Projecting the Knowledge Graph to Syntactic Parsing

Andrea Gesmundo and **Keith B. Hall**
Google, Inc.
{agesmundo,kbhall}@google.com

Abstract

We present a syntactic parser training paradigm that learns from large scale Knowledge Bases. By utilizing the Knowledge Base context only during training, the resulting parser has no inference-time dependency on the Knowledge Base, thus not decreasing the speed during prediction. Knowledge Base information is injected into the model using an extension to the Augmented-loss training framework. We present empirical results that show this approach achieves a significant gain in accuracy for syntactic categories such as coordination and apposition.

1 Introduction

Natural Language Processing systems require large amounts of world knowledge to achieve state-of-the-art performance. Leveraging Knowledge Bases (KB) provides allows us to inject human curated world-knowledge into our systems. As these KBs have increased in size, we are now able to leverage this information to improve upon the state-of-the-art. Large scale KB have been developed rapidly in recent years, adding large numbers of entities and relations between the entities. Such entities can be of any kind: an object, a person, a place, a company, a book, etc. Entities and relations are stored in association with relevant data that describes the particular entity or relation; for example, the name of a book, it's author, other books by the same author, etc.. Large scale KB annotation efforts have focused on the collection of both current and historical entities, but are biased towards the contemporary entities.

Of the many publicly available KBs, we focus this study on the use of Freebase[1]: a large collaborative Knowledge Base composed and updated by a member community. Currently it contains roughly 40 million entities and 1.1 billion relations.

The aim of the presented work is to use the information provided by the KB to improve the accuracy of the statistical dependency parsing task (Kubler et al., 2009). In particular we focus on the recognition of relations such as coordination and apposition. This choice is motivated by the fact that the KB stores information about real-world entities while many of the errors associated with coordination and apposition is the lack of knowledge of these real-world entities.

We begin by defining the task (section 2). Following, we present the modified augmented-loss training framework (section 3). In section 4, we define how the Knowledge Base data is integrated into the training process. Finally, we discuss the empirical results (section 5).

2 Task

Apposition is a relation between two adjacent noun-phrases, where one noun-phrase specifies or modifying the other. For example, in the sentence "My friend Anna", the nouns "friend" and "Anna" are in apposition. Coordination between nouns relates two or more elements of the same kind. The coordination is often signaled by the appearance of a coordinating conjunction. For example, in the sentence "My friend and Anna", the nouns "friend" and "Anna" are in coordination. The semantic difference between the two relations is that the nouns in apposition refer to the same entity,

[1] www.freebase.com

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 28–32,
Gothenburg, Sweden, April 26-30 2014.

while the nouns in coordination refer to distinct entities of the same kind or sharing some properties.

Statistical parsers are inaccurate in classifying relations involving proper nouns that appear rarely in the training set. In the sentence:

"They invested in three companies, Google, Microsoft, and Yahoo."

"companies" is in apposition with the coordination "Google, Microsoft, and Yahoo". By integrating the information provided by a large scale KB into the syntactic parser, we attempt to increase the ability to disambiguate the relations involving these proper nouns, even if the parser has been trained on a different domain.

3 Model

We present a Syntactic Parsing model that learns from the KB. An important constraint that we impose, is that the speed of the Syntactic Parser must not decrease when this information is integrated. As the queries to the KB would significantly slow down the parser, we limit querying the KB to training. This constraint reduces the impact that the KB can have on the accuracy, but allows us to design a parser that can be substituted in any setting, even in the absence of the KB.

We propose a solution based on the Augmented-loss framework (Hall et al., 2011a). Augmented-loss is a training framework for structured prediction tasks such as parsing. It can be used to extend a standard objective function with additional loss-functions and be integrated with the structured perceptron training algorithm. The input is enriched with multiple datasets each associated with a loss function. The algorithm iterates over the datasets triggering parameter updates whenever the loss function is positive.

Loss functions return a positive value if the predicted output is "worse" than the gold standard. Augmented-loss allows for the inclusion of multiple objective functions, either based on intrinsic parsing quality or task-specific extrinsic measures of quality. In the original formalization, both the intrinsic and extrinsic losses require gold standard information. Thus, each dataset must specify a gold standard output for each input.

We extend the Augmented-loss framework to apply it when the additional dataset gold-standard is unknown. Without the gold standard, it is not possible to trigger updates using a loss function. Instead, we use a *sampling function*, $S(\cdot)$, that is defined such that: if $\hat{y}$ is a candidate parse tree, then $S(\hat{y})$ returns a parse tree that is guaranteed to be "not worse" than $\hat{y}$. In other words:

$$L^S(\hat{y}, S(\hat{y})) \geq 0 \quad (1)$$

Where the $L^S(\cdot)$ is the *implicit loss function*. This formalization will allow us to avoid stating explicitly the loss function. Notice that $S(\hat{y})$ is not guaranteed to be the "best" parse tree. It can be any parse tree in the search space that is "not worse" than $\hat{y}$. $S(\hat{y})$ can represent an incremental improvement over $\hat{y}$.

Algorithm 1 Augmented-loss extension

1: {Input loss function: $L(\cdot)$}
2: {Input sample function: $S(\cdot)$}
3: {Input data sets}:
4: $D^L = \{d_i^L = (x_i^L, y_i^L) \mid 1 \leq i \leq N^L\}$
5: $D^S = \{d_i^S = (x_i^S) \mid 1 \leq i \leq N^S\}$
6: $\theta = \vec{0}$
7: **repeat**
8: **for** $i = 1 \ldots N^L$ **do**
9: $\hat{y} = F_\theta(x_i^L)$
10: **if** $L(\hat{y}, y_i^L) > 0$ **then**
11: $\theta = \theta + \Phi(y_i^L) - \Phi(\hat{y})$
12: **end if**
13: **end for**
14: **for** $i = 1 \ldots N^S$ **do**
15: $\hat{y} = F_\theta(x_i^S)$
16: $y^* = S(\hat{y})$
17: $\theta = \theta + \Phi(y^*) - \Phi(\hat{y})$
18: **end for**
19: **until** converged
20: {Return model θ}

Algorithm 1 summarizes the extension to the Augmented-loss algorithm.

The algorithm takes as input: the loss function $L(\cdot)$; the sample function $S(\cdot)$; the loss function data samples D^L; and the sample function data samples D^S. Notice that D^L specifies the gold standard parse y_i^L for each input sentence x_i^L. While, D^S specifies only the input sentence x_i^S.

The model parameter are initialized to the zero vector (line 6). The main loop iterates until the model reaches convergence (lines 7-19). After which the model parameters are returned.

The first inner loop iterates over D^L (lines 8-13) executing the standard on-line training. The candidate parse, $\hat{y}$, for the current input sentence,

x_i^L, is predicted given the current model parameters, θ (line 9). In the structured perceptron setting (Collins and Roark, 2004; Daumé III et al., 2009), we have that:

$$F_\theta(x) = argmax_{y \in \mathcal{Y}} \ \theta \cdot \Phi(y) \qquad (2)$$

Where $\Phi(\cdot)$ is the mapping from a parse tree y to a high dimensional feature space. Then, the algorithm tests if the current prediction is wrong (line 10). In which case the model is updated promoting features that fire in the gold-standard $\Phi(y_i^L)$, and penalizing features that fire in the predicted output, $\Phi(\hat{y})$ (line 11).

The second inner loop iterates over D^S (lines 14-18). First, the candidate parse, $\hat{y}$, is predicted (line 15). Then the sample parse, y^*, is produced by the sample function (line 16). Finally, the parameters are updated promoting the features of y^*. The updates are triggered without testing if the loss is positive, since it is guaranteed that $L^S(\hat{y}, y^*) \geq 0$. Updating in cases where $L^S(\hat{y}, y^*) = 0$ does not harm the model. To optimize the algorithm, updates can be avoided when $\hat{y} = y^*$.

In order to simplify the algorithmic description, we define the algorithm with only one loss function and one sample function, and we formalized it for the specific task we are considering. This definitions can be trivially generalized to integrate multiple loss/sample functions and to be formalized for a generic structured prediction task. This generalization can be achieved following the guidelines of (Hall et al., 2011a). Furthermore, we defined the algorithm such that it first iterates over D^L and then over D^S. In practice, the algorithm can switch between the data sets with a desired frequency by using a scheduling policy as described in (Hall et al., 2011a). For the experiments, we trained on 8 samples of D^L followed by 1 samples of D^S, looping over the training sets.

4 Sample Function

We integrate the Knowledge Base data into the training algorithm using a sampling function. The idea is to correct errors in the candidate parse by using the KB. The sample function corrects only relations among entities described in the KB. Thus, it returns a better or equal parse tree that may still contain errors. This is sufficient to guarantee the constraint on the implicit loss function (equation 1).

The sample function receives as input the candidate dependency parse and the input sentence enriched with KB annotation. Then, it corrects the labels of each arc in the dependency tree connecting two entities. The labels are corrected according to the predictions produced by a classifier. As classifier we use a standard multi-class perceptron (Crammer and Singer, 2003). The classifier is trained in a preprocessing step on a parsed corpus enriched with KB data. The features used by the classifier are:

- Lexical features of the head and modifier.
- Sentence level features: words distance between head and modifier; arc direction (L/R); neighboring words.
- Syntactic features: POS and syntactic label of head and modifier and modifier's left sibling.
- Knowledge Base features: types defined for entities and for their direct relations.

5 Experiments

The primary training corpus is composed of manually annotated sentences with syntactic tress which are converted to dependency format using the Stanford converter v1.6 (de Marneffe et al., 2006). We run experiments using 10k sentences or 70k sentences from this corpus. The test set contains 16k manually syntactically annotated sentences crawled from the web. The test and train sets are from different domains. This setting may degrade the parser accuracy in labelling out-of-domain entities, as we discussed in section 2. Thus, we use web text as secondary training set to be used for the Augmented-loss loss sample training. Web text is available in any quantity, and we do not need to provide gold-standard parses in order to integrate it in the Augmented-loss sample training. The classifier is trained on 10k sentences extracted from news text which has been automatically parsed. We chose to train the classifier on news data as the quality of the automatic parses is much higher than on general web text. We do this despite the fact that we will apply the classifier to a different domain (the web text).

As dependency parser, we use an implementation of the transition-based dependency parsing framework (Nivre, 2008) with the arc-eager transition strategy. The part of Augmented-loss training based on the standard loss function, applies

Training set size	Model	appos F1	conj F1	LAS	UAS
70k sentences	Baseline	54.36	83.72	79.55	83.50
	Augmented-loss	55.64	84.47	79.71	83.71
10k sentences	Baseline	45.13	80.36	75.99	86.02
	Augmented-loss	48.06	81.63	76.16	86.18

Table 1: Accuracy Comparison.

the perceptron algorithm as in (Zhang and Clark, 2008) with a beam size of 16. The baseline is the same model but trained only the primary training corpus without Augmented-loss.

Table 1 reports the results of the accuracy comparison. It reports the metrics for Labeled Attachment Score (LAS) and Unlabeled Attachment Score (UAS) to measure the overall accuracy. The syntactic classes that are affected the most are apposition (appos) and conjunction (conj). On the development set we measured that the percentage of arcs connecting 2 entities that are labeled as conjunction is 36.11%. While those that are labelled as apposition is 25.06%. Each of the other 40 labels cover a small portion of the remaining 38.83%.

Training the models with the full primary training corpus (70k sentences), shows a significant gain for the Augmented-loss model. Apposition F1 gains 1.28, while conjunction gains 0.75. The LAS gain is mainly due to the gain of the two mentioned classes. It is surprising to measure a similar gain also for the unlabeled accuracy. Since the classifier can correct the label of an arc but never change the structure of the parse. This implies that just by penalizing a labeling action, the model learns to construct better parse structures.

Training the model with 10k sentences shows a significantly bigger gain on all the measures. This results shows that, in cases where the set of labeled data is small, this approach can be applied to integrate in unlimited amount of unlabeled data to boost the learning.

6 Related Work

As we mentioned, Augmented-loss (Hall et al., 2011a; Hall et al., 2011b) is perhaps the closest to our framework. Another difference with its original formalization is that it was primarily aimed to cases where the additional weak signal is precisely what we wish to optimize. Such as cases where we wish to optimize parsing to be used as an input to a downstream natural language processing tasks and the accuracies to be optimized are those of the downstream task and not directly the parsing accuracy. While our work is focused on integrating additional data in a semi-supervised fashion with the aim of improving the primary task's accuracy and/or adapt it to a different domain.

Another similar idea is (Chang et al., 2007) which presents a constraint driven learning. In this study, they integrate a weak signal into the training framework with the aim to improve the structured prediction models on the intrinsic evaluation metrics.

7 Conclusion

We extended the Augmented-loss framework defining a method for integrating new types of signals that require neither gold standard data nor an explicit loss function. At the same time, they allow the integration of additional information that can inform training to learn for specific types of phenomena.

This framework allows us to effectively integrate large scale KB in the training of structured prediction tasks. This approach integrates the data at training time without affecting the prediction time.

Experiments on syntactic parsing show that a significant gain for categories that model relation between entities defined in the KB.

References

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL '07: Proceedings of the 45th Conference of the Association for Computational Linguistics*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL '04: Proceedings of the 42rd Conference of the Association for Computational Linguistics*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Submitted to Machine Learning Journal*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure trees. In *LREC*.

Keith Hall, Ryan McDonald, Jason Katz-brown, and Michael Ringgaard. 2011a. Training dependency parsers by jointly optimizing multiple objectives. In *EMNLP '11: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Keith Hall, Ryan McDonald, and Slav Petrov. 2011b. Training structured prediction models with extrinsic loss functions. In *Domain Adaptation Workshop at NIPS*, October.

Sandra Kubler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. In *Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. volume 34, pages 513–553.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP '08: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571.