

Building an end-to-end text reading system based on a packed representation

Doo Soon Kim

Dept. of Computer Science
University of Texas
Austin, TX, 78712

onue5@cs.utexas.edu

Ken Barker

Dept. of Computer Science
University of Texas
Austin, TX, 78712

kbarker@cs.utexas.edu

Bruce Porter

Dept. of Computer Science
University of Texas
Austin, TX, 78712

porter@cs.utexas.edu

Abstract

We previously proposed a packed graphical representation to succinctly represent a huge number of alternative semantic representations of a given sentence. We also showed that this representation could improve text interpretation accuracy considerably because the system could postpone resolving ambiguity until more evidence accumulates. This paper discusses our plan to build an end-to-end text reading system based on our packed representation.

1 Introduction

Our goal is to build an end-to-end text understanding system by assembling together existing components for parsing, semantic interpretation, co-reference resolution and so on. Commonly, these components are combined in a pipeline in which each one passes forward a *single* best interpretation (see (a) in fig. 1). Although this approach is relatively straightforward, it can suffer from overly aggressive pruning; a component might prune those interpretations that downstream components might have been able to recognize as correct. Similarly, a component might prune an interpretation that would be validated by reading subsequent texts. The system’s accuracy would almost certainly improve if it were able to delay pruning until sufficient evidence accumulates to make a principled commitment.

There is a naïve way of delaying pruning decisions in a pipelined architecture: each component passes forward not just a single interpretation, but

multiple alternatives, thereby creating multiple interpretation paths (see (b) in fig 1). Then, the system might choose the best interpretation at the last step of the pipeline. However, this approach is intractable due to the combinatorial explosion in the number of interpretation paths.

In previous work (Kim et al., 2010), we proposed an alternative approach in which each component passes forward multiple interpretations which are compressed into an intensional representation that we call a *packed graphical (PG) representation* (see (c) in fig. 1). Our experiment showed that the approach could improve the interpretation accuracy considerably by delaying ambiguity resolution while avoiding combinatorial explosion.

In this paper, we discuss our plan to build an end-to-end text understanding system using the PG representation. We first introduce the language interpretation system we are currently building, which produces a PG representation from the parse of each sentence. Then, we propose an architecture for an end-to-end reading system that is based on the PG representation. The architecture allows the system to improve as it acquires knowledge from reading texts.

In the following sections, we briefly describe the PG representation and its disambiguation algorithm (see (Kim et al., 2010) for details). Then, we present the plan and the current status in the development of an end-to-end text understanding system.

2 Packed graphical representation

The PG representation compresses a huge number of alternative interpretations by locally represent-

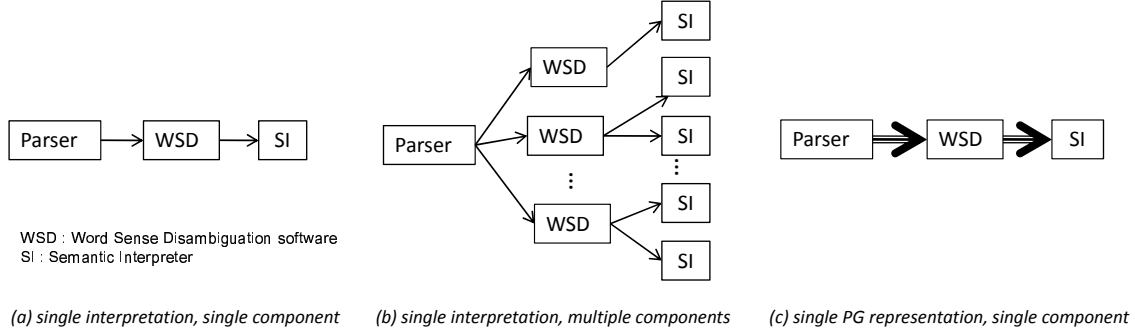


Figure 1: The three different architectures for text understanding system: In (a), each component passes forward a single interpretation. (b) can improve (a) by considering multiple interpretation paths, but suffers from combinatorial explosion. (c) is our approach in which the system considers multiple alternative interpretations (in contrast to (a)) while avoiding combinatorial explosion by packing the alternatives (in contrast to (b)).

ing common types of ambiguities and other types of constraints among the interpretations. Section 2.1 presents these ambiguity and constraint representations. Section 2.2 introduces an algorithm which aims to resolve the ambiguities captured in a PG representation.

2.1 Representation

Fig. 2 shows a PG representation produced from the interpretation of the following sentence:

S1 : The engine ignites the gasoline with its spark plug.

With this example, we will explain the ambiguity representations and the other types of constraints expressed in the PG representation.

Type ambiguity. Ambiguity in the assignment of a type for a word. In PG1, for example, the node engine-2a (corresponding to the word “engine”) has type annotation [LIVING-ENTITY .3 | DEVICE .7]. It means that the two types are candidates for the type of engine-2a and their probabilities are respectively .3 (Living-Entity) and .7 (Device).

Relational ambiguity. Ambiguity in the assignment of semantic relation between nodes. The edge from ignite-3 to engine-2a in PG1 has relation annotation <agent .6 | location .4>. It means that engine-2a is either *agent* (probability .6) or *location* (probability .4) of ignite-3.

Structural ambiguity. It represents structural alternatives in different interpretations. In PG1, for example, D and E represent an ambiguity of prepositional phrase attachment for “with its spark plug”;

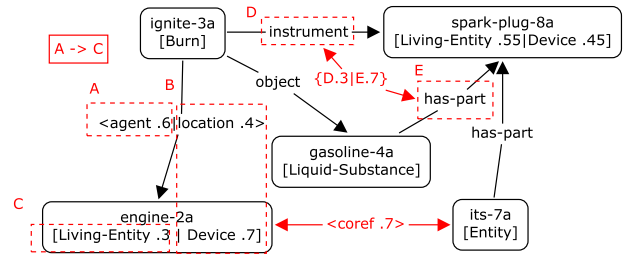


Figure 2: The PG representation for S1 (PG1)

the phrase can be attached to “ignites” (D) or “spark plug” (E). The annotation {D .3 | E .7} means either D or E (not both) is correct and the probability of each choice is respectively .3 and .7.

Co-reference ambiguity. A “co-reference” edge represents a possibility of co-reference between two nodes. For example, the edge labeled <coref .7> represents that the probability of engine-2a and its-7a being co-referent is .7.

Besides ambiguity representations above, the PG representation can also represent dependencies among different interpretations.

Simple dependency. It represents that the existence of one interpretation depends on the existence of another. For example, $A \rightarrow C$ means that if LIVING-ENTITY is found to be a wrong type for engine-2a (by subsequent evidence), the agent relation should be discarded, too.

Mutual dependency. It represents that the interpretations in a mutual dependency set depend on one another – if any interpretation in the set is

found to be wrong or correct (by subsequent evidence), the others should also be rejected or confirmed. For example, the box labeled B means that if either (engine-2a type DEVICE) or (ignite-3a location engine-2a) is confirmed or rejected, the other interpretation should be confirmed or rejected.

Formally, the PG representation can be represented as a list of

- *semantic triples* – e.g., (ignite-3a type BURN), (ignite-3a instrument spark-plug-9a)
- *macros* – e.g., the symbol A refers to (ignite-3a agent engine-2a)
- *constraints* – e.g., A depends on C, D (.3) is exclusive to E (.7)

2.2 Disambiguating ambiguities in a PG representations

In this section, we briefly explain how our disambiguating algorithm resolves ambiguities in a PG representation. For details, please see (Kim et al., 2010).

The PG representation allows the system to delay commitment to an interpretation (by explicitly representing ambiguities) until enough evidence accrues to disambiguate. One source of such evidence is the other texts with redundant content. For a sentence which is hard to interpret, there may be other texts which describe the same content, but in ways that the system can better interpret. These new reliable interpretations can be used to disambiguate the original unreliable interpretations. Our algorithm is based on this approach of combining multiple PG representations to resolve their ambiguities.

The disambiguation algorithm uses graph matching. The algorithm aligns two PG representations to identify their redundant subgraphs (redundant portions of the interpretations), then increases the confidence scores of these subgraphs because the same interpretation was derived from two independent sentences (on the same topic). When the confidence scores reach a high or low threshold, the associated interpretations are confirmed or pruned. Confirming or pruning one interpretation may lead to confirming or pruning others. For example, the dependents of a pruned interpretation should also be pruned.

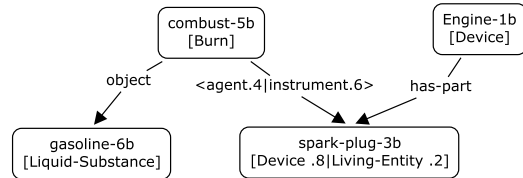


Figure 3: The PG representation for S2 (PG2), “The engine’s spark plug combusts gasoline”

To illustrate the algorithm, we will show how PG1 (fig. 2) is merged with PG2 (fig. 3) to resolve their ambiguities.

1. engine-2 in PG1 is aligned with engine-1 in PG2. This operation chooses Device as the type of engine-2 (i.e., it discards Living-Entity) because Device is favored in both nodes
2. Deleting LIVING-ENTITY causes deletion of the *agent* edge between ignite-3a and engine-2a due to the dependency constraint $A \rightarrow C$, (meaning *agent* (in A) depends on the existence of LIVING-ENTITY (in C)).
3. Co-reference between engine-2a and its-7a is greedily confirmed because merging the two nodes enables the alignment of (its-7a has-part spark-plug-8a) with (Engine-1b has-part spark-plug-3b).
4. The algorithm aligns (ignite-3a instrument spark-plug-8a) with (combust-5b instrument spark-plug-3b), because ignite-3a and combust-5b share the same type, [BURN]. This operation increases the score of D (the structure corresponding to PP attachment of “with its spark plug” to “ignite”) over E (the structure corresponding to attachment of “with its spark plug” to “gasoline”).

3 Taking advantage of the PG representation in an end-to-end system

Our experiment showed that, for ten texts with redundant content, our approach improved the interpretation accuracy by 10% (Kim et al., 2010). Encouraged by this result, we present our on-going work and future plans.

3.1 Producing PG representation

We are currently constructing a fully automated language interpretation system to produce PG representations from English sentences. The system will be able to maintain all possible interpretations generated at each step (including parsing, word sense disambiguation (WSD) and semantic relation assignment) and represent them using the PG representation. This is straightforward for WSD and semantic relation assignment because most off-the-shelf software (e.g., (Patwardhan et al., 2005) (Punyakanok et al., 2005)) outputs a list of candidate choices and confidence scores for type and relational ambiguities. (Kim et al., 2010) describes a prototype system implemented with these WSD and semantic assignment components.

However, ambiguities in parsing are more difficult because it is hard to efficiently identify structural differences among various parses. We are currently developing an algorithm (similar to (Schiehlen, 1996)) which converts a parse forest (the ambiguity-preserving chart built during PCFG parsing) (Tomita, 1986) into the syntactic-level PG representation (as shown in fig. 4). We plan to implement this algorithm in the Stanford Parser (Klein and Manning, 2003) and to evaluate it along the following dimensions.

First, we will measure the improvement in parsing accuracy that results from delaying commitment to a single best parse.

Second, even though the PG representation achieves substantial compression, its size is still bounded. The parser might generate more interpretations than will fit within the bound. We plan to handle this problem in the following way. When a PG representation grows to the bound, the system applies the components downstream of the parser to the candidate parses. Because these components use additional sources of knowledge, including knowledge derived from previous reading (Clark and Harrison, 2009), they might be able to prune some candidate interpretations. In this way, a part of a sentence may be processed early while the other parts are left unprocessed, in contrast with the traditional approach of fully processing each sentence before starting with the next.

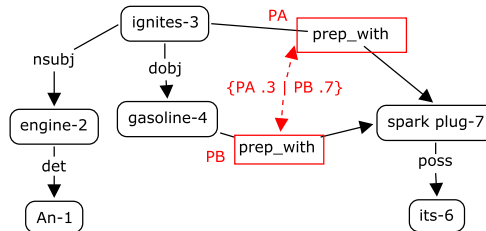


Figure 4: Syntactic-level PG representation for S1: the structural ambiguity represents an ambiguity of attaching the preposition, “with its spark plug”.

3.2 System Architecture

The PG representation and its disambiguating algorithm allow an interesting property in a text understanding system: the system’s interpretation capability could increase as it acquires knowledge from texts. This property can be shown in two ways. First, the ambiguities of the current text could be resolved later when the system reads subsequent texts. Second, the knowledge acquired from the prior texts could be used to resolve the ambiguities of the current text. Fig. 5 shows an architecture that exhibits this property.

Given a text, or a set of texts on the same topic, the language interpreter generates a PG representation. Then, the knowledge integration component (KI) adds the PG representation into the knowledge base. For a first text, the PG representation is simply put into the knowledge base. For subsequent texts, KI merges the subsequent PG representations with the PG representation in the knowledge base. This step may resolve ambiguities in the PG representation maintained in the knowledge base.

When the language interpreter confronts an ambiguity, it has two choices: it either (a) locally represents the ambiguity in the PG representation or (b) asks the RESOLVER to resolve the ambiguity. When the RESOLVER is called, it searches the knowledge base for information to resolve the ambiguity. If this is unsuccessful, it uses the information retrieval module (TEXT MINER) to find relevant documents from external sources which might resolve the ambiguity. The documents are added in the Text Queue to be read subsequently. In the near future, we plan to evaluate the ability of the KI and Resolver modules to resolve ambiguities as the system reads more texts.

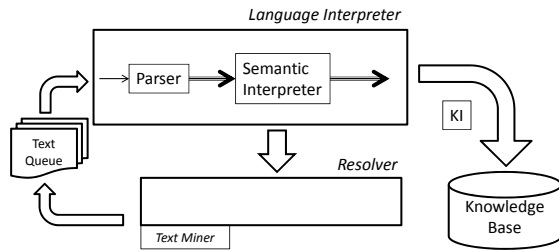


Figure 5: Architecture

4 Summary

In this paper, we discuss the development of an end-to-end text understanding system based on a packed representation. With this representation, the system can delay ambiguity resolution while avoiding combinatorial explosion, thereby effectively improving the accuracy of text interpretation.

References

- Peter Clark and Philip Harrison. 2009. Large-scale extraction and use of knowledge from text. In Yolanda Gil and Natasha Frdman Noy, editors, *K-CAP*, pages 153–160. ACM.
- Doo Soon Kim, Ken Barker, and Bruce Porter. 2010. Improving the quality of text understanding by delaying ambiguity resolution. Technical Report TR-10-12, University of Texas at Austin.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.
- Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2005. Senserelate:: Targetword-A generalized framework for word sense disambiguation. In *ACL*.
- Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*.
- Michael Schiehlen. 1996. Semantic construction from parse forests. In *COLING*, pages 907–912.
- Masaru Tomita. 1986. *Efficient Parsing for Natural Language — A Fast Algorithm for Practical Systems*. Int. Series in Engineering and Computer Science. Kluwer, Hingham, MA.