Getting More Out Of Syntax with PropS

Abstract

We propose PropS, a new representation scheme that focuses on capturing the proposition structure expressed in syntax. Compared to the commonly used dependency parses, our scheme represents propositions of various types more succinctly and uniformly, making them accessible via simple graph traversal. We provide a gold standard corpus for future parser development, as well as a simple baseline parser, and suggest PropS utility for downstream semantic processing.

1 Introduction

Propositions constitute the primary unit of information conveyed in texts. Accordingly, recognizing and matching proposition structure is of great importance for semantic tasks.

Perhaps the most common practice for identifying proposition structure in systems and algorithms is through dependency trees. These are attractive with respect to proposition structure as they directly connect verbal predicates to their arguments, which constitute the 'backbone' for large fraction of the expressed propositions. Some *deep syntax* extensions of dependency trees mark also long distance dependencies, further broadening predicate-argument coverage (De Marneffe and Manning, 2008; Miyao et al., 2008; Oepen et al., 2014; Ballesteros et al., 2014). Despite the attractive properties of (deep) dependency parses, it is quite hard to read out from them the *complete* structure of *all* propositions expressed in a sentence. Different predications are represented in a non-uniform manner (e.g, verbal versus adjectival predication), proposition boundaries are not easy to detect, while substantial parts of the dependency structure represent syntactic detail that is not core to proposition structure. For these reasons, many NLP systems tailor sets of rules and heuristics to unify and extract specific information from the parsed text.

In this work, we propose PropS, a graph-based representation scheme that focuses on capturing the *Proposition Structure* expressed by the syntax of a sentence. Our scheme both unifies and simplifies proposition representation, making it conveniently accessible for downstream semantic processing. It uniformly represents propositions headed by different types of predicates, verbal or not, including those expressed by non-lexicalized syntactic constructs. We explicitly recover predicate-argument and modification relationships, saving the need for subsequent propagations. Further, the representation canonicalizes different syntactic constructions that correspond to the same proposition structure, and decouples independent propositions while explicating proposition boundaries. Finally, the scheme masks non-core syntactic detail, yielding cleaner compact structures. Overall, it enables simple access to all sentence propositions by a uniform graph traversal, offering an appealing alternative to common uses of dependency structures. The representation scheme was designed with semantic applications in mind, aiming to generalize many of the processing steps that are usually applied to syntax trees.

In terms of scope, we chose to focus on English, and target the complete proposition structure expressed by the *syntactic level*, while avoiding deeper semantic analysis and interpretation, such as that provided by SRL. We suggest that by targeting phenomena which are close to the syntactic level we could leverage the relatively high accuracy and robustness levels currently attainable by syntactic parsing algorithms, while still providing a significant improvement over the bare syntactic structure for downstream semantic processing.

After specifying the components of the PropS formalism, the bulk of the paper is dedicated to describing in some detail how the proposition structure in various English syntactic constructions can be mapped effectively onto this scheme. Staying close to the syntactic level allows us to leverage on existing resources (e.g, PTB

(Marcus et al., 1993) or Propbank (Kingsbury and Palmer, 2003)), which we use in Section 4 to automatically annotate a large corpus with PROPS structures. We compare the automatically derived structures to a sample of 100 manually verified PROPS graphs, and find that the automatic conversion attains high accuracy. In addition, we provide a baseline parser, which is a stripped down version of our converter that works on top of automatically produced parse-trees, and compensates for the missing levels of annotations available in the gold trees using a set of heuristics. Finally, in Section 5 we illustrate the utility of the automatically derived PROPS structures on two semantic NLP tasks and show that the structures produced by the baseline parser, while far from being perfect, already provide accuracy gains above the direct use of dependency trees.

All of our code and annotated data will be made available at ANNONYMIZED_URL.[1]

2 Representation Scheme

We present the PROPS formalism, providing succinct representation for rich proposition structures. While formally similar to dependency structure, it allows much easier uniform traversal of propositions, over just a couple node and edge types.

Sentences are represented as labeled directed graphs. Nodes in the graph represent atomic elements participating in a proposition, while labeled directed edges encode relations between these elements. We have three kinds of nodes: (1) predicate nodes, which introduce propositions (either verbal or non-verbal); (2) non-predicate elements, which participate in a proposition without introducing a new one (noun phrases, adverbs and restrictive modifiers); and (3) conjunctions, which we treat as an exception from the other types. All propositions are rooted by predicate nodes, and every predicate node introduces a proposition.

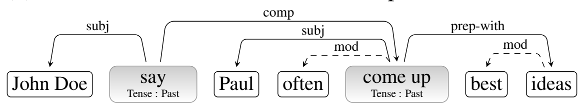
Our edge-labels inventory consists of 14 labels[2] (compared with approximately 50 in dependency trees), all of which are discussed in detail in Section 3.

As a first example, see the graph in example 1.

[1]We include a sample in the supplementary material.

[2](1) syntactic relations between predicates and their arguments: subj, dobj, iobj, comp, prep, and time; (2) a small PROPS specific label set for predicate-argument relations: prop_of, SameAs_arg, condition and outcome; (3) relations between heads and their modifiers: mod, source and poss; and (4) conjunctions, represented using the conj label.

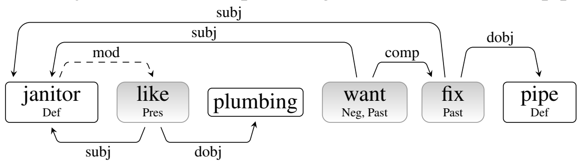
(1)John Doe said that Paul often came up with the best ideas



Predicate nodes are shaded, and modifier edges are dashed. This example also demonstrates additional aspects of the representation: not all words from the sentence appear in nodes and a node may span more than one word. In some cases (not shown in the example), nodes do not correspond to specific words in the sentence. This is elaborated on in section 3.1. In addition to words, nodes may also contain features in a flat key=value structure. Features are used to encode syntactic properties of modality, negation, definiteness, tense, passive or active voice, and other phenomena, abstracting over the ways in which these features are realized in the surface forms.

Similar to the non-tree version of Stanford Dependencies and other deep syntactic representation (De Marneffe and Manning, 2008; Melčuk, 1988; Ballesteros et al., 2014; Oepen et al., 2014), we explicitly mark the arguments of nested predicates and long-distance dependencies in constructions such as relative clauses and control, allowing for nodes with multiple parents:

(2)The janitor who likes plumbing didn't want to fix the pipe



Reading the propositions off of the graph is a simple matter of traversing the shaded nodes, and collecting the nodes reachable from each.

3 Handled Phenomena

The proposition structures in examples (1) and (2) resemble dependency structures. Indeed, syntactic dependency representations do a good job of capturing the propositions that are introduced by verbal predicates, and we follow the deep dependency syntax handling of verbal predicate-argument structures. Yet, we argue that for other types of predication the dependency representation is not uniform and requires further processing by semantic applications. We target our analysis on syntactic phenomena that are frequent (as evident from the Penn Treebank), useful for applications, and can be accurately extracted from existing manual annotations.[3] In this section, we sur-

[3]This analysis yielded the label set listed in Section 2.

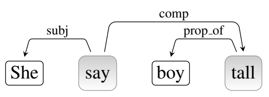
vey interesting cases in which we depart from traditional syntactic representations, in order to provide a unified representation for a wide range of propositions expressed by different syntactic constructions.

3.1 Non-Verbal Predicates

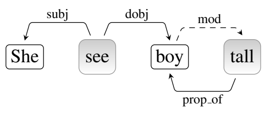
While syntactic analysis is mostly verb centric, some propositions are introduced by non verbal elements. We deal with three of these instances, as follows (the interesting case of nominal predication is left to be integrated in future work).

Adjectival predication Adjectives evoke a proposition (for example sentences such as *the boy is tall* and *the tall boy* evoke the proposition tall(boy)), for which we use the "property_of" label, to denote a 1-ary predication.

(3) She said that the boy is tall



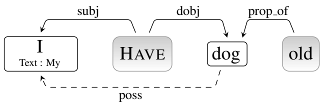
(4) She saw a tall boy



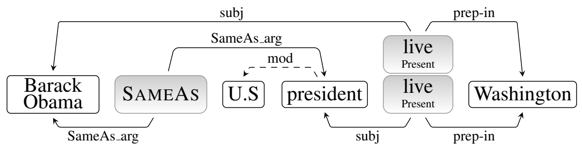
Notice that in (4) we mark *tall* as a modifier of *boy* for the completeness of the see(she,tall boy) proposition, but *tall* is also used as a predicate in a separate tall(boy) proposition.

Non-lexical propositions In some cases, a proposition may not correspond to a specific word in the sentence, but is rather implied from the syntactic structure. This occurs, for example, in possessive constructions: *my dog is old* implies that *I have a dog*, without a word mediating this relation in the sentence. We deal with such cases by introducing a synthetic HAVE node which explicitly represents this predication (example 5). Similarly, we introduce a synthetic SAMEAS node for apposition (and copular) constructions (example 6), explicitly marking equivalence between two entities, and EXISTS node for existentials (example 7).

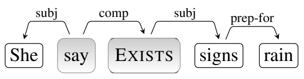
(5) (a) My dog is old
(b) I have a dog, who is old



(6) Barack Obama, the U.S. president, lives in Washington



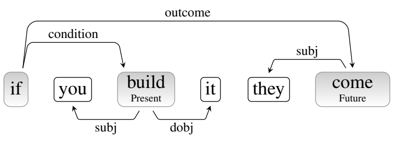
(7) She said that there are signs for rain



Conditionals Conditionals are a syntactic construction in which a word (syntactically referred as "marker") introduces a logical relation between two discrete propositions, where one of the clauses conditions the other. While there has been extensive linguistic study of the subject (Stalnaker, 1981; Bennett, 2003), NLP applications tend to overlook these constructions (a recent exception is (Berant et al., 2014)).

We treat conditional constructions as propositions, as they can be assigned a truth value. We analyze the marker word (e.g, *because*, *although*, *unless*, etc.) as a predicate and as the head of both clauses, using the "condition" and "outcome" labels, as it introduces the logical relation between them (example 8).

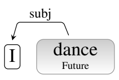
(8) (a) If you build it, they will come
(b) They will come, if you build it



3.2 Verbs that are not predicates

Some verbs do not evoke propositions, but rather serve as modifiers for other propositions. For example, in *I am going to dance*, the verb *going* is used as a future tense indicator. While a syntactic representation will treat *going* similar to other to-infinitive constructions, we express it as a feature on the *dance* predicate:

(9) (a) I am going to dance
(b) I will dance

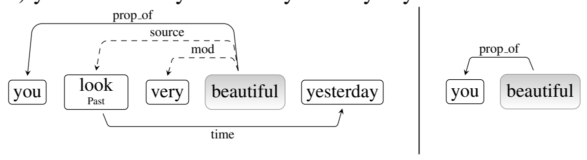


Similarly, verbs in raising to subject (*seems* in *John seems to love Mary*) and adjectival complement constructions (*looked* in *she looked nice*) do not evoke new propositions but rather serve as modal modifiers of other predicates (Huddleston et al., 2002). We treat these by introducing a "source" edge indicating a modification relation.[4] This analysis focuses the attention on the

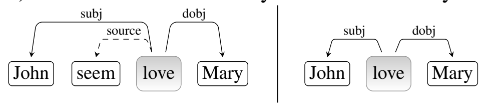
[4]While one may think of modality, and hence "source", as being a feature of a predicate rather than a relation, the relation is needed for cases where the modality itself has a rich internal structure, as seen in example (10).

main predicate and maintains a uniform predication structure, as demonstrated by the structural resemblance to *You are beautiful* and *John loves Mary*.

(10) you looked very beautiful yesterday / you are beautiful



(11) John seems to love Mary / John loves Mary



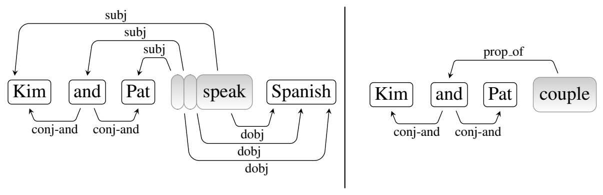
3.3 Argument Propagation

In some cases, most notably in coordination constructions, predicate-argument relations can be propagated according to the syntactic structure, resulting in new propositions. For example in *Dell makes and distributes products* the propagation results in the three propositions make(Dell, products), distribute(Dell, products) and make_and_distribute(Dell, products). Similarly, for *Dell sells laptops and servers* we get sell(Dell, laptos), sell(Dell, servers) and sell(Dell, laptops and servers).

Our representation of coordination follows the Prague-Treebank style (Popel et al., 2013; Böhmová et al., 2003) and posits the coordinating word as the main node of the coordinating-conjunction structure, while the different conjuncts are attached to it using "conj" edges. The semantic of coordination nodes is that the node represents the entire conjunction. Following relation propagation, relations may involve the coordination node (i.e. the entire coordinated structure) and/or the individual conjuncts. A similar approach is taken by the "propagated" variant of Stanford Dependencies (De Marneffe and Manning, 2008), but we differ in choosing the coordination as the main element in the conjunction, and take more care in distinguishing different kinds of coordination constructions and determining the conjunction scope. As one example, we distinguish between distributive and joint coordination (Huddleston et al., 2002). In distributive coordination (*Kim and Pat speak Spanish*) we fully propagate the relations, while for cases of joint coordination (*Kim and Pat are a couple*) we do not propagate, as the relation is meant only for the combination of both entities.

When the relation propagation results in multiple propositions involving the same predicate, the predicate node is duplicated in order to distinguish the different propositions and keep with the 1-1 correspondence between propositions and predicate nodes.

(12) Kim and Pat speak Spanish / Kim and Pat are a couple



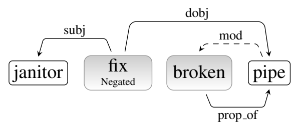
We also propagate relations in appositive and copular constructions involving the SAMEAS node (example 6), e.g. *Amazon, the retail giant, sells products* will evoke both sell(Amazon, products) and sell(retail giant, products).

3.4 Canonicalization and Differentiations

Different syntactic realizations may evoke the same proposition structure (e.g., passive versus active voice), while sentences with similarly looking syntactic structures evoke different proposition structures.We aim to present a unified proposition structure when applicable, and different structures when needed. Some of the cases we consider are detailed below.

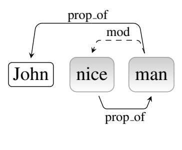
Adjectival modification We unify restrictive adjectival modifications that are expressed as prenominal adjectives (*broken pipe*) or as relative clauses (*pipe which was broken*), representing both using the "mod" relation.

(13) (a) The janitor didn't fix the broken pipe
(b) The janitor did not fix the pipe which was broken



Adjectival predication We provide a unified representation for adjectival predications, which are represented using a "property_of" label connecting a predicate node to its argument. This structure is evoked by adjectives (*John is nice*) and adjectival phrases (*John is a nice man*) in the non-restrictive cases of prenominal adjectives, relative clauses, appositive and copular constructions.

(14) (a) John, who is a nice man
(b) John is a nice man
(c) John, a nice man



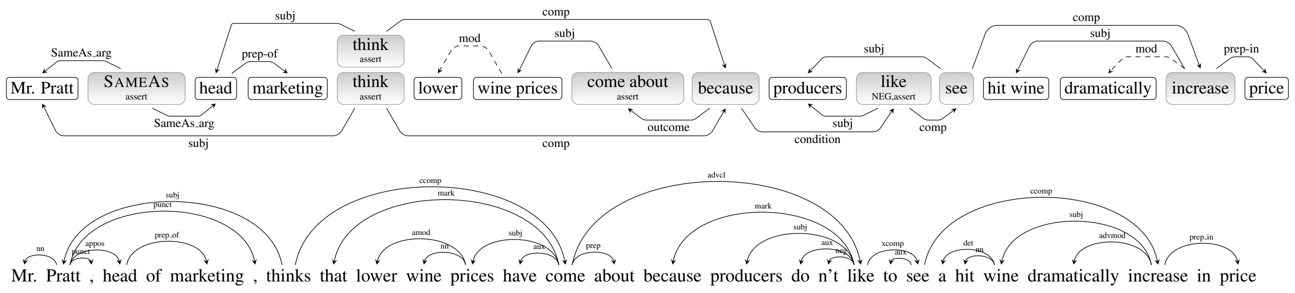


Figure 1: Our representation (top) versus Stanford-dependencies representation (bottom) of the sentence “Mr. Pratt, head of marketing, thinks that lower wine prices have come about because producers don’t like to see a hit wine dramatically increase in price”. See analysis in section 3.6

Copulas and appositions Syntactic copular (*X is Y*) and appositive (*X, Y*) constructions may evoke either an equivalence IsA relation (*Obama, the president*; *Obama is the president*) or group membership relation (*Obama, an american citizen*). We represent the equivalence case using the SAMEAS predication node (example (6) above) and the group-membership case as an adjectival predication (example (14)). The different cases are distinguished based on the syntactic categories and definiteness status of both X and Y.

Similarly, in relative clauses we distinguish between restrictive (*the pipe that was leaking*) and non-restrictive (*John, who is talented*) modifications, as described in the next section.

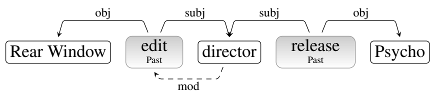
3.5 Proposition Boundaries and Assertions

Marked proposition boundaries While syntactic representations contains much of the proposition structure, they do not clearly mark the boundaries of different propositions and their arguments, making it hard to focus on specific propositions.

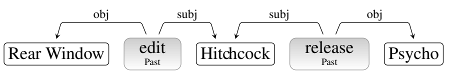
Consider, for instance, the following sentences:
-*The director who edited ‘Rear Window’ released Psycho*
-*Hitchcock, who edited ‘Rear Window’, released Psycho*.
While syntactic analysis will assign similar structure for both instances, each induces different proposition boundaries. In the first sentence, the director who edited Rear-Window is a single argument, yielding the proposition released(the director who edited Rear Window, Psycho). However, in the second sentence we can separate the relative clause from the entity it modifies, yielding two distinct propositions: edited(Hitchcock, Rear Window) and released(Hitchcock, Psycho).

The difference between the two cases is that the first exhibits a *restrictive* while the second exhibits a *non-restrictive* modification (Kamp and Reyle, 1993). Restrictive modifiers are used to select an item from a set and are an integral part of the proposition. We represent these with a “mod” edge (example 15). Non-restrictive modification, on the other hand, “breaks” the propositions and provides additional information on a selected entity, therefore forming a separate predication (example 16).

(15) the director who edited ‘Rear Window’ released Psycho



(16) Hitchcock, who edited ‘Rear Window’, released Psycho



Syntactic assertedness Certain propositions are asserted by the sentence, while others are attributed (compare *John passed the test* versus *the teacher said that John passed the test*). This property was extensibility studied in the PDTB (Prasad et al., 2006) and PARC (Pareti, 2012) corpora, and was shown to be useful in semantic applications (e.g., (Aramaki et al., 2009)). We mark cases where it is possible to syntactically determine if a proposition is asserted. While hierarchical structure generally implies dependence of the nested proposition, in certain constructions the nested proposition is in fact independent of its subordinating clause. This is the case for non restrictive relative clause (*Alfred Hitchcock, who directed Psycho* in which the directed proposition is syntactically asserted), certain conditionals (*Glaciers are melting because temperature rises* in which both rise and melt predications are asserted), and several other constructions.

3.6 Summary

Figure 1 compares PROPS with Stanford-dependency analysis for a typical WSJ sentence. The dependency representation contains 27 nodes (and therefore 27 edges), while our representation contains 17 nodes and 19 edges. A simple traversal from all predicate nodes in our graph yields the following propositions:

(1) lower wine prices have come about	[asserted]
(2) hit wine dramatically increase in price	
(3) producers see (2)	
(4) producers don't like (3)	[asserted]
(5) Mr Pratt is the head of marketing	[asserted]
(6) (1) happens because of (4)	
(7) Mr Pratt thinks that (6)	[asserted]
(8) the head of marketing thinks that (6)	[asserted]

Extracting this set of propositions and their assertion status from the dependency representation will require a non-trivial amount of processing.

4 Annotated Corpus and Parser

We provide two annotated PropS resources: a small manually annotated gold-standard corpus, and a large high-accuracy corpus based on automatic conversion of the WSJ section of the PTB, utilizing existing manual annotations to achieve high accuracy. In addition, we provide a rule-based baseline parser, which works by automatic conversion of the output of an underlying dependency parser, without relying on any external annotations.

Gold Standard We manually annotated a batch of 100 sentences of the WSJ, composed of the first 50 consecutive sentences of sections 2 (train) and 22 (dev). The annotation was performed by manually verifying and fixing the output of the automatic conversion described below.

Automatic Annotation of the WSJ We created a version of the WSJ section of the Penn Tree Bank (PTB) (Marcus et al., 1993) (51,129 sentences) annotated according to the proposed representation. The annotation was done by automatic conversion, taking into account the rich syntactic information available in the gold trees (Bies et al., 1995) (including traces, empty elements and functional annotations, which provided input for approximately 36K edge attachments) enriched with Vadas and Curran's (2007) NP structure annotation (which provided roughly 26K structures), as well as information from PropBank (Kingsbury and Palmer, 2003), which was used to accurately recognize the proposition structure of verbal predicates, distinguish raising and control constructions and provide tense information.

As the annotated corpus is based on automatic conversion, it may include some inaccuracies, especially in difficult cases that are not properly attested in the underlying annotations, such as the identification of apposition constructions and coordination within base-NPs. We asses its accuracy by comparing against the manually annotated gold standard. By adapting the metrics used in (Ballesteros et al., 2014), which we extend to deal with PropS 's non 1-1 mapping between words and graph-nodes, we find that the conversion accuracy performs with 91% $F1$ score of for labeled attachment, and 96% $F1$ score for feature computation.[5] Manual inspection of the errors reveals that the vast majority ($> 80\%$) stem either from errors in the underlying annotations, or disagreement about annotation standards (e.g. PropBank marks "business" in "the business grew" as an object, while we treat it as a subject).

This high accuracy of the automatic conversion, which stems from the use of the manual annotations, warrants its use as a large reference corpus for the development and evaluation of future parsers for PropS .

Baseline Parser In addition to the annotated corpus, we also provide a baseline parser that can be used to assign an approximation of our proposed representation to new sentences. The parser relies on the high accuracy of dependency parsers, first parsing the sentence into Stanford Dependencies and then deterministically converting the resulting dependency-tree into the PropS representation. The conversion component is similar to the one used for deriving the annotated corpus, but lacks access to information sources such as traces and PropBank annotations. Instead, it relies on a set of heuristics for dealing with hard decisions such as correctly propagating properties in conjunctions or distinguishing between raising and control. In addition, it does not attempt to provide inner-structure for base NPs.

When converting from dependency structures derived by running the Stanford converter on top of the Berkeley parser (Petrov and Klein, 2007), and using the previously mentioned metrics we find that the baseline parser performs with $82\% F1$ score for labeled attachment and $89\% F1$ score for feature computation. While the baseline parser is only capable of producing an approximation of the complete representation and is bounded by the

[5]Full details of the evaluation metric are available in the supplementary material. We note that the evaluation metric is very strict – an error in node span will incur errors in all features and edges involving that node. For example, failing to combine *New England Journal of Medicine* into a single node in "results appear in today's New England Journal of Medicine" is penalized with 6 edge errors.

accuracy of the underlying dependency parser, it does succeed in recovering the major parts of the structure with sufficient accuracy, and provides a way of deriving PropS representations for arbitrary texts. In the next section, we demonstrate that the automatically derived PropS structures can already be used as an appealing alternative to dependency trees in two downstream applications. Motivated by the success of these preliminary experiments, we believe that an improved parser which generates PropS representations directly would be of even greater benefit to semantic applications.

	Rules		Learning	
	Dep	PropS	Dep	PropS
Gold	.662	**.765**	**.826**	**.827**
Predicted	.627	**.648**	.727	**.736**

Table 1: $F1$ scores on binary relation identification task. Separated between unsupervised (left) and supervised (right).

Method	Correct
PropS	**66.34%**
dependencies	64.58%
lexical	60.44%

Table 2: Results on MCTest corpus

5 Empirical Demonstration of Utility

We demonstrate the potential utility of the proposed representation to semantic applications on two tasks: relation extraction and text comprehension. The two tasks require sentence level semantic analysis, commonly take dependency trees as input, and can be attacked using algorithms that exhibit a separation between the feature extraction (that rely on the input representation) and subsequent computation (that do not). We replace the input from dependency trees to PropS structures and make the minimal necessary adjustments to the feature extraction.

Relation Extraction The goal of relation extraction is to identify entities in a sentence that share a relation. This can be formulated as a binary classification task in which we are given a sentence and two entities and need to decide if there is a relation between them. A corpus for this binary classification task was created by (Xu et al., 2013). The corpus contains human-provided judgements for 2629 co-occurring entities from 956 WSJ sentences (split into 756-100-100 for train, dev and test). As this annotation was carried over the WSJ, it allows for evaluating the systems' performance both in a gold-standard setting as well as on automatically derived structures. Xu et al. tackle the binary identification task using two methods: rule-based and learning based, both take as input Stanford-dependency trees.

The rule-based method specifies a list of dependency-paths that indicate relations of interest. When using PropS as input, we use instead a simple distance-based rule and mark entities as related iff the minimal graph-distance between them is 3 edges or less (number tuned on the dev set).

Table 1(left) compares the performance of the rule-based methods when using either gold or predicted structures. In both cases, using PropS with the simple distance-based rule achieves the best F score. Further analysis reveals that the PropS based method significantly improves recall over Xu et als rules (83% vs. 55%), while hurting precision (70% vs 83%). This is expected, as PropS was designed for extracting all propositions expressed in the text, while the notion of relations used by Xu et al is more restrictive. Further task-specific tuning of the rules is likely to improve our position on the precision / recall curve.

For the learning-based method, Xu et. al use SVM with Partial Tree Kernels (PTK) (Moschitti, 2006). The tree kernels alleviate the need for manual feature design by automatically considering the entire space of possible subtrees connecting the entities. The input to PTK are trees which encode the minimal labeled path between the entities heads in the dependency tree. When using PropS we take the shortest path in the graph, including the edge labels, but not encode the node features. When using the tree kernels, the results (Table 1(right)) of both representations are similar, with a slight and insignificant edge to PropS . For this particular task, PTK managed to learn the relevant dependency patterns based on the supervised training data, bridging the difference between the representations. However, our representation still performs as well as dependency trees.[6]

Reading Comprehension To further evaluate our automatic converter we use the MCTest corpus for machine comprehension (Richardson et al., 2013), composed of 500 short stories, each followed by 4 multiple choice questions. The MCTest comprehension task does not require ex-

[6]Results may improve by not focusing on shortest-path trees and using instead larger graph structures and incorporating also the node's features.

tensive world knowledge. This makes it ideal for testing underlying representations, as performance will mostly depend on accuracy and informativeness of the representation. We focus our tests on questions which are marked in the corpus as answerable from a single sentence in the story (905 questions followed by 3620 candidate answers).

Richardson et al. (2013) introduce a lexical matching algorithm, which we adapt to use either dependency or PROPS structures. The algorithm converts a question and a possible answer into a candidate answer (CA) (e.g., *Billy is the name of the boy* is obtained from the question/answer pair "what is the name of the boy?"/"Billy"), counts lexical matches between CA and a sliding window over the story, and returns the answer with the maximal number of matches.

Our adaptation works by counting matches of *representation-units* instead of words: The CA is compared against each story sentence (instead of a sliding window), and instead of counting lexical matches counts the number of identical representation-units in the CA and the sentence.[7] Table 2 shows the adaptation is effective: using dependencies improve over the lexical baseline reported in the MCTest paper, and using PROPS further improves results over dependency trees.

PROPS outperforms dependencies in answering questions involving phenomena addressed by our framework (e.g., in appositives: *John and his best friend, Rick, shared their love for peaches.* to answer: "What did John and Rick both love?").

6 Related Work

Aside of deep dependency representations, UCCA(Abend and Rappoport, 2013) is perhaps the most similar effort to ours. They proposed a new representation scheme geared towards semantic applications and a small accompanying gold standard corpus. However, UCCA differs in their main motivations. Namely they aim to be a cognitive interlingua representation. The design choices of UCCA seem to focus on descriptiveness and universality rather than immediate use, resulting in a cognitive-motivated label-set that, for example, does not distinguish between subject and object giving both a "participant" status. Finally, to the best of our knowledge there is no automated system capable of inferring UCCA structures from text, while we provide a working baseline implementation and demonstrate its effectiveness.

Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) is also centered around proposition structures, with the goal of identifying predicates and linking each predicate to its semantic arguments (agent, patient, etc.) and adjuncts (locative, temporal, etc.). While SRL does not cover the full scope of propositions and unifications handled by PROPS, for the propositions it does cover it tackles a challenging semantic task that PROPS does not attempt to address, namely assigning arguments with *semantic* roles and grounding predicates to a semantic lexicon. PROPS graphs may be extended with SRL annotations, which can be incorporated on top of PROPS structures by adding semantic-role labels to predicate-argument relations.

Other notable semantic formalisms include GMB (Basile et al., 2012), UNL (Uchida and Zhu, 2001) and recently AMR (Banarescu et al., 2013). While AMR's rooted tree can not capture independent propositions within a single sentence, it otherwise subsumes both PROPS and SRL and attempts to provide the complete semantic structure of the sentence (along with inner structure and inter-relations). Producing accurate AMR structures requires in-depth semantic analysis, and it is currently produced with relatively low accuracy (Flanigan et al., 2014). In contrast, our representation is designed to be relatively easy to induce. Generally, PROPS can be viewed as a step between syntax and more semantic representations, providing a strong foundation on top of which more elaborate semantic annotations can be added.

7 Conclusion

We presented PROPS – a sentence representation framework in which all predications are represented in a uniform manner. PROPS allows semantic applications to explore the entire proposition structure, oblivious to the way it was expressed in the surface form. In addition to annotation guidelines we also produce two annotated resources (a small gold standard corpus and an automatic high accuracy conversion) and a baseline parser, illustrated to be of potential benefit to semantic applications. In conclusion, we suggest that PROPS may provide a practical intermediate representation level between syntactic parsing and deeper, but harder, semantic representations.

[7] Both PROPS and Stanford-dependency trees were obtained by using Berkeley Parser (Petrov and Klein, 2007) as the base parser. In both cases the representation-units are (source-word(s), label, target-word(s)) triplets.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL (1)*, pages 228–238.

Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Mashuichi, and Kazuhiko Ohe. 2009. Text2table: medical text summarization system based on named entity recognition and modality identification. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 185–192. Association for Computational Linguistics.

Miguel Ballesteros, Bernd Bohnet, Simon Mille, and Leo Wanner. 2014. Deepsyntactic parsing. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *LREC*, volume 12, pages 3196–3200.

Jonathan Bennett. 2003. A philosophical guide to conditionals.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

J. Flanigan, S. Thomson, J. Carbonell, C. Dyer, and N. A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*, Baltimore, Maryland, June. Association for Computational Linguistics.

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, pages 1–23.

Hans Kamp and Uwe Reyle. 1993. *From discourse to logic: Introduction to model theoretic semantics of natural language, formal logic and discourse representation theory*. Number 42. Springer Science & Business Media.

Paul Kingsbury and Martha Palmer. 2003. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, volume 3.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Igor Aleksandrovič Melčuk. 1988. *Dependency syntax: theory and practice*. SUNY Press.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proc. of ACL*, volume 8, pages 46–54.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *17th European conference on Machine Learning, ECML06*, page 318329.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Silvia Pareti. 2012. A database of attribution relations. In *LREC*, pages 3213–3217. Citeseer.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, volume 7, pages 404–411.

Martin Popel, David Marecek, Jan Stepánek, Daniel Zeman, and Zdenek Zabokrtskỳ. 2013. Coordination structures in dependency treebanks. In *Proc. of ACL*, pages 517–527.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Aravind Joshi, and Bonnie Webber. 2006. Annotating attribution in the penn discourse treebank. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 31–38. Association for Computational Linguistics.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

Robert C Stalnaker. 1981. A theory of conditionals. In *Ifs*, pages 41–55. Springer.

Hiroshi Uchida and Meiying Zhu. 2001. The universal networking language beyond machine translation. In *International Symposium on Language in Cyberspace, Seoul*, pages 26–27.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 240.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877, Atlanta, Georgia, June. Association for Computational Linguistics.