Combining Punctuation and Disfluency Prediction: An Empirical Study

Xuancong Wang[1,3] **Khe Chai Sim**[2] **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
[3]Human Language Technology, Institute for Infocomm Research, Singapore
xuancong84@gmail.com, {simkc, nght}@comp.nus.edu.sg

Abstract

Punctuation prediction and disfluency prediction can improve downstream natural language processing tasks such as machine translation and information extraction. Combining the two tasks can potentially improve the efficiency of the overall pipeline system and reduce error propagation. In this work[1], we compare various methods for combining punctuation prediction (PU) and disfluency prediction (DF) on the Switchboard corpus. We compare an isolated prediction approach with a cascade approach, a rescoring approach, and three joint model approaches. For the cascade approach, we show that the soft cascade method is better than the hard cascade method. We also use the cascade models to generate an n-best list, use the bi-directional cascade models to perform rescoring, and compare that with the results of the cascade models. For the joint model approach, we compare mixed-label Linear-chain Conditional Random Field (LCRF), cross-product LCRF and 2-layer Factorial Conditional Random Field (FCRF) with soft-cascade LCRF. Our results show that the various methods linking the two tasks are not significantly different from one another, although they perform better than the isolated prediction method by 0.5–1.5% in the F1 score. Moreover, the clique order of features also shows a marked difference.

1 Introduction

The raw output from automatic speech recognition (ASR) systems does not have sentence boundaries or punctuation symbols. Spontaneous speech also contains a significant proportion of disfluency. Researchers have shown that splitting input sequences into sentences and adding in punctuation symbols improve machine translation (Favre et al., 2008; Lu and Ng, 2010). Moreover, disfluencies in speech also introduce noise in downstream tasks like machine translation and information extraction (Wang et al., 2010). Thus, punctuation prediction (PU) and disfluency prediction (DF) are two important post-processing tasks for automatic speech recognition because they improve not only the readability of ASR output, but also the performance of downstream Natural Language Processing (NLP) tasks.

The task of punctuation prediction is to insert punctuation symbols into conversational speech texts. Punctuation prediction on long, unsegmented texts also achieves the purpose of sentence boundary prediction, because sentence boundaries are identified by sentence-end punctuation symbols: periods, question marks, and exclamation marks. Consider the following example,

How do you feel about the Viet Nam War ? Yeah , I saw that as well .

The question mark splits the sequence into two sentences. This paper deals with this task which is more challenging than that on text that has already been split into sentences.

The task of disfluency prediction is to identify word tokens that are spoken incorrectly due to speech disfluency. There are two main types of disfluencies: filler words and edit words. Filler words mainly include filled pauses (e.g., 'uh', 'um') and discourse markers (e.g., "I mean", "you know"). As they are insertions in spontaneous speech to indicate pauses or mark boundaries in discourse, they do not convey useful content information. Edit words are words that are spoken wrongly and then corrected by the speaker. For example, consider the following utterance:

[1]The research reported in this paper was carried out as part of the PhD thesis research of Xuancong Wang at the NUS Graduate School for Integrated Sciences and Engineering.

$$\text{I want a flight }\overbrace{\cancel{\textit{to Boston}}}^{\text{Edit}}\overbrace{\textit{uh I mean}}^{\text{Filler}}\overbrace{\underline{\text{to Denver}}}^{\text{Repair}}$$

The phrase "to Boston" forms the edit region to be replaced by "to Denver". The words "uh I mean" are filler words that serve to cue the listener about the error and subsequent corrections.

The motivation of combining the two tasks can be illustrated by the following two utterances:

~~I am uh~~ I am not going with you .

I am sorry . I am not going with you .

Notice that the bi-gram *"I am"* is repeated in both sentences. For the first utterance, if punctuation prediction is performed first, it might break the utterance both before and after "uh" so that the second-stage disfluency prediction will treat the whole utterance as three sentences, and thus may not be able to detect any disfluency because each one of the three sentences is legitimate on its own. On the other hand, for the second utterance, if disfluency prediction is performed first, it might mark *"I am sorry"* as disfluent in the first place and remove it before passing into the second-stage punctuation prediction. Therefore, no matter which task is performed first, certain utterances can always cause confusion.

There are many ways to combine the two tasks. For example, we can perform one task first followed by another, which is called the cascade approach. We can also mix the labels, or take the cross-product of the labels, or use joint prediction models. In this paper, we study the mutual influence between the two tasks and compare a variety of common state-of-the-art joint prediction techniques on this joint task.

In Section 2, we briefly introduce previous work on the two tasks. In Section 3, we describe our baseline system which performs punctuation and disfluency prediction separately (i.e., in isolation). In Section 4, we compare the soft cascade approach with the hard cascade approach. We also examine the effect of task order, i.e., performing which task first benefits more. In Section 5, we compare the cascade approach with bi-directional n-best rescoring. In Section 6, we compare the 2-layer Factorial CRF (Sutton et al., 2007) with the cross-product LCRF (Ng and Low, 2004), mixed-label LCRF (Stolcke et al., 1998), the cascade approach, and the baseline isolated prediction. Section 7 gives a summary of our overall findings. Section 8 gives the conclusion.

2 Previous Work

There were many works on punctuation prediction or disfluency prediction as an isolated task. For punctuation prediction, Huang and Zweig (2002) used maximum entropy model; Christensen et al. (2001) used finite state and multi-layer perceptron method; Liu et al. (2005) used conditional random fields; Lu and Ng (2010) proposed using dynamic conditional random fields for joint sentence boundary type and punctuation prediction; Wang et al. (2012) has added prosodic features for the dynamic conditional random field approach and Zhang et al. (2013) used transition-based parsing.

For disfluency prediction, Shriberg et al. (1997) uses purely prosodic features to perform the task. Johnson and Charniak (2004) proposed a TAG-based (Tree-Adjoining Grammar) noisy channel model. Maskey et al. (2006) proposed a phrase-level machine translation approach for this task. Georgila (2009) used integer linear programming (ILP) which can incorporate local and global constraints. Zwarts and Johnson (2011) has investigated the effect of using extra language models as features in the reranking stage. Qian and Liu (2013) proposed using weighted Max-margin Markov Networks (M3N) to balance precision and recall to further improve the F1-score. Wang et al. (2014) proposed a beam-search decoder which integrates M3N and achieved further improvements.

There were also some works that addressed both tasks. Liu et al. (2006) and Baron et al. (1998) carried out sentence unit (SU) and disfluency prediction as separate tasks. The difference between SU prediction and punctuation prediction is only in the non-sentence-end punctuation symbols such as commas. Stolcke et al. (1998) mixed sentence boundary labels with disfluency labels so that they do not predict punctuation on disfluent tokens. Kim (2004) performed joint SU and Interruption Point (IP) prediction, deriving edit and filler word regions from predicted IPs using a rule-based system as a separate step.

In this paper, we treat punctuation prediction and disfluency prediction as a joint prediction task, and compare various state-of-the-art joint prediction methods on this task.

3 The Baseline System

3.1 Experimental Setup

We use the Switchboard corpus (LDC99T42) in our experiment with the same train/develop/test split as (Qian and Liu, 2013) and (Johnson and Charniak, 2004). The corpus statistics are shown in Table 1. Since the proportion of exclamation marks and incomplete SU boundaries is too small, we convert all exclamation marks to periods and remove all incomplete SU boundaries (treat as no punctuation). In the Switchboard corpus, the utterances of each speaker have already been segmented into short sentences when used in (Qian and Liu, 2013; Johnson and Charniak, 2004). In our work, we concatenate the utterances of each speaker to form one long sequence of words for use as the input to punctuation prediction and disfluency prediction. This form of input where, utterances are not pre-segmented into short sentences, better reflects the real-world scenarios and provides a more realistic test setting for punctuation and disfluency prediction. Punctuation prediction also gives rise to sentence segmentation in this setting.

Data set	train	develop	test
# of tokens	1.3M	85.9K	65.5K
# of sentences	173.7K	10.1K	7.9K
# of sequences*	1854	174	134
# of edit words	63.6K	4.7K	3.7K
# of filler words	137.1K	9.6K	7.3K
# of Commas	52.7K	1.8K	2.1K
# of Periods	97.6K	6.5K	4.5K
# of Questions	6.8K	363	407
# of Exclamations	67	4	1
# of Incomplete	189	2	0

Table 1: Corpus statistics for all the experiments. *: each conversation produces two long/sentence-joined sequences, one from each speaker.

Our baseline system uses M3N (Taskar et al., 2004), one M3N for punctuation prediction and the other for disfluency prediction. We use the same set of punctuation and disfluency labels (as shown in Table 2) throughout this paper. To compare the various isolated, cascade, and joint prediction models, we use the same feature templates for both tasks as listed in Table 3. Since some of the feature templates require predicted filler labels and part-of-speech (POS) tags, we have trained a POS tagger and a filler predictor both using CRF (i.e., using the same approach as that in Qian and Liu (2013)). The same predicted POS tags and fillers are used for feature extraction in all the experiments in this paper for a fair comparison. The degradation on disfluency prediction due to the concatenation of utterances of each speaker is shown in Table 4. The pause duration features are extracted by running forced alignment on the corresponding Switchboard speech corpus (LDC97S62).

Task	Label	Meaning
Disfluency prediction	*E*	edit word
	F	filler word
	O	otherwise / fluent
Punctuation prediction	*Comma*	comma
	Period	full-stop
	QMark	question mark
	None	no punctuation

Table 2: Labels for punctuation prediction and disfluency prediction.

3.2 Features

We use the standard NLP features such as $F(w_{-1}w_0$=‘so that’), i.e., the word tokens at the previous and current node position are ‘so’ and ‘that’ respectively. Each feature is associated with a *clique order*. For example, since the clique order of this feature template is 2 (see Table 3), its feature functions can be $f(w_{-1}w_0$=‘so that’, y_0=‘*F*’, y_{-1}=‘*O*’, $t)$. The example has a value of 1 only when the words at node $t-1$ and t are ‘so that’, and the labels at node t and $t-1$ are ‘*F*’ and ‘*O*’ respectively. The maximum length of the y history is called the *clique order* of the feature (in this feature function, it is 2 since only y_0 and y_{-1} are covered). The feature templates are listed in Table 3. w_i refers to the word at the i^{th} position relative to the current node; *window size* is the maximum span of words centered at the current word that the template covers, e.g., $w_{-1}w_0$ with a window size of 9 means $w_{-4}w_{-3}$, $w_{-3}w_{-2}$, ..., w_3w_4; p_i refers to the POS tag at the i^{th} position relative to the current node; $w_{i\sim j}$ refers to any word from the i^{th} position to the j^{th} position relative to the current node, this template can capture word pairs which can potentially indicate a repair, e.g., “was ... is ...”, the speaker may have spoken any word(s) in between and it is very difficult for the standard n-gram features to capture all possible variations; $w_{i,\neq F}$ refers to the i^{th} non-filler word with respect to the current position, this template can extract n-gram features skipping filler words; the multi-pair comparison function $I(a, b, c, ...)$ indicates whether each pair (a and b, b and c, and so on) is identical, for example, if $a = b \neq c = d$, it will output "101" ('1' for being equal, '0' for being unequal), this feature template can capture consecutive word/POS repetitions which can further improve upon the standard repetition features; and *ngram-score* features are the natural logarithm of the following 8 probabilities: $P(w_{-3}, w_{-2}, w_{-1}, w_0)$, $P(w_0|w_{-3}, w_{-2}, w_{-1})$, $P(w_{-3}, w_{-2}, w_{-1})$, $P(\langle/s\rangle|w_{-3}, w_{-2}, w_{-1})$, $P(w_{-3})$, $P(w_{-2})$, $P(w_{-1})$ and $P(w_0)$ (where "$\langle/s\rangle$" denotes sentence-end).

Feature Template	**Window Size**	**Clique Order**
w_0	9	1
$w_{-1}w_0$	9	2
$w_{-2}w_{-1}w_0$	9	2
p_0	9	1
$p_{-1}p_0$	9	2
$p_{-2}p_{-1}p_0$	9	2
$w_0w_{-6\sim-1}$, $w_0w_{1\sim6}$	1	1
$I(w_i, w_j)$	21	2
$I(w_i, w_j, w_{i+1}, w_{j+1})$	21	2
$I(w_i, w_j)(w_i$ if w_i=$w_j)$	21	2
$I(p_i, p_j)$	21	3
$I(p_i, p_j, p_{i+1}, p_{j+1})$	21	3
$I(p_i, p_j)(p_i$ if p_i=$p_j)$	21	3
$p_{-1}w_0$	5	2
$w_{-1}p_0$	5	2
$w_{-2,\neq F}w_{-1,\neq F}$	1	2
$w_{-3,\neq F}w_{-2,\neq F}w_{-1,\neq F}$	1	2
$p_{-2,\neq F}p_{-1,\neq F}$	1	2
$p_{-3,\neq F}p_{-2,\neq F}p_{-1,\neq F}$	1	2
ngram-score features	1	3
pause duration before w_0	1	3
pause duration after w_0	1	3
transitions	1	3

Table 3: Feature templates for disfluency prediction, or punctuation prediction, or joint prediction for all the experiments in this paper.

The performance of the system can be further improved by adding additional prosodic features (Savova and Bachenko, 2003; Shriberg et al., 1998; Christensen et al., 2001) apart from pause durations. However, since in this work we focus on model-level comparison, we do not use other prosodic features for simplicity.

3.3 Evaluation and Results

Experiment	**F1 (PU)**	**F1 (DF)**
Short sentences, with precision/recall balancing, clique order of features up to 3, and labels {*E,F,O*}	N.A.	84.7
Short sentences, with precision/recall balancing, clique order of features up to 3, and labels {*E,O*}	N.A.	84.3
Join utterances into long sentences	71.1	79.2
Join utterances into long sentences + remove precision/recall balancing	71.1	78.2
Join utterances into long sentences + remove precision/recall balancing + reduce clique order of all features	68.5	76.4

Table 4: Baseline results showing the degradation by joining utterances into long sentences, removing precision/recall balancing, and reducing the clique order of features. All models are trained using M3N.

We use the standard F1 score as our evaluation metric and this is similar to that in Qian and Liu (2013). For training, we set the frequency pruning threshold to 5 to control the number of parameters. The regularization parameter is tuned on the development set. Since the toolkits used to run different experiments have slightly different limitations, in order to make fair comparisons across different toolkits, we do not use weighting to balance precision and recall when training M3N and we have reduced the clique order of transition features to two and all the other features to one in some of our experiments. Since the performance of filler word prediction on this dataset is already very high, (>97%), we only focus on the F1 score of edit word prediction in this paper when reporting the performance of disfluency prediction. Table 4 shows our baseline results. Our preliminary study shows the following general trends: (i) for disfluency prediction: joining utterances into long sentences will cause a 5–6% drop in F1 score; removing precision/recall balance in M3N will cause about 1% drop in F1 score; and reducing the clique order in Table 3 will cause about 1–2% drop in F1 score; and (ii) for punctuation prediction: removing precision/recall balance in M3N will cause negligible drop in F1 score; and reducing clique order will cause about 2–3% drop in F1 score. Conventionally, the degradation from reducing the clique orders can be mostly compensated by using the BIES (Begin, Inside, End, and Single) labeling scheme. In this work, for consistency and comparability across various experiments, we will stick to the same set of labels in Table 2.

4 The Cascade Approach

Instead of decomposing the joint prediction of punctuation and disfluency into two independent tasks, the cascade approach considers one task to be conditionally dependent on the other task such that the predictions are performed in sequence, where the results from the first step is used in the second step. In this paper, we compare two types of cascade: hard cascade versus soft cascade.

4.1 Hard Cascade

For the hard cascade, we use the output from the first step to modify the input sequence *before* extracting features for the second step. For PU→DF (PUnctuation prediction followed by DisFluency prediction), we split the input sequence into sentences according to the sentence-end punctuation symbols predicted by the first step, and then perform the DF prediction on the short/sentence-split sequences in the second step. For DF→PU, we remove the edit and filler words predicted by the first step, and then predict the punctuations using the cleaned-up input sequence. The hard cascade method may be helpful because the disfluency prediction on short/sentence-split sequences is better than on long/sentence-joined sequences (see the second and third rows in Table 4). On the other hand, the punctuation prediction on fluent text is more accurate than that on non-fluent text based on our preliminary study.

For this experiment, four models are trained using M3N without balancing precision/recall. For the first step, two models are trained on long/sentence-joined sequences with disfluent tokens - one for PU prediction and the other for DF prediction. These are simply the isolated baseline systems. For the second step, the DF prediction model is trained on the short/sentence-split sequences with disfluent tokens while the PU prediction model is trained on the long/sentence-joined sequences with disfluent tokens removed. Note that in the second step of DF→PU, punctuation labels are predicted only for the fluent tokens since the disfluent tokens predicted by the first step has already been removed. Therefore, during evaluation, if the first step makes a false positive by predicting a fluent token as an edit or filler, we set its punctuation label to the neutral label, *None*. All the four models are trained using the same feature templates as shown in Table 3. The regularization parameter is tuned on the development set.

4.2 Soft Cascade

For the soft cascade method, we use the labels predicted from the first step as additional features for the second step. For PU→DF, we model the joint probability as:

$$P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{PU}|\mathbf{x}) \times P(\text{DF}|\text{PU}, \mathbf{x}) \quad (1)$$

Likewise, we model the joint probability for DF→PU as:

$$P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{DF}|\mathbf{x}) \times P(\text{PU}|\text{DF}, \mathbf{x}) \quad (2)$$

For this experiment, four models are trained using M3N without balancing precision/recall. As with the case of hard cascade, the two models used in the first step are simply the isolated baseline systems. For the second step, in addition to the feature templates in Table 3, we also pass on the labels (at the previous, current and next position) predicted by the first step as three third-order-clique features. We also tune the regularization parameter on the development set to obtain the best model.

4.3 Experimental Results

Table 5 compares the performance of the hard and soft cascade methods with the isolated baseline systems. In addition, we have also included the results of using the true labels in place of the labels predicted by the first step to indicate the upper-bound performance of the cascade approaches. The results show that both the hard and soft cascade methods outperform the baseline systems, with the latter giving a better performance

Experiment	F1 for PU	F1 for DF
isolated baseline	71.1	78.2
hard cascade	71.2	79.1
hard cascade (using true labels)	72.6	83.5
soft cascade	71.6	79.6
soft cascade (using true labels)	72.1	82.7

Table 5: Performance comparison between the hard cascade method and the soft cascade method with respect to the baseline isolated prediction. All models are trained using M3N without balancing precision and recall.

(statistical significance at p=0.01). However, hard cascade has a higher upper-bound than soft cascade. This observation can be explained as follows.

For hard cascade, the input sequence is modified prior to feature extraction. Therefore, many of the features generated by the feature templates given in Table 3 will be affected by these modifications. So, provided that the modifications are based on the correct information, the resulting features will not contain unwanted artefacts caused by the absence of the sentence boundary information for the presence of disfluencies. For example, in *"do you do you feel that it was worthy"*, the punctuation prediction system tends to insert a sentence-end punctuation after the first *"do you"* because the speaker restarts the sentence.

If the disfluency was correctly predicted in the first step, then the hard cascade method would have removed the first *"do you"* and eliminated the confusion. Similarly, in "*I 'm sorry . I 'm not going with you tomorrow .* ", the first "*I 'm*" is likely to be incorrectly detected as disfluent tokens since consecutive repetitions are a strong indication of disfluency. In the case of hard cascade, PU→DF, the input sequence would have been split into sentences and the repetition feature would not be activated. However, since the hard cascade method has a greater influence on the features for the second step, it is also more sensitive to the prediction errors from the first step.

Another observation from Table 5 is that the improvement of the soft cascade over the isolate baseline is much larger on DF (1.4% absolute) than that on PU (only 0.5% absolute). The same holds true for the hard cascade, despite the fact that there are more DF labels than PU labels in this corpus (see Table 1) and the first step prediction is more accurate on DF than on PU. This suggests that their mutual influence is not symmetrical, in the way that the output from punctuation prediction provides more useful information for disfluency prediction than the other way round.

5 The Rescoring Approach

In Section 4, we have described that the two tasks can be cascaded in either order, i.e., PU→DF and DF→PU. However, the performance of the second step greatly depends on that of the first step. In order to reduce sensitivity to the errors made in the first step, one simple approach is to propagate multiple hypotheses from the first step to the second step to obtain a list of joint hypotheses (with both the DF and PU labels). We then rerank these hypotheses based on the joint probability and pick the best. We call this the rescoring approach. From (1) and (2), the joint probabilities can be expressed in terms of the probabilities generated by four models: $P(\text{PU}|\mathbf{x})$, $P(\text{DF}|\text{PU}, \mathbf{x})$, $P(\text{DF}|\mathbf{x})$, and $P(\text{PU}|\text{DF}, \mathbf{x})$. We can combine the four models to form the following joint probability function for rescoring:

$$\begin{aligned} P(\text{DF}, \text{PU}|\mathbf{x}) = P(\text{DF}|\mathbf{x})^{\alpha_1} \times P(\text{PU}|\text{DF}, \mathbf{x})^{\alpha_2} \\ \times P(\text{PU}|\mathbf{x})^{\beta_1} \times P(\text{DF}|\text{PU}, \mathbf{x})^{\beta_2} \end{aligned}$$

where α_1, α_2, β_1, and β_2 are used to weight the relative importance between (1) and (2); and between the first and second steps. In practice, the probabilities are computed in the log domain where the above expression becomes a weighted sum of the log probabilities. A similar rescoring approach using two models is described in Shi and Wang (2007).

The experimental framework is shown in Figure 1. For PU→DF, we first use $P(\text{PU}|\mathbf{x})$ to generate an n-best list. Then, for each hypothesis in the n-best list, we use $P(\text{DF}|\text{PU}, \mathbf{x})$ to obtain another n-best list. So we have n^2-best joint hypotheses. We do the same for DF→PU to obtain another n^2-best joint hypotheses. We rescore the $2n^2$-best list using the four models. The four weights α_1, α_2, β_1, and β_2 are tuned to optimize the overall F1 score on the development set. We used the MERT (minimum-error-rate training, (Och, 2003)) algorithm to tune the weights. We also vary the size of n.

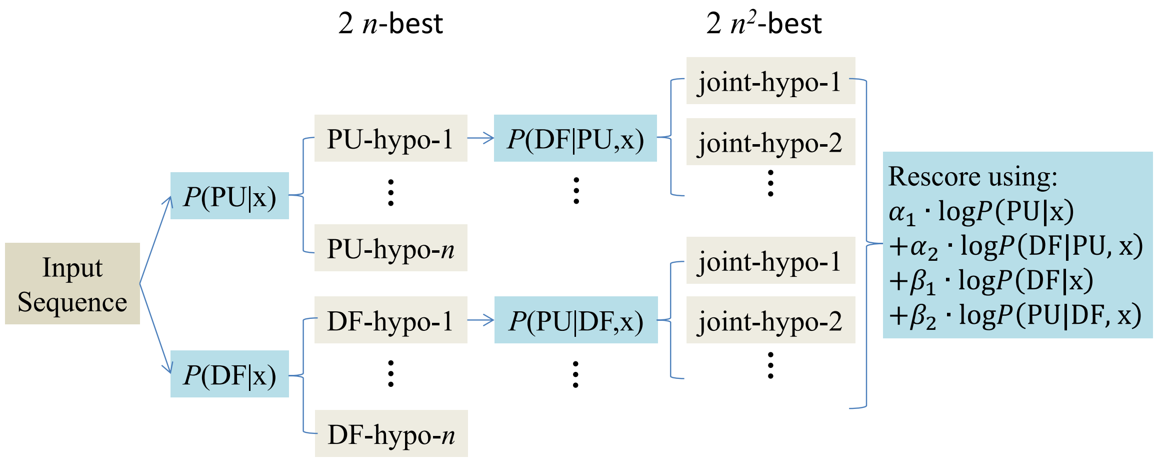

2 n-best
2 n^2-best
Input Sequence
P(PU|x)
P(DF|x)
PU-hypo-1
PU-hypo-n
DF-hypo-1
DF-hypo-n
P(DF|PU,x)
P(PU|DF,x)
joint-hypo-1
joint-hypo-2
joint-hypo-1
joint-hypo-2
Rescore using:
$\alpha_1 \cdot \log P(\text{PU}|\text{x})$
$+\alpha_2 \cdot \log P(\text{DF}|\text{PU, x})$
$+\beta_1 \cdot \log P(\text{DF}|\text{x})$
$+\beta_2 \cdot \log P(\text{PU}|\text{DF, x})$

Figure 1: Illustration of the rescoring pipeline framework using the four M3N models used in the soft-cascade method: $P(\text{PU}|\mathbf{x})$, $P(\text{DF}|\text{PU}, \mathbf{x})$, $P(\text{DF}|\mathbf{x})$ and $P(\text{PU}|\text{DF}, \mathbf{x})$

The results shown in Table 6 suggest that the rescoring method does not improve over the soft-cascade baseline. This can be due to the fact that we are using the same four models for the soft-cascade and the rescoring methods. It may be possible that the information contained in the two models for the soft-cascade PU→DF mostly overlaps with the information contained in the other two models for the soft-cascade DF→PU since all the four models are trained using the same features. Thus, no additional information is gained by combining the four models.

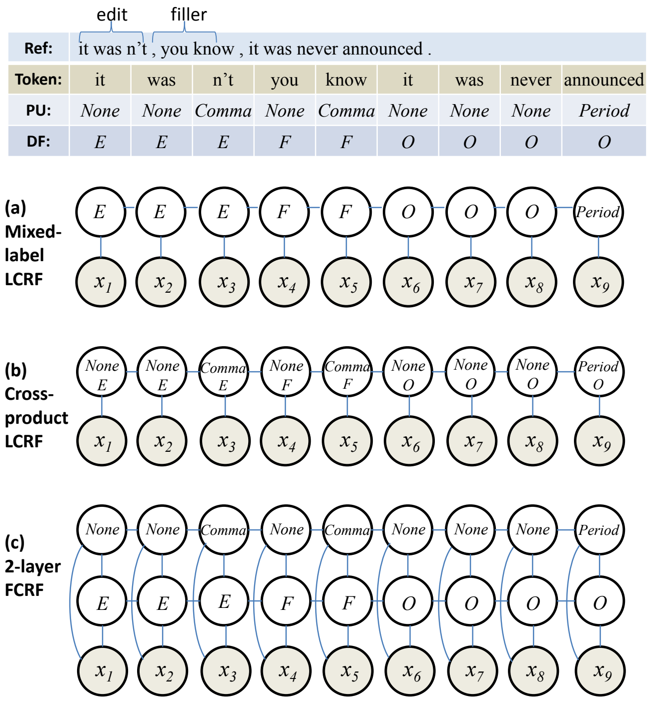
6 The Joint Approach

In this section, we compare 2-layer FCRF (Lu and Ng, 2010) with mixed-label LCRF (Stolcke et al., 1998) and cross-product LCRF on the joint prediction task. For the 2-layer FCRF, we use punctuation labels for the first layer and disfluency labels for the second layer (see Table 2). For the mixed-label LCRF, we split the neutral label $\{O\}$ into $\{$*Comma, Period, QMark, None*$\}$ so that we have six labels in total, $\{$*E, F, Comma, Period, QMark, None*$\}$. In this approach, disfluent tokens do not have punctuation labels because in real applications, if we just want to get the cleaned-up/fluent text with punctuations, we do not need to predict punctuations on disfluent tokens as they will be removed during the clean-up process. Since this approach does not predict punctuation labels on disfluent tokens, its punctuation F1 score is only evaluated on those fluent tokens. For the cross-product LCRF, we compose each of the three disfluency labels with the four punctuation labels to get 12 PU-DF-joint labels (Ng and Low, 2004). Figure 2 shows a comparison of these three models in the joint prediction of punctuation and disfluency. All the LCRF and FCRF models are trained using the GRMM toolkit (Sutton, 2006). We use the same feature templates (Table 3) to generate all the features for the toolkit. However, to reduce the training time, we have set clique order to 2 for the transitions and 1 for all other features. We tune the Gaussian prior variance on the development set for all the experiments to obtain the best model for testing.

Table 7 shows the comparison of results. On DF alone, the improvement of the cross-product LCRF over the mixed-label LCRF, and the improvement of the mixed-label LCRF over the isolated baseline are not statistically significant. However, if we test the statistical significance on the overall performance of both PU and DF, both the 2-layer FCRF and the cross-product LCRF perform better than the mixed-label LCRF. And we also obtain the same conclusion as Stolcke et al. (1998) that mixed-label LCRF performs better than isolated prediction. However, for the comparison between the 2-layer FCRF and the cross-product LCRF, although the 2-layer FCRF performs better than the cross-product LCRF on disfluency prediction, it does worse on punctuation prediction. Overall, the two methods perform about the same, their difference is not statistically significant. In addition, both the 2-layer FCRF and the cross-product LCRF slightly outperform the soft cascade method (statistical significance at p=0.04).

Experiment	F1 for PU	F1 for DF
isolated baseline	71.1	78.2
soft-cascade	71.6	79.6
rescore n=1	71.5 (72.5)	79.3 (81.1)
rescore n=2	71.2 (73.0)	79.3 (81.8)
rescore n=3	71.2 (73.3)	79.9 (82.6)
rescore n=4	71.2 (73.6)	79.8 (82.8)
rescore n=5	71.2 (73.9)	79.4 (83.3)
rescore n=6	71.1 (74.0)	79.6 (83.5)
rescore n=8	71.2 (74.2)	79.8 (84.0)
rescore n=10 *	71.2 (74.4)	79.8 (84.3)
rescore n=12	71.1 (74.5)	79.7 (84.6)
rescore n=15	71.2 (74.8)	79.8 (84.9)
rescore n=18	71.1 (74.9)	79.7 (85.1)
rescore n=25	70.7 (75.2)	79.3 (85.5)

Table 6: Performance comparison between the rescoring method and the soft-cascade method with respect to the baseline isolated prediction. The rescoring is done on $2n^2$ hypotheses. All models are trained using M3N without balancing precision and recall. Figures in the bracket are the oracle F1 scores of the $2n^2$ hypotheses. *:on the development set, the best overall result is obtained at $n = 10$.



edit
filler
Ref: it was n't , you know , it was never announced .
Token: it was n't you know it was never announced
PU: None None Comma None Comma None None None Period
DF: E E E F F O O O O
(a) Mixed-label LCRF
(b) Cross-product LCRF
(c) 2-layer FCRF

Figure 2: Illustration using (a) mixed-label LCRF; (b) cross-product LCRF; and (c) 2-layer FCRF, for joint punctuation (PU) and disfluency (DF) prediction. Shaded nodes are observations and unshaded nodes are variables to be predicted.

7 Discussion

In this section, we will summarise our observations based on the empirical studies that we have conducted in this paper.

Firstly, punctuation prediction and disfluency prediction do influence each other. The output from one task does provide useful information that can improve the other task. All the approaches studied in this work, which link the two tasks together, perform better than their corresponding isolated prediction baseline.

Secondly, as compared to the soft cascade, the hard cascade passes more information from the first step into the second step, and thus is much more sensitive to errors in the first step. In practice, unless the first step has very high accuracy, soft cascade is expected to do better than hard cascade.

Thirdly, if we train a model using a fine-grained label set but test it on the same coarse-grained label set, we are very likely to get improvement. For example:

- The edit word F1 for mixed edit and filler prediction using {*E, F, O*} is better than that for edit prediction using {*E, O*} (see the second and third rows in Table 4). This is because the former actually splits the *O* in the latter into *F* and *O*. Thus, it has a finer label granularity.

- Disfluency prediction using mixed-label LCRF (using label set {*E, F, Comma, Period, Question, None*}) performs better than that using isolated LCRF (using label set {*E, F, O*}) (see the second and fourth rows in Table 7). This is because the former dis-

Experiment	F1 for PU	F1 for DF
isolated baseline	68.7	77.0
soft cascade	69.0	77.5
mixed-label LCRF	69.0	77.2
cross-product LCRF	69.9	77.3
2-layer FCRF	69.2	77.8

Table 7: Performance comparison among 2-layer FCRF, mixed-label LCRF and cross-product LCRF, with respect to the soft-cascade and the isolated prediction baseline. All models are trained using GRMM (Sutton, 2006), with reduced clique orders.

tinguishes between different punctuations for fluent tokens and thus has a finer label granularity.

- Both the cross-product LCRF and 2-layer FCRF perform better than mixed-label LCRF because the former two distinguish between different punctuations for edit, filler and fluent tokens while the latter distinguishes between different punctuations only for fluent tokens. Thus, the former has a much finer label granularity.

From the above comparisons, we can see that increasing the label granularity can greatly improve the accuracy of a model. However, this may also increase the model complexity dramatically, especially when higher clique order is used. Although the joint approach (2-layer FCRF and cross-product LCRF) are better than the soft-cascade approach, they cannot be easily scaled up to using higher order cliques, which greatly limits their potential. In practice, the soft cascade approach offers a simpler and more efficient way to achieve a joint prediction of punctuations and disfluencies.

8 Conclusion

In general, punctuation prediction and disfluency prediction can improve downstream NLP tasks. Combining the two tasks can potentially improve the efficiency of the overall framework and minimize error propagation. In this work, we have carried out an empirical study on the various methods for combining the two tasks. Our results show that the various methods linking the two tasks perform better than the isolated prediction. This means that punctuation prediction and disfluency prediction do influence each other, and the prediction outcome in one task can provide useful information that helps to improve the other task. Specifically, we compare the cascade models and the joint prediction models. For the cascade approach, we show that soft cascade is less sensitive to prediction errors in the first step, and thus performs better than hard cascade. For joint model approach, we show that, when clique order of one is used, all the three joint model approaches perform significantly better than the isolated prediction baseline. Moreover, the 2-layer FCRF and the cross-product LCRF perform slightly better than the mix-label LCRF and the soft-cascade approach, suggesting that modelling at a finer label granularity is potentially beneficial. However, the soft cascade approach is more efficient than the joint approach when a higher clique order is used.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

Don Baron, Elizabeth Shriberg, and Andreas Stolcke. 2002. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Proc. of ICSLP.*

Heidi Christensen, Yoshihiko Gotoh, and Steve Renals. 2001. Punctuation annotation using statistical prosody models. In *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding.*

Benoit Favre, Ralph Grishman, Dustin Hillard, Heng Ji, Dilek Hakkani-Tur, and Mari Ostendorf. 2008. Punctuating speech for information extraction. In *Proc. of ICASSP.*

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proc. of NAACL.*

Jing Huang and Geoffrey Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proc. of INTERSPEECH.*

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *Proc. of ACL.*

Joungbum Kim. 2004. Automatic detection of sentence boundaries, disfluencies, and conversational fillers in spontaneous speech. Master dissertation of University of Washington.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proc. of ACL.*

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1526–1540.

Wei Lu and Hwee Tou Ng. 2010. Better punctuation prediction with dynamic conditional random fields. In *Proc. of EMNLP.*

Sameer Maskey, Bowen Zhou, and Yuqing Gao. 2006. A phrase-level machine translation approach for disfluency detection using weighted finite state transducers. In *Proc. of INTERSPEECH.*

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? Word-based or character-based? In *Proc. of EMNLP.*

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL.*

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proc. of NAACL.*

Guergana Savova, Joan Bachenko. 2003. Prosodic features of four types of disfluencies. In *ISCA Tutorial and Research Workshop on Disfluency in Spontaneous Speech.*

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer CRFs based joint decoding method for cascaded segmentation and labeling tasks. In *Proc. of IJCAI.*

Elizabeth Shriberg, Rebecca Bates and Andreas Stolcke. 1997. A prosody-only decision-tree model for disfluency detection. In *Proc. of Eurospeech.*

Elizabeth Shriberg, Andreas Stolcke, Daniel Jurafsky, Noah Coccaro, Marie Meteer, Rebecca Bates, Paul Taylor, Klaus Ries, Rachel Martin, and Carol Van Ess-Dykema. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? In *Language and speech 41, no. 3-4: 443-492.*

Andreas Stolcke, Elizabeth Shriberg, Rebecca A. Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gokhan Tur, and Yu Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proc. of ICSLP.*

Charles Sutton. 2006. GRMM: GRaphical Models in Mallet. *http://mallet.cs.umass.edu/grmm/*

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *Journal of Machine Learning Research*, 8: 693–723.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Proc. of NIPS.*

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. Automatic disfluency removal for improving spoken language translation. In *Proc. of ICASSP.*

Xuancong Wang, Hwee Tou Ng, and Khe Chai Sim. 2012. Dynamic conditional random fields for joint sentence boundary and punctuation prediction. In *Proc. of Interspeech.*

Xuancong Wang, Hwee Tou Ng, and Khe Chai Sim. 2014. A beam-search decoder for disfluency detection. In *Proc. of COLING.*

Dongdong Zhang, Shuangzhi Wu, Nan Yang, and Mu Li. 2013. Punctuation prediction with transition-based parsing. In *Proc. of ACL.*

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proc. of ACL.*