SYNTACTICALLY-INFORMED MODELS FOR COMMA PREDICTION

Benoit Favre[1], *Dilek Hakkani-Tür*[1], *Elizabeth Shriberg*[1,2]

[1]International Computer Science Institute, Berkeley, USA, {favre,dilek}@icsi.berkeley.edu
[2]SRI International, Menlo Park, USA, {ees}@speech.sri.com

ABSTRACT

Providing punctuation in speech transcripts not only improves readability, but it also helps downstream text processing such as information extraction or machine translation. In this paper, we improve by 7% the accuracy of comma prediction in English broadcast news by introducing syntactic features inspired by the role of commas as described in linguistics studies. We conduct an analysis of the impact of those features on other subsets of features (prosody, words...) when combined through CRFs. The syntactic cues can help characterizing large syntactic patterns such as appositions and lists which are not necessarily marked by prosody.

Index Terms— Speech Processing, Punctuation, Machine Learning

1. INTRODUCTION

Automatic speech recognition systems typically output an unannotated sequence of words. This stream of words is devoid of punctuation, capitalization, formatting, and other such structural information, making it challenging for humans to read and for machines to process. In particular, unannotated word sequences pose a challenge for downstream natural language processing modules that have been trained on *annotated* text. Various approaches for punctuation restoration have been proposed in the literature; they generally take advantage of lexical, prosodic and structural information, as in [1, 2, 3] for instance. Most previous work in punctuation restoration has focused on sentence boundaries, leaving apart sentence internal punctuation.

In this paper, we focus on subsentential punctuation, specifically on commas. Restoring commas have been shown to help Chinese part-of-speech tagging [4] and English information extraction [1, 2]. Comma restoration can also help in finding appositions ("*M*ark, my old friend, ..."), which can aid coreference resolution, thereby aiding generally further language processing tasks, such as question answering and summarization. Comma restoration is challenging, however, because the different contexts in which a comma is used (e.g., noun phrase lists, versus appositions, versus dates) do not necessarily share syntactic or prosodic features. Moreover, not all commas are mandatory or used stylistically in the same way across different individuals or different communication styles. This factor reduces inter-annotator agreement and can result in less accurate training and testing for learning systems.

Previous work on automatic comma detection has used hidden-event language models (HELMs) trained from large text corpora in combination with a local classifier (Boosting [4], Maxent [3], Neural Networks [5] or decision trees) trained from a smaller speech corpus. Such systems essentially rely on prosodic information (pause duration, pitch and energy discontinuities, phoneme duration) and lexical features (words and part-of-speech tags) for detecting commas at inter-word boundaries. In the linguistics community, the role of commas is often described according to syntactic patterns such as in [6] for Wall Street Journal text. A large number of such patterns include long-range dependencies that are out of scope for a local model such as that one applied in [4]. In the best case, part-of-speech tag sequences can characterize word boundaries at which a comma should not be added, such as between a determiner and a noun. But they are less able to capture the many positive instances in which a comma should be inserted.

In this paper we propose a novel approach to integrating more sophisticated syntactic features for automatic comma prediction. Specifically, we aim to capture long distance information. We test different feature subsets and their interaction with parsing features when modeled by a CRF classifier. The output of this classifier is then combined with a factored HELM (fHELM), an extension of the factored language model that can handle multiple streams of features (here, words and part-of-speech tags). The syntactic features result in an improvement of 7% relative over the basic set of features.

The paper is organized as follows: Section 2 describes the models involved in our comma prediction system. Section 3 details the different features used for classification, with a focus on our novel syntactic pattern features. We then present experiments on the TDT4 English Broadcast News corpus to show the benefit of the new features (Section 4). A discussion is provided in Section 5.

2. MODELS

In this work, comma annotation is formalized as a binary classification task for the boundary between two consecutive words. The classes denote the presence or absence of a comma at the boundary. We consider several sequence models to implement this classification problem: the Hidden-Event Language Model (HELM) that was quite successful for sentence segmentation [7], the factored Hidden Event Language Model (fHELM) — an extension of the HELM that can deal with several classes of features — and the Conditional Random Fields (CRF) [8].

The HELM is simply a regular n-gram language model trained on the sequence of words, with corresponding boundary class symbols (C=comma or N=no-comma), as in the following example:

... however C the N president C a N great N man C decided N to N sell N the N company ...

At decoding time a lattice is constructed with both hypotheses for each boundary, and the highest probability path is used to output comma predictions. More formally, the joint probability of the

This work is supported by the Defense Advanced Research Projects Agency (DARPA) GALE project, under Contract No. HR0011-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

word and event (C and N) sequence is estimated through the conditional probabilities of each word given their context. (Here, $X = X_1 \dots X_n$ is used to represent words and $Y = Y_1 \dots Y_n$ is used to represent classification labels, where $Y_i \in C, N$ for $i = 1, \dots n$).

$$P(Y, X) \sim \prod_i P(Y_i|X_i, Y_{i-1}, X_{i-1}, Y_{i-2}, X_{i-2}) \quad (1)$$

Regular back-off and smoothing rules can be used as for an n-gram language model. The factored HELM (fHELM) is an extension of the factored Language Model (fLM) [9] that adds support for hidden events. The observations used in the HELM can be extended to a set of factors (or features) to obtain a joint modeling of the words, the hidden events, and the factors. The originality of the fLM (and therefore of the fHELM) resides in the choice of the backoff paths when individual n-grams are under-represented in the training data. Intuitively, less reliable features should be dropped before more frequently-observed features.

The third model that we consider is the Conditional Random Fields [8] model. In this case, the conditional probability of sequence of labels given the observations is estimated through a log-linear model for which parameters can be trained efficiently.

$$P(Y|X) \sim \frac{1}{Z(X)} exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m}\lambda_i f_i(Y_{t-1}, Y_t, X_t)\right) \quad (2)$$

$$Z(X) = \sum_Y exp\left(\sum_{t=1}^{n}\sum_{i=1}^{m}\lambda_i f_i(Y_{t-1}, Y_t, X_t)\right)$$

where $f_i(\cdot)$ are feature functions depending on the observations and labels, and λ_i are the corresponding weights. While CRFs have been shown to have performance advantages over language models, unlike HELM or fHELM they have the disadvantage that they do not tend to scale to large datasets. HELM also have the useful property that they allow the integration of boundary-wise probabilities $P(Y_i|X_i)$. Such probabilities can also be generated from the hypotheses of a CRF using the forward-backward algorithm, allowing one to combine the two types of models.

In the experiments section, we use the SRILM toolkit [10] for running the HELM and fHELM, and the CRF++ [1] toolkit for CRFs.

3. FEATURES

According to [6], commas in English can be classified into several categories:

- Elements in a list: "a, b or c".
- Sentence initial and final elements: "Here, it usually snows a lot in winter, if I remember correctly".
- Nonrestrictive phrases (phrases that can be removed)
- Appositives: "The president, a great man, ...", "San Francisco, California".
- Interrupters (parenthetical elements).
- Quotations: "He was captured, said the officer".

The authors characterized the uses of commas in the Wall Street Journal corpus by means of 211 syntactic rules (parent → left context ¡C¿ right context). In order to capture such patterns in the ASR output, we trained a parser[2] on non-capitalized text with punctuation removed. As illustrated in Figure 1, each inter-word boundary is annotated with the highest constituents, in the syntactic parse tree, ending and beginning at that location, as well as with their parent. We create additional patterns in which constituents can be substituted by arbitrary constituents to obtain better coverage. For example, in Figure 1, "PP→IN, NP" is relaxed to "*→IN, NP", "PP→*, NP", "PP→IN, *" .... The syntactic patterns complement a set of features designed for sentence segmentation but which can be applied to detect any type of punctuation.

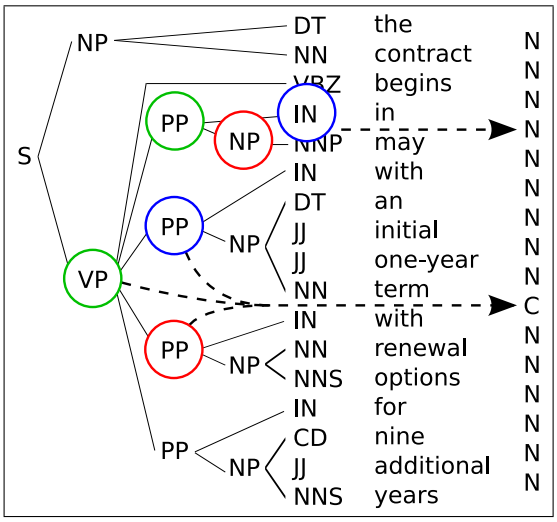

Fig. 1. Using the parse tree, each inter-word boundary is characterized by the highest constituents that begin or end at that location and by their parent (circled constituents). For instance, the features extracted for the boundary between "in" and "may" are: NP IN PP NP_IN NP_PP IN_PP NP_IN_PP.

Different types of features are used to characterize word boundaries and to train comma models: lexical (W), part-of-speech tags (T), syntactic (S) and prosodic (P) features. Apart from the syntactic features which require parsing of sentences, all other features are derived from the ICSI+ sentence segmentation system [11].

- Lexical: word bigrams across the boundary, words before and after the boundary.
- Part-of-speech tags: bigram of tags across the boundary, tag before and after the boundary. When used with lexical features, joint features (word+tag) are generated.
- Prosodic: pause duration, normalized pitch and energy on each side of the boundary and their difference across the boundary, normalized vowel and rhyme duration before and after the boundary.

While the HELM uses only lexical features, in our experiments the fHELM is trained on both part-of-speech tags and words. Only those two feature types are used in order to train models on a very large text corpus. This allows us to run only a part-of-speech tagger[3]. CRFs are trained on these features (for comparison purposes) and the full set of features extracted from speech data, including syntactic patterns.

[1] http://crfpp.sourceforge.net/

[2] We use the Berkeley parser to generate parse trees, available at http://nlp.cs.berkeley.edu

[3] Part-of-speech tagger available at http://www-tsujii.is.s.u-tokyo.ac.jp/

System	TDT4	Gigaword
HELM (W)	42.9	45.8
fHELM (W+T)	46.7	46.9
CRF (W+T)	48.3	n/a

Table 1. Comparison of HELM (words only), fHELM and CRF (words and part-of-speech tags) on trained on TDT4 and Gigaword data (F-measure).

4. EXPERIMENTS

The data set used for our experiments is a subset of the TDT4 English data.[4] It consists of 200 hours of close-captioned broadcast news. We use one million words of this set for training, 83k words as a development set, and 84k words as the test set. The prior probability for commas is 4.4% with an average of 0.85 commas per sentence. The data is recognized using SRI's English broadcast news ASR system; the word error rate on this dataset is estimated to be around 18%. It is important to note that in this work we use the true sentence boundary locations, to remove the confounding effect of sentence boundary errors on comma prediction. The punctuation-annotated data was created by automatically aligning ASR output with the reference text, and copying punctuation marks from the reference stream into the ASR stream. Commas aligned to a deleted word are push on the previous word and when the alignment is ambiguous (sequence of errors), commas are attached to the word that minimize the alignment cost. Because this alignment is not very reliable when the word error rate is high, we decided to only consider segments with a word error rate of less than 30%, effectively removing about 20% of the data (training and test).

A initial comparison we conduct is to compare the HELM, the fHELM and CRFs using the same basic features across different corpus sizes. The first corpus is the TDT4 data of about one million words on which CRF, fHELM and HELM are trained. The second is a subset of the English Gigaword corpus distributed by LDC, corresponding roughly to 500 million words. This corpus is tagged with part-of-speech tags, and is too large for CRF training. Results are presented in Table 1 with F-measure values for the TDT4 test set. One can observe that when a small amount of training data is available, there is a large improvement from HELM to fHELM and from fHELM to CRF. However, when much more data is available, the gap between HELM and fHELM reduces, while CRF cannot be applied. We expect that the more training data available, the better the coverage of words in the HELM, and thus the better the results.

In the second experiment, CRFs are trained with a range feature subsets, either excluding or including the new syntactic features. The results, depicted in Figure 2, show that the addition of syntactic features is beneficial to all feature subsets. There are large improvements when only one set is used (words, prosody or tags) and less improvement when multiple features are used jointly with the syntactic patterns. It is notable that the improvement from W+S (words and syntax) to W+T+S (words, part-of-speech tags and syntax) is relatively small, showing a potential redundancy between tags and syntax when words are available. On the other hand, syntax still adds 7% relative to the F-measure of W+T+P (all features). This is surprising given that prosody alone did not perform well. The results suggest that syntactic cases may help disambiguate prosodically-marked commas from pauses and hesitations.

Our final experiment explores the combination of a CRF trained with various subsets of features, with the HELM or the fHELM. Results for this experiment are detailed in Figure 3. For this experiment, we used the models trained on Gigaword, combined with CRF trained on TDT4 data on each subset of features. The merging of HELM or fHELM with CRF always results in performance improvements, and the weaker the CRF (that is, a CRF trained with smaller subsets of features), the greater the relative improvement. The best performance is achieved by a combination of CRF using all subsets of features with the fHELM. Syntactic features appear useful when added to each subset of features, implying a rather low redundancy with the information covered by the language model. Another interesting finding is that the fHELM is not consistently better than the HELM, suggesting the limited benefit of using multiple factors as the size of the training corpus gets larger.

[4]LDC publication LDC2005S11.

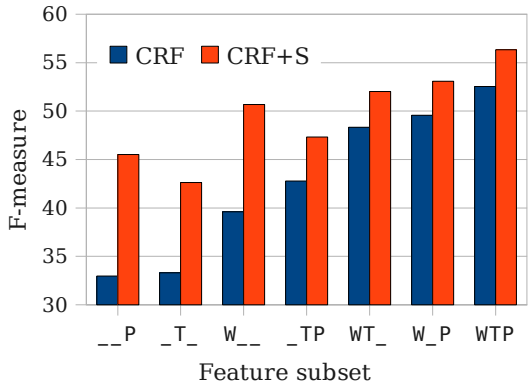

Fig. 2. Impact of adding syntactic features broken down by feature group (F-measure performance of CRF-based comma annotation on ASR words). Among the feature groups: words (W), part-of-speech tags (T) and prosodic features (P).

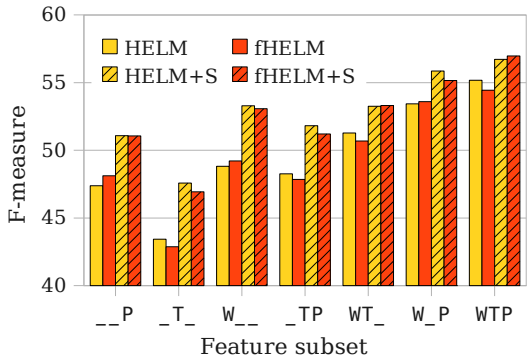

Fig. 3. CRF combined with HELM (trained on words) and fHELM (trained on words plus part-of-speech tags). "+S" variants are combined with a CRF containing syntactic features.

5. DISCUSSION

Figure 4 shows some of the most highly-weighted features in a CRF model trained on syntactic features only. The symbols in this figure are syntactic constituents according to a Penn Treebank notation (for example, S denotes a sentence, NP denotes a noun phrase). An asterisk represents the constituent that stands at that place in the parse tree, regardless of its identity. The first column represents relevant features for the "comma" class; the the second column refers

to the "no comma" class. Note that the weights (λ_i in equation 2) are jointly optimized with other features from the sequence, and are therefore somewhat difficult to interpret as is. Also, here we only represent the unigram classes (zeroth-order CRF) but weights are also impacted by a bigram of classes (ex: a comma followed by a non-comma).

Comma			**No-comma**		
parent→	left , right	λ_i	parent→	left right	λ_i
S→	* , NP	0.44	*→	CC *	0.92
S→	RB , NP	0.41	NP→	* *	0.88
*→	NP , NP	0.39	*→	DT *	0.70
S→	NP , NP	0.31	PP→	* *	0.66
S→	RB , *	0.29	VP→	* *	0.65
S→	NP , PP	0.29	*→	* NN	0.60
*→	JJ , JJ	0.28	*→	IN *	0.60
VP→	* , SQ	0.27	QP→	* *	0.59
SBAR→	RB , S	0.26	NP→	DT *	0.59
S→	* , PP	0.25	*→	* NNS	0.56
VP→	PP , VP	0.24	*→	* VP	0.56
VP→	VP , *	0.22	ADVP→	* *	0.56
VP→	* , SBARQ	0.22	*→	* RB	0.52
VP→	MD , PP	0.22	SQ→	* *	0.52
*→	MD , PP	0.22	WHNP→	* *	0.51
*→	ADVP , NP	0.22	*→	PRP$ *	0.48
S→	NP , SBAR	0.22	*→	* CC	0.48
*→	RB , S	0.21	SINV→	* *	0.48
*→	RB , NP	0.20	*→	VBZ *	0.47
ADJP→	RB , RB	0.20	*→	* NP	0.46

Fig. 4. Highly weighted features in the CRF model for both the "comma" and "no-comma" classes.

The first column of this table shows that obvious patterns where a comma is allowed are detected by the model. For instance, "S→RB, NP" or "S→RB, *" place a comma after an adverb at the beginning of a sentence. On the other hand, end-of-sentence phrases separated by a comma are represented by patterns such as "S→*, NP" or "S→*, PP". "*→JJ, JJ" could be a list of adjectives and "*→ NP, NP" is likely to by part of an apposition. A study of the second column revels that the model successfully learns that a noun phrase, a verb phrase or a prepositional phrase are unlikely to contain a comma (NP→* *, VP→* *, ...). Similarly, commas are hardly placed after coordinate conjunctions, determiners or prepositions (*→ IN *, *→ DT *, ...). Additionally, features such as "NP→DT *" reinforce the weight of the examples cited previously. The distribution of $lambda$ will be greatly modified by the inclusion of other features such as words as some boundary are going to be much easier to classify.

We also conducted a study of the errors of the system by considering comma insertions and deletions for which the system outputs high confidence scores. The error analysis suggests that a large quantity of these examples are either labeling errors (the human annotator missed a comma where it should have been), alignment errors (due to ASR errors, the punctuation is transfered from the reference to the wrong words) or non-mandatory commas (a comma is optional at that location, but the annotator decided to not use it). As soon as the confidence scores get lower, the system makes real errors due to unseen conditions or limited features. Moreover, the study showed that in some cases such as appositions and lists which are made of a series of commas, the classifier would get one of the commas right but miss the others. And indeed, we observed, for instance, that appositions are marked prosodically after the second comma, but the first one is void of such a mark. Here, the system would clearly benefit from a global decision in which the second comma becomes a clue of the presence of the first one.

6. CONCLUSIONS

Restoring commas in automatic speech transcripts is important for many natural language processing tasks that are trained on larger text corpora. In this article, we have aimed at incorporating syntactic features in a comma annotation system in order to capture long-range patterns that lead to the insertion of commas. We observed a significant gain of 7% on a broadcast news corpus, a rewarding gain. Nevertheless, we see many existing challenges for future work on this task. For instance, optional commas tend to confuse discriminative learning methods as training examples labeled as non-commas are going to look like real commas. Another problem comes from the fact that a pair of commas can enclose a long string of words requiring a scope too large for usual sequence models. We envision that the next generation of punctuation systems will consider the sentence as a whole instead of being limited to word boundaries.

7. REFERENCES

[1] B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tür, and M. Ostendorf, "Punctuating Speech for Information Extraction," *Proc. of ICASSP, Las Vegas, NV*, 2008.

[2] J. Makhoul, A. Baron, I. Bulyko, L. Nguyen, L. Ramshaw, D. Stallard, R. Schwartz, and B. Xiang, "The Effects of Speech Recognition and Punctuation on Information Extraction Performance," in *Eurospeech, Lisboa, Portugal*, 2005.

[3] J. Huang and G. Zweig, "Maximum entropy model for punctuation annotation from speech," in *Proc. of ICSLP, Denver, CO*, 2002.

[4] D. Hillard, Z. Huang, H. Ji, R. Grishman, D. Hakkani-Tür, M. Harper, M. Ostendorf, and W. Wang, "Impact of Automatic Comma Prediction on POS/Name Tagging of Speech," *Proc. of SLT*, pp. 58–61, 2006.

[5] H. Christensen, Y. Gotoh, and S. Renals, "Punctuation Annotation using Statistical Prosody Models," in *Proc. of Prosody in Speech Recognition and Understanding, Red Bank, NJ*, 2001.

[6] M. Bayraktar, B. Say, and V. Akman, "An Analysis of English Punctuation: The Special Case of Comma," *International Journal of Corpus Linguistics*, vol. 3, no. 1, pp. 33–57, 1998.

[7] A. Stolcke and E. Shriberg, "Automatic linguistic segmentation of conversational speech," in *Proc. of ICSLP*, Philadelphia, PA, 1996.

[8] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. of ICML*, 2001, pp. 282–289.

[9] D. Vergyri, K. Kirchhoff, K. Duh, and A. Stolcke, "Morphology-based language modeling for Arabic speech recognition," in *Proc. of ICSLP*, Jeju-Island, Korea, 2004.

[10] A. Stolcke, "SRILM–An extensible language modeling toolkit," in *Proc. of ICSLP*, Denver, CO, September 2002.

[11] M. Zimmermann, D. Hakkani-Tür, J. Fung, N. Mirghafori, L. Gottlieb, Y. Liu, and E. Shriberg, "The ICSI+ multi-lingual sentence segmentation system," in *Proc. of Interspeech*, Pittsburgh, PA, September 2006.