Jay Kaiser and Mike Czerniakowski
Dickinson LING-L545/445
3 May 2016

Using Nivre's Arc-Eager Algorithm toward an Extensible Dependency Parser

Introduction

Dependency grammar is an ever-growing alternative to traditional Chomskyan grammar formalisms. Focusing not on constituents but instead on the dependents and governors in a sentence, dependency grammar provides a strong means for sentential parsing, and much research in the natural language processing field has already worked toward more accurate and quicker parsing in the dependency formalism (Nivre and Fernandez-Gonzalez 2014, Covington 2001, Yamada and Matsumoto 2003). One work in particular, the projective dependency work done by Nivre (Nivre and Fernandez-Gonzalez 2014) proves especially promising in parsing, due to its simplicity, high accuracy, and rapidness of parsing; therefore, it is the work of Nivre that we choose to utilize here. However, these methods tend to utilize machine learning technology in order to describe dependencies between elements of the sentence, as opposed to some form of classifier that takes into account grammatical elements of the language they are parsing.

Therefore, our research expands on traditional methods of dependency parsing, and instead of using shallow machine learning techniques, we have chosen to incorporate elements from extensible dependency grammar (XDG) as a means to naturally decide connections between words based on grammatical information of the language. However, because attribute value matrices are used in XDG to create the lexicon of words that can be parsed, and because it is impossible to adequately and efficiently represent the lexicon of a language through the manual creation of every possible word form that can be found, part-of-speech (POS) tags were used as the basis of the lexicon. Because of this, POS tags are the real source of information used in the parse, not individual words in the sentence; moreover, POS tags must be incorporated into the sentence input in order for a parsed output to be provided.

Related Work

A variety of dependency parsing research has already been done in the field. Covington (2001) developed an incremental deterministic algorithm that parses sentences in $O(n^2)$ time utilizing a set of three different linking operations. Yamada and Matsumoto (2003) developed a shift-reduce type algorithm adapted for English that too parses sentences in $O(n^2)$ time using only shift, left-arc, and right-arc parsing actions.

Nivre has created and edited his shift-reduce type algorithm originally found in (Nivre 2003) for the past decade, incorporating additions to increase both processing accuracy (Goldberg and Nivre 2012; Nivre and Fernandez-Gonzalez 2014) and the parsing of non-projective sentences through an additional *swap* action (Nivre 2009; Nivre, Kuhlmann, & Hall 2009). His algorithm consistently approaches linear runtime, and accuracy only increases after every iteration, making it a perfect candidate for expansion in this work herewith.

Extensible dependency grammar was first outlined in (Debusmann, Duchier, and Kruijff 2004) as a means to utilize dependency grammar in multiple levels of linguistic analyzation. This formalism aims to track the interactions between syntax, semantics, and morphology in a single dependency formalism with an easily machine parsable grammatical foundation. Further work and study has been done since (Debusmann 2007).

Extensible Dependency Grammar

Extensible dependency grammar is a flavor of dependency grammar that utilizes attribute value matrices in explicitly marking the relationships between the words in a sentence, unlike other flavors where dependencies are only marked through trees or connected graphs; XDG also utilizes a given set of grammatical dimensions that can be used to distinguish syntactic parses of a sentence from semantic, phonological, or morphological ones, etc (Debusmann 2007). Because of this, XDG provides an excellent means of finding dependencies between words in the context of the grammar of a language, as opposed to merely through discovering connections through machine learning techniques. Though all XDG dimensions were available for us to work with in

this creation, only the syntactic dimension is used here, though further work could work the semantic or morphological XDG dimensions into a parse as well.[1] Therefore, in this work, only the syntactic dimension was worked into the entries of the lexicon. We based our possible dependency labels off of an edited subset of Stanford's official dependency labelset, simplifying the original fifty or so labels to an easier fourteen. An example attribute-value-matrix (AVM) that we used in our parsing program for third-person singular verb forms (labeled with a *VBZ* POS tag) is presented in Figure 1 below.

```
⎡   in: {root?, sbar?}                                          ⎤
⎢  out: {subj?, vmod*, obj?}                                    ⎥
 order: {(subj,↑), (subj,obj), (↑,obj), (↑,vmod), (obj,vmod)}
⎨ agrs: {(3, sg)}                                               ⎬
⎢agree: {subj}                                                  ⎥
⎣                                                               ⎦
```

Figure 1: XDG AVM for the POS tag VBZ

The *in* labelset indicates the possible labels for instances where this word is dependent on another. The *out* labelset indicates the possible labels for possible dependencies this word can have. The *order* labelset provides a narrow possible ordering for this word and its dependents in a given sentence. The *agrs* labelset describes the plurality and person of this word[2], features with which surrounding words may have to agree. The *agree* labelset lists which dependencies, if any, this word must agree with based on an element from the *agrs* labelset. Therefore, Figure 1 indicates that a word with this POS tagged label is either the root of a sentence or part of a relative clause, and a single subject and object are possible dependents of this word, as well as any number of adjunctive verbal modifiers. The ordering in the immediate vicinity where this word occurs appears in the linear order of (subject, *word*, object, verbal modifier) where applicable, and the subject must agree in person and number (specifically third-person and singular number) with this word in the immediate clause.

---

[1] For an XDG framework to be sound and complete, a set of principles that explain the interaction between the elements of a sentence must be documented; we have chosen to use the example syntactic principles found in (Debusmann 2007), section 2.2.4, for this task.

[2] Other languages may incorporate other features into *agrs* tuples: languages with noun classes, for example, would list the appropriate noun classes possible to occur with a given word if applicable.

Part-of-Speech Tags

In order to provide the most accurate possible representation of English in the simplest possible application, all English words must be parsable in this XDG form. However, while building a lexicon with every individual English word and its XDG attribute value matrix would present a complete means to parse any English sentence whose complexity lies within the confines of our work, providing a separate lexicon entry for every word form in English is both difficult and memory-intensive, and it will never be complete due to the rapid changing language undergoes daily. Therefore, a different means to creating the lexicon had to be developed to allow for a complete parse that did not require vast sums of data to complete. To solve this issue, we chose to utilize part-of-speech (POS) tags, whose accuracy and ease in finding make them the perfect candidate to use as a lexicon. Therefore, using an altered subset of the Penn Treebank POS tag labelset (Santorini 1991), we have created a parser that focuses on parts-of-speech, instead of individual word entries. The final parser used a list of 43 possible POS tags, a simplified and edited subset from the Penn Treebank's original forty-five or so (de Marneffe and Manning 2008).

The most major change between our tagset and that of the Penn Treebank is that pronouns and irregular verb forms must be given separate entries to ensure proper agreement. These have been marked differently from the other POS tags, with underscores and lowercase letters, as opposed to the traditional format of all uppercase letters.

Nivre's Algorithm

A shift-reduce type algorithm developed by Nivre provided the basis of our parser (Nivre and Fernandez-Gonzalez 2014). This algorithm transforms the sentence into a last-in-first-out stack, from which partially processed words of a sentence are stored, and a first-in-first-out queue, where the unparsed words remain. Through only four separate parsing actions, a sentence can be quickly parsed, and dependency labels are added to

a final list from which a graphical representation of the parse can be created. Each of the parsing actions is outlined below.

The right-arc action acts finds dependencies in instances where the first word *i* of the stack is the head of the first word *j* of the queue. When a dependency is found, the dependent word *j* is removed from the queue and added to the stack; head word *i* is not removed as it would be in the left-arc action, as words to the right of the dependent word *j* may still be further dependents of the head word *i*.

The left-arc action finds dependencies in instances where the first word *i* of the stack is dependent on the first word *j* of the queue. When a dependency is found, the dependent word *i* from the stack is removed from parsing, as any given word in most dependency formalisms are allowed only one head, and in head-final languages like English, if the head has already been found, then there should not remain any further dependents that take the given word *i* as their head. Moreover, the left-arc action cannot take the *root* node of a sentence as a dependent, as that is impossible in this dependency formalism.

The reduce action occurs in cases where a given word on the stack has no more dependents in the queue. In this instance the first word on the stack is removed from parsing, as it cannot provide any further dependencies of either form, left or right.

The shift action occurs in cases where a given word on the stack is neither a dependent nor a head of the first word of the queue. In this instance the first word of the queue is added to the stack, as to allow potential future parsing of following words.

Results

An input sentence where each word is marked with its respective POS tag, after being converted into a data structure usable by our parser, is inputted into a master algorithm that uses each of Nivre's parse actions in turn in the order they have been listed above. For the dependency-arc creation action steps, a separate algorithm compares the *in*-set and *out*-set of two nodes, and if there is a match between them, and if their ordering appears in the *order*-set of the governor node, then their agreement is checked through the governor node's *agrs*-set and *agree*-set. If these too provide a match, or if neither need agreement, then a dependency is formed.

Our parser is designed around a limited set of English sentences that lacked complex dependency phenomena that would break our parser. We present below in Figure 2 the various example sentences we attempted and the number of correct dependency arcs our parser found for each.

Sentences	**Correct Arcs**
(a) I/Pro_I gave/VBD a/DT plant/NN to/IN Bob/PN ./SYM	6/7
(b) The/DT whale/NN that/THAT scared/VBD us/Pro_Us jumped/VBD into/IN the/DT air/NN ./SYM	5/10
(c) Give/VB me/Pro_Me a/DT pie/NN !/SYM	2/5
(d) Bob/NNP is/VVB_Is bigger/JJR than/THAN my/PRP cat/NN ./SYM	4/7
(e) He/Pro_He thinks/VBZ that/THAT you/Pro_You are/VBB_Are mean/JJ ./SYM	4/7

Figure 2: Parsed sentences and their accuracies

Discussion

The number of correct dependency arcs across the board is interesting in nature. Across all sentences, in no instances was the final arc connecting the ROOT and the punctuation present. Moreover, there were instances where even relatively simple dependencies like nouns to their determiner were missed by the parser, despite being found in other sentences. This clearly indicates some kind of error in either our parsing algorithm or in our lexicon, though a more thorough investigation of both is needed to know for sure. Appendix A (found after References) consists of graphical comparisons of our parses and the correct expected parse. They have been created using an online chart-designer, though future work could consist of creating an automatic graphical representation as well.

Although the results were not as optimistic as originally expected, the very fact that parses arrived as well as they did is encouraging, and we believe that after a thorough rework of the lexicon and the algorithm using knowledge we learned through the parser-construction process, accuracy will most likely drastically increase. The

viability of utilizing XDG as opposed to some form of machine learning is evident, though more work is needed to see to what extent this can be done.

Conclusion

We have developed a means of dependency parsing, not through traditional machine learning methods, but instead through a grammar-based system based around the syntactic dimension of extensible dependency grammar, centered around a lexicon based on part-of-speech tags. Though only done on a small subset of both dependency labels and POS tags, the process of expanding these to become more inclusive to a varying number of sentence forms would only involve the editing and expanding of the lexicon, with no change to the remainder of the code.

Although our initial results show promise, this method of parsing runs quickly into an immediate theoretical problem: complex parsing phenomena, like fronting and coordination, have no easy translation into XDG format. Because these phenomena, and any other phenomena not easily handled through dependency formalisms, are not regular in usage, trying to provide attribute-value-matrices for each and every case perhaps proves impossible, as there is too much variation to cover to be able to maintain feasible explicit marking of possible dependents and orderings. Because of this, this method of parsing will perhaps never be able to cover the full scope of a language, instead always dealing with partial, more-regular grammar fragments instead.

Despite this, more work can be done in exploring this type of parser. Potential next steps include building non-projectivity into the parser, either through pseudo-projective methods (Nivre and Nilsson 2005) or through the inclusion of a swap action as outlined in (Nivre 2009). This is a potentially simple addition, though it's inclusion could possibly increase the parsing time by more than would be acceptable, unless we chose a different, more efficient programming language. Moreover, this process can be extended to other languages; it is possible that a language with fewer possible POS tags may prove easier to parse, though that cannot be known for certain unless more research is done.

References

Covington, M. A. (2001). A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference* (pp. 95-102). Athens, GA: University of Georgia. Retrieved from http://web.stanford.edu/~mjkay/covington.pdf

Debusmann, R. (2007). *Extensible Dependency Grammar: A Modular Grammar Formalism Based On Multigraph Description* (Unpublished doctoral thesis). Saarland University. Retrieved from https://www.ps.uni-saarland.de/~rade/papers/diss.pdf

Debusmann, R., Duchier, D., & Kruijff, G. M. (2004). Extensible Dependency Grammar: A New Methodology. In *Proceedings of the COLING Workshop on Recent Advances in Dependency Grammar* (pp. 78-85). Geneva, Switzerland. Retrieved from http://www.aclweb.org/anthology/W04-1510.pdf

de Marneffe, M., & Manning, C. D. (2008). *Stanford typed dependencies manual* (Tech.). Retrieved from http://nlp.stanford.edu/software/dependencies_manual.pdf

Goldberg, Y., & Nivre, J. (2012). A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012: Technical Papers* (pp. 959-976). Mumbai, India: COLING 2012. Retrieved from http://www.aclweb.org/anthology/C12-1059

Nivre, J., & Fernández-González, D. (2014). Arc-eager parsing with the tree constraint. *Computational Linguistics, 40*(2), june 2014, 259-267. Retrieved from http://www.aclweb.org/anthology/J14-2002

Nivre, J. (2003). An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)* (Vol. IWPT 2003, pp. 149-160). Nancy, France. Retrieved from http://stp.lingfil.uu.se/~nivre/docs/iwpt03.pdf

Nivre, J. (August 2009). Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP* (pp. 351-359). Suntec, Singapore: ACL and AFNLP 2009. Retrieved from http://www.aclweb.org/anthology/P09-1040

Nivre, J., Kuhlmann, M., & Hall, J. (October 2009). An Improved Oracle for Dependency Parsing with Online Reordering. In *Proceedings of the 11th International*

Conference on Parsing Technology (pp. 73-76). Paris, France: IWPT 2009. Retrieved from http://www.aclweb.org/anthology/W09-3811

Nivre, J., & Nilsson, J. (2005). Pseudo-Projective Dependency Parsing. In *ACL '05 Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 99-106). Ann Arbor, MI. Retrieved from http://www.aclweb.org/anthology/P05-1013

Popel, M., Marecek, D., Stepanek, J., Zeman, D., & Zaborkrtsky, Z. (2013). Coordination Structures in Dependency Treebanks. In *Proceedings in the Annual Meeting of the Association of Computational Linguistics, 2013* (pp. 517-527). Sofia, Bulgaria: ACL 2013. Retrieved from https://ufal.mff.cuni.cz/~popel/papers/2013_coordination.pdf

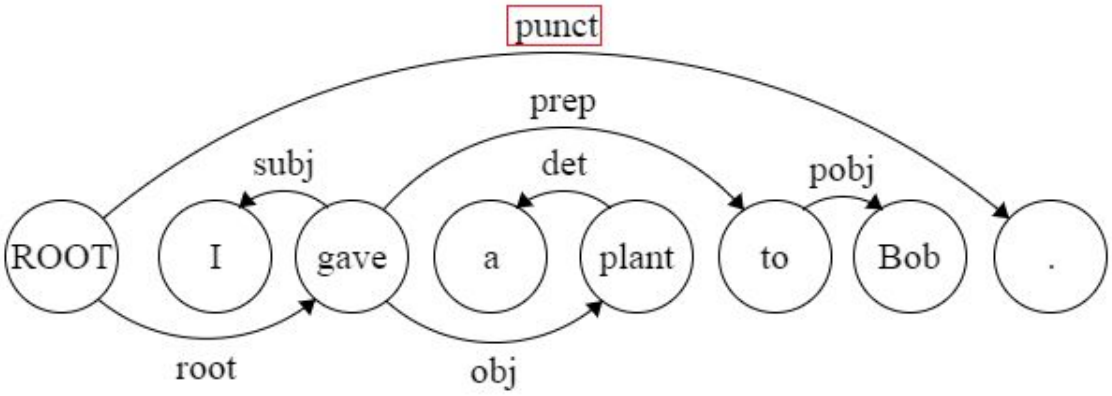
Santorini, B. (1991). *Part-of-Speech Tagging Guidelines for the Penn Treebank Project* (Tech.). Retrieved from http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/Penn-Treebank-Tagset.pdf

Yamada, H., & Matsumoto, Y. (2003). Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)* (pp. 195-206). Nancy, France. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2289&rep=rep1&type=pdf

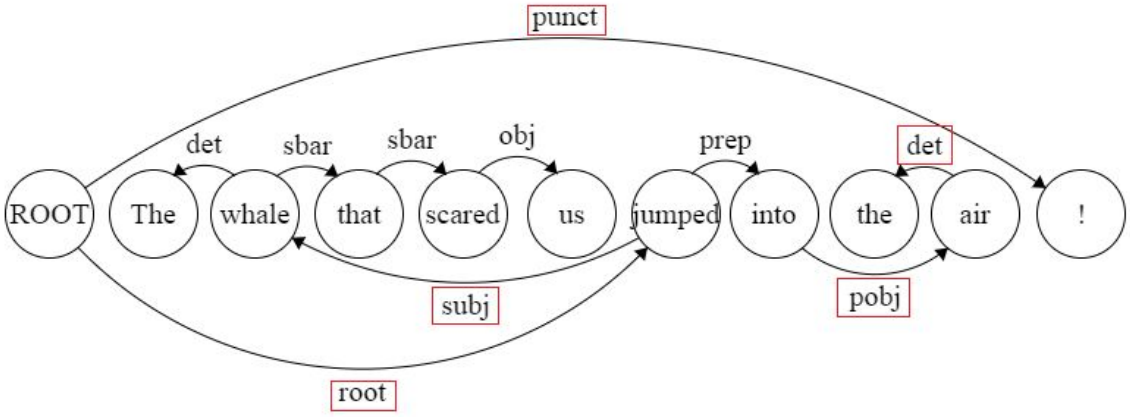
Appendix A

Each of the five example sentences are given graphical representations here. Missed arcs are marked in red boxes. Labels that were marked incorrectly are crossed through in red, and their incorrect label is indicated above the correct one. Graphs are made in a dual-layered condensed fashion to reduce space.

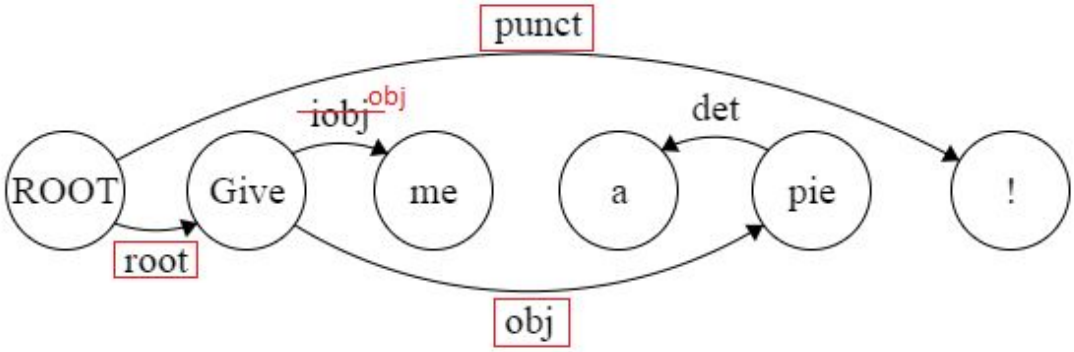
(a)



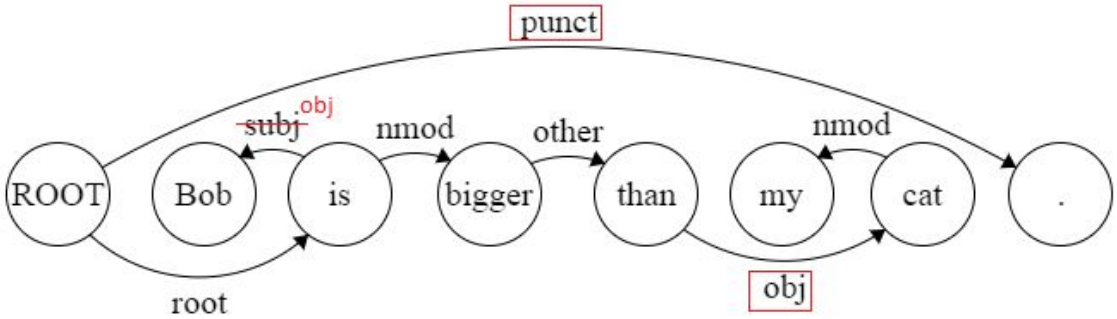
(b)



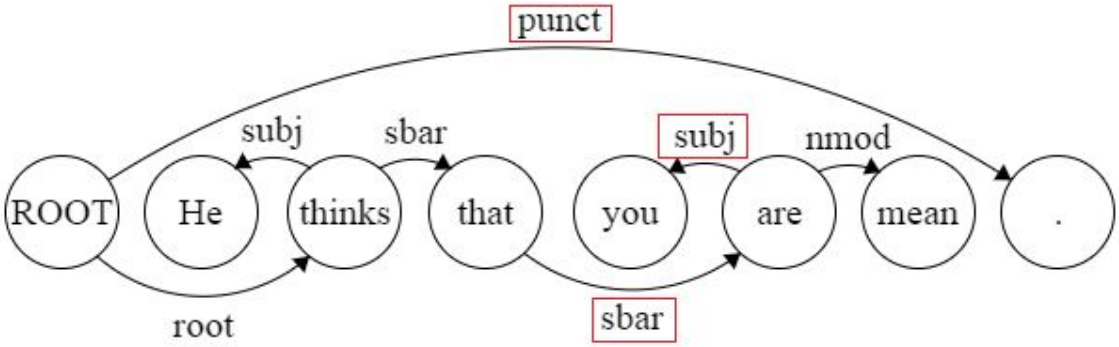
(c)


punct
iobj
obj
det
ROOT
Give
me
a
pie
!
root
obj

(d)


punct
subj
obj
nmod
other
nmod
ROOT
Bob
is
bigger
than
my
cat
.
root
obj

(e)


punct
subj
sbar
subj
nmod
ROOT
He
thinks
that
you
are
mean
.
root
sbar