INTERSPEECH 2009 BRIGHTON

Phrase and Word Level Strategies for Detecting Appositions in Speech

Benoit Favre, Dilek Hakkani-Tür

International Computer Science Institute, Berkeley, USA

{favre,dilek}@icsi.berkeley.edu

Abstract

Appositions are grammatical constructs in which two noun phrases are placed side-by-side, one modifying the other. Detecting them in speech can help extract semantic information useful, for instance, for co-reference resolution and question answering. We compare and combine three approaches: word-level and phrase-level classifiers, and a syntactic parser trained to generate appositions. On reference parses, the phrase-level classifier outperforms the other approaches while on automatic parses and ASR output, the combination of the apposition-generating parser and the word-level classifier works best. An analysis of the system errors reveals that parsing accuracy and world knowledge are very important for this task.

Index Terms: Speech understanding, Punctuation, Apposition detection.

1. Introduction

Appositions are grammatical constructions usually involving two consecutive noun phrases, in which one of the phrases (the attribute) defines or modifies the other (the head). The head phrase forms a reference to an entity, and the attribute phrase characterizes that entity.

Detecting that two consecutive noun phrases are involved in an apposition relation is fundamental for language processing applications, such as question answering, co-reference resolution and textual entailment, because these noun phrases give a direct access to a semantic information between two entities, such as is_a relations. For example, we examined about 180 questions, querying biographic details of people (definition and reverse definition questions), from the Linguistic Data Consortium's question corpus formed for the DARPA GALE project (LDC2008E18). In a third of all the questions that have answers in speech data (from broadcast news and conversations shows), the answer were actually included in an apposition and could be found easily if the apposition was detected.

In text, the relation between the head and the attribute is usually marked by commas, which makes finding appositions relatively easy. On the other hand, due to lack of punctuation and capitalization in speech recognition output, it is more difficult to detect them in spoken documents. Without punctuation, a sentence like "I suggested John a gardener" is ambiguous: Is John the gardener? Or did I suggest him the name of a gardener? Furthermore, the performance of all language processing tasks that may help to detect appositions, such as parsing and part-of-speech tagging, often degrades on speech recognizer output, due to the lack of these punctuation marks, as well as ungrammatical sentences and disfluencies.

Previous work on spoken language processing has mainly focused on detecting commas on speech recognizer output [1, 2], and it was observed that finding commas in appositions was made difficult by the absence of prosodic cues on one side of the apposition. Comma detection (including commas in appositions) has been shown to help part-of-speech tagging [1] and relation detection [3]. [4] has studied the resolution of comma types (list, apposition, etc) in text and has shown positive effects on relation detection. However, neither those works, nor the abundant literature on co-reference resolution in text has focused on directly detecting appositions in speech when commas are not available.

In this paper, we explore apposition detection in news broadcasts of the OntoNotes corpus (release 2.9) English data. Specifically:

- We propose three methods for detecting appositions: the training of a parser to generate apposition labels, a word-level classifier on inside-outside-begin (IOB) labels, and a phrase-level classifier. Both classifiers use richer features than the parser, but the phrase-level approach requires phrases from a parser or a chunker (Section 2).
- We explore the performance and limits of those methods on various combinations of reference and automatically generated words and parse trees (Section 3).
- We analyze the results and suggest directions for future work on detecting appositions (Section 4).

2. Approach

We are interested in detecting not only the appositions but also their constituents, the head and the attribute. The attribute more often occurs before the head, but it can be after it as well. Occasionally, multiple attribute phrases refer to the same head, or multiple appositions are embedded in each other. In this paper, the head of the apposition is represented by the label "HEAD" and the attribute by the label "ATTR".

The most natural approach to detect appositions is to predict a parse tree in which phrases are to be labeled as "HEAD" and "ATTR". Our first approach is to concatenate the apposition label to the constituency label of each node in the parse tree and train the parser accordingly. Such an approach, however, depends highly on the quality of the parser, and cannot include speech-related features if the parser has not been designed for that purpose. Therefore, we contrast this approach with two other approaches: the classification of word-level labels using an IOB scheme and the classification of phrases from the parse tree using extended features compared to those used by the parser.

The word-level approach is implemented as a sequence model over inside-outside-begin (IOB) labels for the "HEAD" and "ATTR" classes. Figure 1 gives an example of IOB labeling. Since this model cannot represent embedded appositions, we only generate labels for the outermost annotation (embeddings are rather rare).

While the IOB framework is independent of the parser, its

Figure 1: *Example IOB labels for the fragment "...in the capital belgrade he...".*

Word	Label
...in	O
the	B-ATTR
capital	I-ATTR
belgrade	B-HEAD
he...	O

sequential structure is not very well suited for apposition annotation because, in order to account for words at the end of the apposition when predicting words at its beginning, either the scope of the model or the quantity of features have to be prohibitively extended. Additionally, a lot of classification power has to be spent to model out-of-apposition words.

For those reasons, we also consider phrase-level labels where each node of the parse tree is annotated with either "HEAD" or "ATTR" if it is involved in an apposition, or "O" if it's neither a head nor an attribute. Figure 2 shows an example annotation of a parse tree fragment with apposition labels. In "my brother John", the noun phrase "my brother" is annotated with "ATTR" and "John" is annotated with "HEAD". The rest of the nodes in the tree (including nodes under the HEAD and ATTR phrases) are annotated with "O".

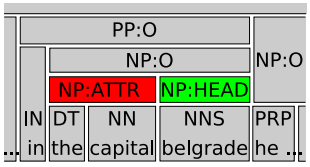

Figure 2: *Illustration of phrase level labeling on top of a fragment of the parse tree.*

3. Experiments

3.1. Data

The OntoNotes corpus [5] provides apposition annotation through its co-reference annotation effort. All noun phrases in appositions are annotated with head and attribute labels without being restricted to the Automatic Content Extraction (ACE) semantic classes. The choice of the head and the attribute annotation is guided by the genericness of the reference (for example, "a man" is more generic than "John"). In all experiments, we use news broadcasts from release 2.9 of the OntoNotes data (LDC2009E05). Table 1 shows the number of words, number of appositions and word error rate of the training, validation and test subsets that we use. The Automatic Speech Recognition (ASR) output is supplied by SRI's broadcast news speech recognizer [6], retrained to exclude data contemporary to the OntoNotes shows.

Table 1: *Statistics on the OntoNotes BN corpus: number of words and appositions, mean word error rate (WER).*

Stat	Train	Dev	Test
Words	162,246	16,530	18,945
No. Appositions	567	62	64
WER	20.6%	17.9%	19.4%

We had to "speechify" the reference transcriptions of the OntoNotes corpus in order to match the ASR output. First, numbers are converted to words, punctuation marks are removed, words are uppercased, and the tokenization is matched to the ASR tokenization (for instance, possessive marks are attached to the words and acronyms are split into individual letters).

Then, the reference words are aligned to the ASR output in order to obtain timing information and reference apposition labels for both conditions. The apposition reference annotations are transferred at the word level using the IOB labels. Since the ASR is not perfect, deleted words in the resulting alignment do not have time information (preventing the creation of speech-related features for those words) and inserted words do not have apposition labels. We apply simple rules to restore the apposition labels for inserted words: (1) Inside labels are propagated backwards to words without labels; (2) Unlabeled words between two apposition spans are included inside the first span; (3) All other unlabeled words get the outside label; (4) Spans that begin with inside labels are changed to the begin label.

For the word-level classifier, we convert the apposition annotation to IOB labels in the final step above. In the case of the phrase-level systems (the parser and the classifier), we align the appositions to the closest phrases in term of boundaries and length. Hence, the transfer is often approximate in the case of automatically estimated parse trees.

3.2. Systems

In our experiments, we use the Berkeley Parser [7], a generative constituency parser which automatically refines constituent labels at training time for improved accuracy. This parser obtains state-of-the-art results on English texts and therefore is expected to give realistic performance on speech data. We did not modify the parser in order to support speech, but rather trained it on "speechified" data from the OntoNotes corpus.

The classification experiments are run within a Conditional Random Field (CRF) framework for the IOB model. CRFs model the posterior probability of a sequence of labels ($y = y_0 \dots y_n$) for the input ($x = x_0 \dots x_n$):

$$P(y|x) \propto \frac{1}{Z(x)} exp\left[\sum_i \sum_k \lambda_k f_k(y_i, y_{i-1}, x)\right] \quad (1)$$

where $f_k(\cdot)$ is a feature function over a clique of labels (the current and previous label) and the examples, λ_k is a parameter of the model (the weight of the feature function), and $Z(\cdot)$ is a normalization factor. We implement that model through a first-order linear-chain CRF using the CRF++ toolkit [8].

Phrase-level decisions are made through an Adaboost [9] classifier that generates posterior probability estimations from weighted decision stumps (one-level decision trees):

$$P(y_i|x_i) \propto \left[1 + exp\left(-2m \sum_{k=1}^{m} w_k f_k(y_i, x_i)\right)\right]^{-1} \quad (2)$$

where $f_k(\cdot)$ is a decision stump (presence of a discrete feature or position relative to a threshold of a continuous feature) over a single feature, w_k is the weight given to that decision stump, and m is the number of decision stumps (or training iterations). Adaboost is trained by iterating over the selection of the best decision stump and the reweighing of examples on which the overall classifier at that iteration makes mistakes. The implementation used in our experiments is icsiboost [10]. We perform 1,000

training iterations and determine a decision threshold for each class in order to maximize F-score on the development set.

We do not restrict the phrase-level classifier to noun phrases so that it can be trained on erroneous parse trees and ASR output (all phrases are candidates). Additionally, we ensure that a head is always in the vicinity of an attribute by post-processing the output, which increases precision.

3.3. Features

The Features for word-level classification are summarized here:

- Word and part-of-speech n-grams (from 1- to 3-gram) from two words before and after the current word.
- Quantified pause duration before and after the word.
- Syntactic label of the highest node in the parse tree beginning/ending on the previous, current and next word.
- IOB named-entity label of the word (persons, organizations, geo-political entities, locations, dates, and time).

These features are used for both zero- order (current label) and first order (previous and current label) predictions. All features occurring fewer than 5 times in the training data are dropped.

Features for the phrase-level model are as follows:

- Positional and non-positional n-gram of words and part-of-speech tags for the node (from 1- to 4-gram).
- Quantified pause duration before and after the phrase.
- Syntactic labels of the node, its previous and next siblings, and its parent.
- Word and part-of-speech n-grams for the words before and after the constituent.
- Labels of the named entities found in the node and the named-entity label of the node itself.

For named-entity annotation, we train an IOB classifier (CRF++ toolkit) on the OntoNotes corpus training data (mono-case, no punctuation) using the same features as in [11], resulting in an F-score of 74.25% on the test set. The Berkeley Parser is also trained on the OntoNotes speechified data, resulting in a bracketing F-score of 84.14% and a part-of-speech tagging accuracy of 95.28%. For experiments on automatically generated parse trees and ASR output, we first process the training data with the parser and the named-entity tagger in order to reproduce the kind of errors that are seen at test time. Normally, one would train those components on different data but, given the low quantity of in-domain data available, we prefer to reuse the OntoNotes training set.

3.4. Results

We first look at baselines and oracles. A simple baseline annotates all consecutive pairs of noun phrases as "HEAD, ATTR" (the most frequent form of apposition). The oracle result looks at whether the phrase boundaries of both the head and the attribute are found accurately in the parse tree. We also compare the oracle from the parser to an oracle from a noun phrase chunker trained on CoNLL data (mono-cased, no punctuation) with the same classifier and features as for named entity extraction. That chunker performs at 91.07% (F-score) on the CoNLL test set when case information and punctuation are removed (93.44% otherwise). Table 2 lists the corresponding results for the baseline and the oracle. We observe that the baseline performs poorly, even on reference parses, showing the difficulty of finding appositions. By looking at the oracle results, we see that the parser is a big hit to the performance ceiling, if we completely rely on its phrase boundaries. Additionally, it seems inadequate to use a chunker to hypothesize appositions.

Table 2: *Baseline and oracle F-score on reference words (REF) and (ASR) for the gold parses, the parser and the chunker.*

System	Words	Ref. Parses	Parser	Chunker
Baseline	REF	17.33	7.66	-
Oracle	REF	100.00	76.74	48.83
Baseline	ASR	-	1.95	-
Oracle	ASR	-	65.78	27.63

The next set of results concerns the use of classifiers for detecting appositions. Table 3 presents F-scores on the "HEAD" and "ATTR" classes for three conditions: reference words with reference parses (REF-manual), reference words with automatic parses (REF-auto), and ASR words with automatic parses (ASR-auto). Our first system (BP) is the parser trained with apposition labels concatenated to syntactic labels (if the head is a noun phrase, it is labeled NP-HEAD). The second system (IOB) is the word-level classifier, and the third system (PH) is the phrase-level classifier. The results show that the parser is very competitive at finding appositions, especially on ASR output (it has a better precision than the classifiers). The word-level classifier, though, consistently performs more poorly than the phrase-based classifier, on reference parses, automatic parses and ASR output.

Table 3: *F-score (on HEAD and ATTR) for various systems. BP is the Berkeley Parser trained to generate appositions; IOB is the word-level classifier and PH is the phrase-level classifier.*

System	REF-manual	REF-auto	ASR-auto
BP	-	41.38	39.73
IOB	48.09	32.76	17.39
PH	61.54	40.41	26.67

Next, we look at combining the apposition annotation of the parser with the other systems, by using the parse trees with HEAD and ATTR labels in place of the original parse trees. All features that make use of the syntactic labels now benefit from the hypothesized apposition labels. The results displayed in Table 4 show that an improvement can be obtained for both classifiers with even higher results for the word-level systems. These results are not surprising because the word-level information is complementary to the information used by the parser while the overlap is higher at the phrase level. The fact that combining the parser with the phrase-level classifier results in lower scores than the uncombined version can be explained by the processing of the training data with a parser trained on the same data (it has a bracketing F-score of 94.30% on the training set). The classifier probably over-relies on the parser's decisions and does not correct them as much as the IOB model. On ASR output, even on the training set, the parser has a lower accuracy which minimizes the mismatch with the test set.

Table 4: *F-scores for combined systems from Table 3.*

System	REF-auto	ASR-auto
BP+IOB	42.31	42.59
BP+PH	34.22	40.35

4. Discussion

We performed an analysis of the errors of the phrase-level classifier on the test set. Results are reported in Table 5. For this analysis, we disabled decision post-processing (removal of lonely heads and attributes), which decreases precision, but is expected to give better insight into the errors of the system. A lot of insertions and misses in both ASR and REF are due to a poor choice of the decision thresholds on the development set. The use of a larger set could probably improve the situation. Decisions to label spurious heads or attributes often come from the presence of proper names in the vicinity of the phrase, and misses seem to originate from totally unseen conditions where either the named entity tagger missed an entity or only common nouns are used in the references. Errors in the parse tree, including phrase segmentation issues, have the strongest effect on the ASR side, but even when using reference trees (REF), the classifier can be confused by structures that look just like appositions due to the absence of punctuation, such as lists or when a comma is expected. As already observed in the results of the system, word error rate does not dominate the source of apposition errors. Finally, a share of the mistakes comes from labeling errors and alignment problems, when the phrase structure does not match the apposition annotation. Some candidates, such as "[syria's president] [bashar al assad]", are ambiguous and can be interpreted either as an apposition or as the use of a nominal modifier. Only the acoustics might tell the intent of the speaker. Some appositions also seem very difficult to find without world knowledge: In "[the son of milosevic] [marko]", one must know that two people are referred to instead of one (if milosevic can be a first name, and marko a family name).

The small size of the training corpus seems to be a big factor in the errors made by the system. We presume that applying unsupervised learning and introducing some world knowledge in the approach could help spot possible relations between noun phrases (for example, "the president" followed by a person name is a likely relation while "the book" followed by the same name is unlikely).

Table 5: *Analysis of the errors by type and by expected reason (when one was found), for the phrase-level system on reference text with manual parses (REF) and ASR output with automatic parses (ASR).*

Error	REF	ASR
Del. apposition	48%	42%
Del. attribute	2%	0%
Del. head	24%	9%
Ins. apposition	10%	9%
Ins. attribute	16%	30%
Ins. head	10%	0%
Swapped appos.	2%	1%

Reason	REF	ASR
Unseen words	10%	3%
Labeling	4%	5%
Parse tree	8%	30%
Proper name	12%	12%
Alignment	2%	7%
Segmentation	2%	5%
ASR errors	0%	9%

It is also obvious that improving parsing would greatly improve apposition detection, thanks to the phrase model, even though the apposition generating parser is unlikely to be improved unless more training data is available. We performed an extra experiment where we train a parser with extra symbols in the word sequence to mark the beginnings and ends of appositions. The performance of this parser on sentences containing appositions increases from 79.50% to 85.95% compared to a parser that does not know about appositions. Even if this looks like a chicken-and-egg problem, a system accurate enough to generate appositions without using parse trees would be able to help parsing by a large amount.

Finally, in the phrase-level approach, annotating the head and the attribute separately seems inadequate. Predicting properly the whole apposition is a structure prediction problem that follows neither a sequence nor a tree (especially when the parse tree is not accurate). We might have to turn to general graph learning models so that each phrase is treated in function of the phrases before and after it, independent of the parse tree.

5. Conclusion

This work presents approaches for identifying appositions in speech. We show that on reference parses, a phrase-level classifier performs much better than predicting labels at the word level. This trend is also true on automatic parses and ASR output if no other information is available. We propose training the parser so that it generates apposition labels, resulting in higher performance. The best performance, though, is obtained by combining the apposition-generating parser with the word-level model because they have different scopes. The phrase-level approach seems to be more affected by parsing errors and requires more training data to be effective. As future work, we plan to apply unsupervised learning to enrich the coverage of our models (by self-training the parser on ASR output, for instance), and to work towards a structured model where the decisions taken for a phrase affect the decisions on the neighboring phrases.

Acknowledgments: 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.

6. References

[1] 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," in *IEEE-SLT*, 2006, pp. 58–61.

[2] B. Favre, D. Hakkani-Tür, and S. Shriberg, "Syntactically informed models for comma prediction," in *ICASSP, Taipei, Taiwan*, 2009.

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

[4] V. Srikumar, R. Reichart, M. Sammons, A. Rappoport, and D. Roth, "Extraction of entailed semantic relations through syntax-based comma resolution," in *ACL-08: HLT*, June 2008, pp. 1030–1038.

[5] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "OntoNotes: The 90% Solution," in *HLT-NAACL*, 2007, p. 57.

[6] A. Venkataraman, A. Stolcke, W. Wang, D. Vergyri, V. Gadde, and J. Zheng, "SRIs 2004 broadcast news speech to text system," in *EARS Rich Transcription 2004 Workshop*, 2004.

[7] S. Petrov and D. Klein, "Learning and inference for hierarchically split PCFGs," in *AAAI*, vol. 22, no. 2, 2007, p. 1663.

[8] T. Kudo, "CRF++: Yet Another CRF Toolkit," 2005. [Online]. Available: http://crfpp.sourceforge.net

[9] R. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Machine learning*, vol. 39, no. 2, pp. 135–168, 2000.

[10] B. Favre, "icsiboost: an opensource implementation of BoosTexter," 2007. [Online]. Available: http://code.google.com/p/icsiboost

[11] T. Kudoh and Y. Matsumoto, "Use of support vector learning for chunk identification," in *CoNLL*, 2000, pp. 142–144.