features (words, n-grams, etc.) to predict punctuation symbols during speech recognition, where Huang and Zweig (2002) uses a maximum entropy model, Christensen et al. (2001) focus on finite state and multi-layer perceptron methods, and Liu et al. (2005) uses conditional random fields. However, in some scenarios the prosodic cues are not available due to inaccessible original raw speech waveforms. Matusov et al. (2006) integrate segmentation features into the log-linear model in the statistical machine translation (SMT) framework to improve the translation performance when translating transcribed speech texts. Lu and Ng (2010) uses dynamic conditional random fields to perform both sentence boundary and sentence type prediction. They achieved promising results on both English and Chinese transcribed speech texts. The above work only exploits local features, so they were limited to capturing long range dependencies for punctuation prediction.

It is natural to incorporate global knowledge, such as syntactic information, to improve punctuation prediction performance. Roark et al. (2006) use a rich set of non-local features including parser scores to re-rank full segmentations. Favre et al. (2008) integrate syntactic information from a PCFG parser into a log-linear and combine it with local features for sentence segmentation. The punctuation prediction in these works is performed as a post-procedure step of parsing, where a parse tree needs to be built in advance. As their parsing over the stream of words in transcribed speech text is exponentially complex, their approaches are only feasible for short input processing. Unlike these works, we incorporate punctuation prediction into the parsing which process left to right input without length limitations.

Numerous dependency parsing algorithms have been proposed in the natural language processing community, including transition-based and graph-based dependency parsing. Compared to graph-based parsing, transition-based parsing can offer linear time complexity and easily leverage non-local features in the models (Yamada and Matsumoto, 2003; Nivre et al., 2006b; Zhang and Clark, 2008; Huang and Sagae, 2010). Starting with the work from (Zhang and Nivre, 2011), in this paper we extend transition-based dependency parsing from the sentence-level to the stream of words and integrate the parsing with punctuation prediction.

Joint POS tagging and transition-based dependency parsing are studied in (Hatori et al., 2011; Bohnet and Nivre, 2012). The improvements are reported with the joint model compared to the pipeline model for Chinese and other richly inflected languages, which shows that it also makes sense to jointly perform punctuation prediction and parsing, although these two tasks of POS tagging and punctuation prediction are different in two ways: 1). The former usually works on a well-formed single sentence while the latter needs to process multiple sentences that are very lengthy. 2). POS tags are must-have features to parsing while punctuations are not. The parsing quality in the former is more sensitive to the performance of the entire task than in the latter.

3 Transition-based dependency parsing

In a typical transition-based dependency parsing process, the shift-reduce decoding algorithm is applied and a queue and stack are maintained (Zhang and Nivre, 2011). The queue stores the stream of transcribed speech words, the front of which is indexed as the current word. The stack stores the unfinished words which may be linked with the current word or a future word in the queue. When words in the queue are consumed from left to right, a set of transition actions is applied to build a parse tree. There are four kinds of transition actions conducted in the parsing process (Zhang and Nivre, 2011), as described in Table 1.

Action	Description
Shift	Fetches the current word from the queue and pushes it to the stack
Reduce	Pops the stack
LeftArc	Adds a dependency link from the current word to the stack top, and pops the stack
RightArc	Adds a dependency link from the stack top to the current word, takes away the current word from the queue and pushes it to the stack

Table 1. Action types in transition-based parsing

The choice of each transition action during the parsing is scored by a linear model that can be trained over a rich set of non-local features extracted from the contexts of the stack, the queue and the set of dependency labels. As described in (Zhang and Nivre, 2011), the feature templates could be defined over the lexicons, POS-tags and the combinations with syntactic information.

In parsing, beam search is performed to search the optimal sequence of transition actions, from which a parse tree is formed (Zhang and Clark, 2008). As each word must be pushed to the stack once and popped off once, the number of actions needed to parse a sentence is always $2n$, where n is the length of the sentence. Thus, transition-based parsing has a linear complexity with the length of input and naturally it can be extended to process the stream of words.

4 Our method

4.1 Model

In the task of punctuation prediction, we are given a stream of words from an automatic transcription of speech text, denoted by $w_1^n := w_1, w_2, \dots, w_n$. We are asked to output a sequence of punctuation symbols $S_1^n := s_1, s_2, \dots, s_n$ where s_i is attached to w_i to form a sentence like Figure 1(c). If there are no ambiguities, S_1^n is also abbreviated as S, similarly for w_1^n as w. We model the search of the best sequence of predicted punctuation symbols S^* as:

$$S^* = \text{argmax}_S P(S_1^n | w_1^n) \quad (1)$$

We introduce the transition-based parsing tree T to guide the punctuation prediction in Model (2), where parsing trees are constructed over the transcribed text while containing no punctuations.

$$S^* = \text{argmax}_S \sum_T P(T|w_1^n) \times P(S_1^n|T, w_1^n) \quad (2)$$

Rather than enumerate all possible parsing trees, we jointly optimize the punctuation prediction model and the transition-based parsing model with the form:

$$(S^*, T^*) = \text{argmax}_{(S,T)} P(T|w_1^n) \times P(S_1^n|T, w_1^n) \quad (3)$$

Let T_1^i be the constructed partial tree when w_1^i is consumed from the queue. We decompose the Model (3) into:

$$(S^*, T^*) = \text{argmax}_{(S,T)} \prod_{i=1}^n P\left(T_1^i \middle| T_1^{i-1}, w_1^i\right) \times P\left(s_i \middle| T_1^i, w_1^i\right) \quad (4)$$

It is noted that a partial parsing tree uniquely corresponds to a sequence of transition actions, and vice versa. Suppose T_1^i corresponds to the action sequence A_1^i and let a_i denote the last action in A_1^i. As the current word w_i can only be consumed from the queue by either *Shift* or *RightArc* according to Table 1, we have $a_i \in \{Shift, RightArc\}$. Thus, we synchronize the punctuation prediction with the application of *Shift* and *RightArc* during the parsing, which is explained by Model (5).

$$(S^*, T^*) = \text{argmax}_{(S,T)} \prod_{i=1}^n P\left(T_1^i, A_1^i \middle| T_1^{i-1}, w_1^i\right) \times P\left(s_i \middle| a_i, T_1^i, w_1^i\right) \quad (5)$$

The model is further refined by reducing the computation scope. When a full-stop punctuation is determined (i.e., a segmentation is formed), we discard the previous contexts and restart a new procedure for both parsing and punctuation prediction over the rest of words in the stream. In this way we are theoretically able to handle the unlimited stream of words without needing to always keep the entire context history of streaming words. Let b_i be the position index of last full-stop punctuation[1] before i, $T_{b_i}^i$ and $A_{b_i}^i$ the partial tree and corresponding action sequence over the words $w_{b_i}^i$, Model (5) can be rewritten by:

$$(S^*, T^*) = \text{argmax}_{(S,T)} \prod_{i=1}^n P\left(T_{b_i}^i, A_{b_i}^i \middle| T_{b_i}^{i-1}, w_{b_i}^i\right) \times P\left(s_i \middle| a_i, T_{b_i}^i, w_{b_i}^i\right) \quad (6)$$

With different computation of Model (6), we induce two joint models for punctuation prediction: the cascaded punctuation prediction model and the unified punctuation prediction model.

4.2 Cascaded punctuation prediction model (CPP)

In Model (6), the computation of two sub-models is independent. The first sub-model is computed based on the context of words and partial trees without any punctuation knowledge, while the computation of the second sub-model is conditional on the context from the partially built parsing tree $T_{b_i}^i$ and the transition action. As the words in the stream are consumed, each computation of transition actions is followed by a computation of punctuation prediction. Thus, the two sub-models are computed in a cascaded way, until the optimal parsing tree and optimal punctuation symbols are generated. We call this model the ***cascaded punctuation prediction model*** **(CPP)**.

4.3 Unified punctuation prediction model (UPP)

In Model (6), if the punctuation symbols can be deterministically inferred from the partial tree, $P(s_i|a_i, T_{b_i}^i, w_{b_i}^i)$ can be omitted because it is always 1. Similar to the idea of joint POS tagging and parsing (Hatori et al., 2011; Bohnet and Nivre, 2012), we propose attaching the punctuation prediction onto the parsing tree by embedding s_i into a_i. Thus, we extend the conventional transition actions illustrated in Table 1 to a new set of transition actions for the parsing, denoted by $\hat{A}$:

[1] Specially, b_i is equal to 1 if there are no previous full-stop punctuations.

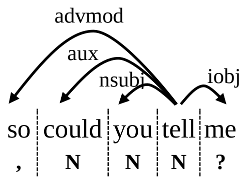
$$\hat{A} = \{LeftArc, Reduce\} \cup \{Shift(s) | s \in Q\} \\ \cup \{RightArc(s) | s \in Q\}$$

where Q is the set of punctuation symbols to be predicted, s is a punctuation symbol belonging to Q, *Shift*(s) is an action that attaches s to the current word on the basis of original *Shift* action in parsing, *RightArc*(s) attaches s to the current word on the basis of original *RightArc* action.

With the redefined transition action set $\hat{A}$, the computation of Model (6) is reformulated as:

$$(S^*, T^*) = \\ \operatorname{argmax}_{(S,T)} \prod_{i=1}^{n} P\left(T_{b_i}^{i}, \widehat{A}_{b_i}^{i} \middle| T_{b_i}^{i-1}, \widehat{A}_{b_i}^{i-1}, w_{b_i}^{i}\right) \quad (7)$$

Here, the computation of parsing tree and punctuation prediction is unified into one model where the sequence of transition action outputs uniquely determines the punctuations attached to the words. We refer to it as the ***unified punctuation prediction model*** **(UPP)**.



(a). Parsing tree and attached punctuation symbols

Shift(,), *Shift*(N), *Shift*(N), *LeftArc*, *LeftArc*, *LeftArc*, *Shift*(N), *RightArc*(?), *Reduce*, *Reduce*

(b). The corresponding sequence of transition actions

Figure 2. An example of punctuation prediction using the UPP model, where N is a null type punctuation symbol denoting no need to attach any punctuation to the word.

Figure 2 illustrates an example how the UPP model works. Given an input "*so could you tell me*", the optimal sequence of transition actions in Figure 2(b) is calculated based on the UPP model to produce the parsing tree in Figure 2(a). According to the sequence of actions, we can determine the sequence of predicted punctuation symbols like ",NNN?" that have been attached to the words shown in Figure 2(a). The final segmentation with the predicted punctuation insertion could be "*so, could you tell me?*".

4.4 Model training and decoding

In practice, the sub-models in Model (6) and (7) with the form of $P(Y|X)$ is computed with a linear model $Score(Y, X)$ as

$$Score(Y, X) = \Phi(Y, X) \cdot \vec{\lambda}$$

where $\Phi(Y, X)$ is the feature vector extracted from the output Y and the context X, and $\vec{\lambda}$ is the weight vector. For the features of the models, we incorporate the bag of words and POS tags as well as tree-based features shown in Table 2, which are the same as those defined in (Zhang and Nivre, 2011).

(a)	w_s; w_0; w_1; w_2; p_s; p_0; p_1; p_2; w_sp_s; w_0p_0; w_1p_1; w_2p_2; $w_sp_sw_0p_0$; $w_sp_sw_0$; $w_sp_sp_0$; $w_sw_0p_0$; $p_sw_0p_0$; w_sw_0; p_sp_0; p_0p_1; $p_sp_0p_1$; $p_0p_1p_2$;
(b)	$p_{sh}p_sp_0$; $p_sp_{sl}p_0$; $p_sp_{sr}p_0$; $p_sp_0p_{0l}$; w_sd; p_sd; w_0d; p_0d; w_sw_0d; p_sp_0d; w_sv_l; p_sv_l; w_sv_r; p_sv_r; w_0v_l; p_0v_l; w_{sh}; p_{sh}; t_s; w_{0l}; p_{0l}; t_{0l}; w_{0r}; p_{0r}; t_{0r}; w_{1l}; p_{1l}; t_{1l}; w_{sh2}; p_{sh2}; t_{sh}; w_{sl2}; p_{sl2}; t_{sl2}; w_{sr2}; p_{sr2}; t_{sr2}; w_{0l2}; p_{0l2}; t_{0l2}; $p_sp_{sl}p_{sl2}$; $p_sp_{sr}p_{sr2}$; $p_sp_{sh}p_{sh2}$; $p_0p_{0l}p_{0l2}$; w_sT_l; p_sT_l; w_sT_r; p_sT_r; w_0T_l; p_0T_l;

Table 2. (a) Features of the bag of words and POS tags. (b). Tree-based features. *w*–word; *p*–POS tag; *d*–distance between w_s and w_0; *v*–number of modifiers; *t*–dependency label; *T*–set of dependency labels; *s*, 0, 1 and 2 index the stack top and three front items in the queue respectively; *h*–head; *l*–left/leftmost; *r*–right/rightmost; *h*2–head of a head; *l*2–second leftmost; *r*2–second rightmost.

The training data for both the CPP and UPP models need to contain parsing trees and punctuation information. Due to the absence of annotation over transcribed speech data, we adapt the Treebank data for the purpose of model training. To do this, we remove all types of syntactic information related to punctuation symbols from the raw Treebank data, but record what punctuation symbols are attached to the words. We normalize various punctuation symbols into two types: Middle-paused punctuation (M) and Full-stop punctuation (F). Plus null type (N), there are three kinds of punctuation symbols attached to the words. Table 3 illustrates the normalizations of punctuation symbols. In the experiments, we did not further distinguish the type among full-stop punctuation because the question mark and the exclamation mark have very low frequency in Treebank data.

But our CPP and UPP models are both independent regarding the number of punctuation types to be predicted.

Punctuations	Normalization
Period, question mark, exclamation mark	Full-stop punctuation (F)
Comma, Colon, semicolon	Middle-paused punctuation (M)
Multiple Punctuations (e.g., !!!!?)	Full-stop punctuation (F)
Quotations, brackets, etc.	Null (N)

Table 3. Punctuation normalization in training data

As the feature templates are the same for the model training of both CPP and UPP, the training instances of CPP and UPP have the same contexts but with different outputs. Similar to work in (Zhang and Clark, 2008; Zhang and Nivre, 2011), we train CPP and UPP by generalized perceptron (Collins, 2002).

In decoding, beam search is performed to get the optimal sequence of transition actions in CPP and UPP, and the optimal punctuation symbols in CPP. To ensure each segment decided by a full-stop punctuation corresponds to a single parsing tree, two constraints are applied in decoding for the pruning of deficient search paths.

(1) Proceeding-constraint: If the partial parsing result is not a single tree, the full-stop punctuation prediction in CPP cannot be performed. In UPP, if *Shift*(F) or *RightArc*(F) fail to result in a single parsing tree, they cannot be performed as well.

(2) Succeeding-constraint: If the full-stop punctuation is predicted in CPP, or *Shift*(F) and *RightArc*(F) are performed in UPP, the following transition actions must be a sequence of *Reduce* actions until the stack becomes empty.

5 Experiments

5.1 Experimental setup

Our training data of transition-based dependency trees are converted from phrasal structure trees in English Web Treebank (LDC2012T13) and the English portion of OntoNotes 4.0 (LDC2011T03) by the Stanford Conversion toolkit (Marneffe et al., 2006). It contains around 1.5M words in total and consist of various genres including weblogs, web texts, newsgroups, email, reviews, question-answer sessions, newswires, broadcast news and broadcast conversations. To simulate the transcribed speech text, all words in dependency trees are lowercased and punctuations are excluded before model training. In addition, every ten dependency trees are concatenated sequentially to simulate a parsing result of a stream of words in the model training.

There are two test data sets used in our experiments. One is the English corpus of the IWSLT09 evaluation campaign (Paul, 2009) that is the conversional speech text. The other is a subset of the TDT4 English data (LDC2005T16) which consists of 200 hours of closed-captioned broadcast news.

In the decoding, the beam size of both the transition-based parsing and punctuation prediction is set to 5. The part-of-speech tagger is our re-implementation of the work in (Collins, 2002).

The evaluation metrics of our experiments are precision (*prec.*), recall (*rec.*) and F1-measure (F_1).

For the comparison, we also implement a baseline method based on the CRF model. It incorporates the features of bag of words and POS tags shown in Table 2(a), which are commonly used in previous related work.

5.2 Experimental results

We test the performance of our method on both the correctly recognized texts and automatically recognized texts. The former data is used to evaluate the capability of punctuation prediction of our algorithm regardless of the noises from speech data, as our model training data come from formal text instead of transcribed speech data. The usage of the latter test data set aims to evaluate the effectiveness of our method in real applications where lots of substantial recognition errors could be contained. In addition, we also evaluate the quality of our transition-based parsing, as its performance could have a big influence on the quality of punctuation prediction.

5.2.1 Performance on correctly recognized text

The evaluation of our method on correctly recognized text uses 10% of IWSLT09 training set, which consists of 19,972 sentences from BTEC (Basic Travel Expression Corpus) and 10,061 sentences from CT (Challenge Task). The average input length is about 10 words and each input contains 1.3 sentences on average. The evaluation results are presented in Table 4.

	Measure	Middle-Paused	Full-stop	Mixed
Baseline (CRF)	*prec.*	33.2%	81.5%	78.8%
	rec.	25.9%	83.8%	80.7%
	F_1	29.1%	82.6%	79.8%
CPP	*prec.*	51%	89%	89.6%
	rec.	50.3%	93.1%	92.7%
	F_1	50.6%	91%	91.1%
UPP	*prec.*	52.6%	93.2%	92%
	rec.	59.7%	91.3%	92.3%
	F_1	55.9%	92.2%	92.2%

Table 4. Punctuation prediction performance on correctly recognized text

We achieved good performance on full-stop punctuation compared to the baseline, which shows our method can efficiently process sentence segmentation because each segment is decided by the structure of a single parsing tree. In addition, the global syntactic knowledge used in our work help capture long range dependencies of punctuations. The performance of middle-paused punctuation prediction is fairly low between all methods, which shows predicting middle-paused punctuations is a difficult task. This is because the usage of middle-paused punctuations is very flexible, especially in conversional data. The last column in Table 4 presents the performance of the pure segmentation task where the middle-paused and full-stop punctuations are mixed and not distinguished. The performance of our method is much higher than that of the baseline, which shows our method is good at segmentation. We also note that UPP yields slightly better performance than CPP on full-stop and mixed punctuation prediction, and much better performance on middle-paused punctuation prediction. This could be because the interaction of parsing and punctuation prediction is closer together in UPP than in CPP.

5.2.2 Performance on automatically recognized text

Table 5 shows the experimental results of punctuation prediction on automatically recognized text from TDT4 data that is recognized using SRI's English broadcast news ASR system where the word error rate is estimated to be 18%. As the annotation of middle-paused punctuations in TDT4 is not available, we can only evaluate the performance of full-stop punctuation prediction (i.e., detecting sentence boundaries). Thus, we merge every three sentences into one single input before performing full-stop prediction. The average input length is about 43 words.

	Measure	Full-stop
Baseline (CRF)	*prec.*	37.7%
	rec.	60.7%
	F_1	46.5%
CPP	*prec.*	63%
	rec.	58.6%
	F_1	60.2%
UPP	*prec.*	73.9%
	rec.	51.6%
	F_1	60.7%

Table 5. Punctuation prediction performance on automatically recognized text

Generally, the performance shown in Table 5 is not as high as that in Table 4. This is because the speech recognition error from ASR systems degrades the capability of model prediction. Another reason might be that the domain and style of our training data mismatch those of TDT4 data. The baseline gets a little higher recall than our method, which shows the baseline method tends to make aggressive segmentation decisions. However, both precision and F1 score of our method are much higher than the baseline. CPP has higher recall than UPP, but with lower precision and F1 score. This is in line with Table 4, which consistently illustrates CPP can get higher recall on full-stop punctuation prediction for both correctly recognized and automatically recognized texts.

5.2.3 Performance of transition-based parsing

Performance of parsing affects the quality of punctuation prediction in our work. In this section, we separately evaluate the performance of our transition-based parser over various domains including the Wall Street Journal (WSJ), weblogs, newsgroups, answers, email messages and reviews. We divided annotated Treebank data into three data sets: 90% for model training, 5% for the development set and 5% for the test set. The accuracy of our POS-tagger achieves 96.71%. The beam size in the decoding of both our POS-tagging and parsing is set to 5. Table 6 presents the results of our experiments on the measures of UAS and LAS, where the overall accuracy is obtained from a general model which is trained over the combination of the training data from all domains.

We first evaluate the performance of our transition-based parsing over texts containing punctuations (TCP). The evaluation results show that our transition-based parser achieves state-of-the-art performance levels, referring to the best dependency parsing results reported in the shared task of SANCL 2012 workshop[2], although they cannot be compared directly due to the different training data and test data sets used in the experiments. Secondly, we evaluate our parsing model in CPP over the texts without punctuations (TOP). Surprisingly, the performance over TOP is better than that over TCP. The reason could be that we cleaned out data noises caused by punctuations when preparing TOP data. These results illustrate that the performance of transition-based parsing in our method does not degrade after being integrated with punctuation prediction. As a by-product of the punctuation prediction task, the outputs of parsing trees can benefit the subsequent text processing tasks.

	Data sets	UAS	LAS
Texts containing punctuations (**TCP**)	WSJ	92.6%	90.3%
	Weblogs	90.7%	88.2%
	Answers	89.4%	85.7%
	Newsgroups	90.1%	87.6%
	Reviews	90.9%	88.4%
	Email Messages	89.6%	87.1%
	Overall	90.5%	88%
Texts without punctuations (**TOP**)	WSJ	92.6%	91.1%
	Weblogs	92.5%	91.1%
	Answers	95%	94%
	Newsgroups	92.6%	91.2%
	Reviews	92.6%	91.2%
	Email Messages	92.9%	91.7%
	Overall	92.6%	91.2%

Table 6. The performance of our transition-based parser on written texts. UAS=unlabeled attachment score; LAS=labeled attachment score

6 Conclusion and Future Work

In this paper, we proposed a novel method for punctuation prediction of transcribed speech texts. Our approach jointly performs parsing and punctuation prediction by integrating a rich set of syntactic features. It can not only yield parse trees, but also determine sentence boundaries and predict punctuation symbols from a global view of the inputs. The proposed algorithm has linear complexity in the size of input, which can efficiently process the stream of words from a purely text processing perspective without the dependences on either the ASR systems or subsequent tasks. The experimental results show that our approach outperforms the CRF-based method on both the correctly recognized and automatically recognized texts. In addition, the performance of the parsing over the stream of transcribed words is state-of-the-art, which can benefit many subsequent text processing tasks.

In future work, we will try our method on other languages such as Chinese and Japanese, where Treebank data is available. We would also like to test the MT performance over transcribed speech texts with punctuation symbols inserted based on our method proposed in this paper.

References

B. Bohnet and J. Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In Proc. EMNLP-CoNLL 2012.

H. Christensen, Y. Gotoh, and S. Renals. 2001. Punctuation annotation using statistical prosody models. In Proc. of ISCA Workshop on Prosody in Speech Recognition and Understanding.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In Proc. EMNLP'02, pages 1-8.

B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tur, and M. Ostendorf. 2008. Punctuating speech for information extraction. In Proc. of ICASSP'08.

B. Favre, D. HakkaniTur, S. Petrov and D. Klein. 2008. Efficient sentence segmentation using syntactic features. In Spoken Language Technologies (SLT).

A. Gravano, M. Jansche, and M. Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In Proc. of ICASSP'09.

J. Hatori, T. Matsuzaki, Y. Miyao and J. Tsujii. 2011. Incremental joint POS tagging and dependency parsing in Chinese. In Proc. Of IJCNLP'11.

J. Huang and G. Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In Proc. Of ICSLP'02.

[2] https://sites.google.com/site/sancl2012/home/shared-task/results

J.H. Kim and P.C. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In Proc. of EuroSpeech'01.

Y. Liu, A. Stolcke, E. Shriberg, and M. Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In Proc. of ACL'05.

W. Lu and H.T. Ng. 2010. Better Punctuation Prediction with Dynamic Conditional Random Fields. In Proc. Of EMNLP'10. Pages 177-186.

M. Marneffe, B. MacCartney, C.D. Maning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In Proc. LREC'06.

E. Matusov, A. Mauser, and H. Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In Proc. of IWSLT'06.

S. Nakamura. 2009. Overcoming the language barrier with speech translation technology. In Science & Technology Trends - Quarterly Review. No. 31. April 2009.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In Proceedings of IWPT, pages 149–160, Nancy, France.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In Proc. COLING'04.

M. Paul. 2009. Overview of the IWSLT 2009 Evaluation Campaign. In Proceedings of IWSLT'09.

B. Roark, Y. Liu, M. Harper, R. Stewart, M. Lease, M. Snover, I. Shafran, B. Dorr, J. Hale, A. Krasnyanskaya, and L. Yung. 2006. Reranking for sentence boundary detection in conversational speech. In Proc. ICASSP, 2006.

A. Stolcke and E. Shriberg, "Automatic linguistic segmentation of conversational speech," Proc. ICSLP, vol. 2, 1996.

A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In Proc. of ICSLP' 98.

Takezawa, T. Morimoto, T. Sagisaka, Y. Campbell, N. Iida, H. Sugaya, F. Yokoo, A. Yamamoto, Seiichi. 1998. A Japanese-to-English speech translation system: ATR-MATRIX. In Proc. ICSLP'98.

Y. Zhang and J. Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In Proc. of ACL'11, pages 188-193.

Y. Zhang and S. Clark. A Tale of Two Parsers: investigating and combing graph-based and transition-based dependency parsing using beam-search. 2008. In Proc. of EMNLP'08, pages 562-571.