Linguistics 98a Final Project

Sharanya Pulapura

October 2017

1 Introduction

One of the foremost research goals in the field of natural language processing is determining the relationship between a string of characters and a statement with meaning. That is, how can a computer interpret the meaning of a given text? And furthermore, how can a computer generate a string of characters which a human can read and interpret with a given meaning?

The first step in bridging the gap between text and meaning is at the individual word level. A computer should be able to parse a string of natural language and identify the part of speech of each word. With the development of many part of speech tagged corpora, it becomes easier for computers to identify the part of speech of each word in an input text, based on data from the corpora and the surrounding words. However, being able to identify the part of speech of each word is obviously insufficient to interpret the meaning of the text [1]. Thus, the next step towards natural language interpretation is to arrange a given sentence into its syntactic structure, in the form of a syntax tree.

The goal of this experiment was to determine the most likely syntactic position for a given word, focusing specifically on adverbs. The ability to determine the syntactic position of an adverb is particularly interesting because of the wide range of syntactic positions that an adverb can occupy. For example, certain types of adverbs including "transition" words like "nevertheless" often appear at the beginning of a sentence, while others often appear directly before or after the verb they modify. In this experiment, I used the Penn TreeBank (PTB) Wall Street Journal corpus, which contains data from 2,499 Wall Street Journal articles. Each word is annotated with its part of speech, and each sentence is annotated with its X-bar syntactic tree [2]. Using the PTB, for each different type of adverb phrase annotated within the corpus, I determined the

[1] Matsubayashi, Yuichiroh, et al. "Generalization of Semantic Roles in Automatic Semantic Role Labeling." Journal of Natural Language Processing, vol. 21, no. 4, 2014, pp. 841–875., doi:10.5715/jnlp.21.841.

[2] "Treebank-3." Linguistic Data Consortium , University of Pennsylvania , 30 Oct. 2017, catalog.ldc.upenn.edu/ldc99t42.

most frequent nodes which governed the adverb phrase, the most frequent nodes which were also governed by that adverb phrase's parent node, and the most frequent nodes which were in turn governed by the adverb phrase. From these data, I determined the likelihood that each type of adverb would occur in a given syntactic position. Certain types of adverb phrases consistently appeared in the same position: for example, WH-adverb phrases almost always appeared directly under the SBAR level and Locative adverbs almost always appeared directly under the VP. Other types of adverbs had a greater degree of syntactic variation. Nevertheless, we can use the results of this experiment to determine the most likely syntactic position for a given type of adverb phrase for an AI attempting to interpret or generate text.

2 Methods

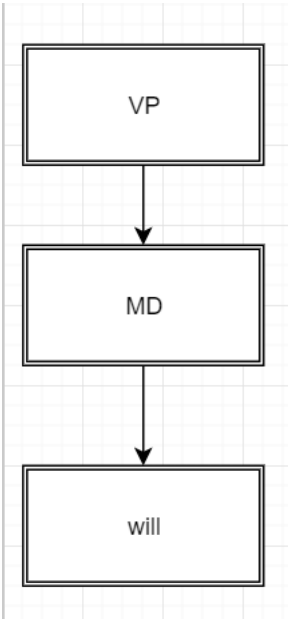
2.1 The Tree Data Structure

The basic framework of this project was a syntax tree data structure designed to store the syntax data from the PTB. This data structure (the 'Tree' class in the attached Python file) is a simple modification of the standard tree data structure, in that each tree is recursively composed of smaller trees. A given instance of the Tree class represents a single node of a syntax tree and has five fields—token (a string: the syntax tag, part of speech tag, or word stored at that give node), parent (a pointer to the node that immediately dominates the current node), children (a pointer to a list of nodes directly dominated by the current node), level (an integer, level n means that a given node starts at the nth character of the line in the PTB file), and head (a pointer to the node at the top of the tree that this node is a part of). In addition to basic field accessor and modifier methods, each Tree class also has functions to add a child directly to a node in a Tree and add another Tree into the given Tree, as well as functions which determine if a given Tree is a leaf node (has no children) and to print a Tree out as a string. The final function in the Tree class will be discussed in depth later, but essentially serves to find and return all sub-trees of a given tree that contain an adverb.

2.2 Parsing PTB Data

Before I began analyzing the position of adverbs within the syntax trees, I first had to parse the data from the PTB and store it in the Tree data structure for ease of manipulation and interpretation by the computer. First, I compiled all the PTB ".parse" files into one document containing all the syntax trees in the corpus (attached as "allwsj.parse"). Reading this file line by line, I used the regular expressions to find the word stored in each line and all the syntax tags for all the nodes preceding the word on the given line. I concatenated these

matches as well as the character number (for the "level" field) at which each match occurred in the line and then stored each line as a Tree where each node was the parent of the node to its right. For example, the line "(VP (MD will)" would become the Tree :



Then, for each sentence (i.e. for each collection of lines not separated by a blank line), I composed all the individual line Trees based on the level of the top node, using the add function in the Tree class. Essentially, this function identifies where in the overall Tree a sub-Tree goes using the fact that all children of a given node will have the same level/ amount of indentation in the .parse file. Ultimately, this gave me a list of Tree instances which represented each syntax tree given in the PTB corpus.

2.3 Identification of Adverb Phrases and their Environments

After storing all the data within the Tree data structure, I iterated through the list of Trees for all the sentences in the PTB to find the syntactic structure of adverb phrases—for this purpose, I used the find_adv() function in the Tree class. Recursively traversing each node of a given Tree, this function returns a list of all sub-Trees within the Tree that begin with a node which has some type of adverb phrase as a child. For example, given the following Tree:

```
(TOP (S (S-TPC-1 (NP-SBJ (NP (NP (DT A)
                                 (NN form))
                             (PP (IN of)
                                 (NP (NN asbestos))))
                         (VP (ADVP-TMP (RB once))
```

```
                        (VBN used)
                        (NP (-NONE- *))
                        (S-CLR (NP-SBJ (-NONE- *PRO*))
                               (VP (TO to)
                                   (VP (VB make)
                                       (NP (NNP Kent)
                                           (NN cigarette)
                                           (NNS filters)))))))
                (VP (VBZ has)
                    (VP (VBN caused)
                        (NP (NP (DT a)
                                (JJ high)
                                (NN percentage))
                            (PP (IN of)
                                (NP (NN cancer)
                                    (NNS deaths)))
                            (PP-LOC (IN among)
                                    (NP (NP (DT a)
                                            (NN group))
                                        (PP (IN of)
                                            (NP (NP (NNS workers))
                                                (VP (VBN exposed)
                                                    (NP (-NONE- *))
                                                    (PP-CLR (IN to)
                                                            (NP (PRP it)))
                                                    (ADVP-TMP (NP (QP (RBR more)
                                                                      (IN than)
                                                                      (CD 30))
                                                                  (NNS years))
                                                              (IN ago)))))))))))
        (, ,)
        (NP-SBJ (NNS researchers))
        (VP (VBD reported)
            (SBAR (-NONE- 0)
                  (S (-NONE- *T*-1))))
        (. .)))
```

we would return a list containing the following sub-Trees:

```
(VP (ADVP-TMP (RB once))
    (VBN used)
    (NP (-NONE- *))
    (S-CLR (NP-SBJ (-NONE- *PRO*))
           (VP (TO to)
               (VP (VB make)
                   (NP (NNP Kent)
                       (NN cigarette)
                       (NNS filters)))))))
```

```
(VP (VBN exposed)
    (NP (-NONE- *))
    (PP-CLR (IN to)
            (NP (PRP it)))
    (ADVP-TMP (NP (QP (RBR more)
                      (IN than)
                      (CD 30))
                  (NNS years))
              (IN ago))))))))))))
```

(Note: above the Trees are written out in PTB string format, my program instead handles them as instances of the Tree class).

This way, we not only know the syntactic position of the adverb phrase, we also know what other types of phrases are often found as the parents, siblings, or children of adverb phrases, which enables us to understand the larger syntactic structure surrounding the adverb phrase and gives the computer a more complete picture of the adverb phrase's usual environment.

2.4 Analysis of Adverb Phrases

Once I obtained each adverb phrase sub-Tree, I determined the nodes which were most likely to appear as parent, sister, and child nodes of the adverb phrase. I created three dictionaries which mapped each type of adverb phrase to a list of all the nodes which appeared as that type of adverb phrase's parent node, sister node, or child node (d_prev, d_same, and d_next respectively). Then, for each type of adverb phrase, I created three new dictionaries which mapped each parent node, each sister node, and each child node to its frequency for that particular adverb phrase. For example, SBAR occurred as the parent node to WHADVP-1 in 0.504 of all cases, so within the parent dictionary for WHADVP-1, SBAR was mapped to 0.504. I then compiled all the data into three .csv files: "prevs.csv", "sames.csv", and "nexts.csv" which contain a listing of each type of adverb phrase, the nodes which occur as the parent/sibling/child of that adverb phrase more than 0.1 (10%) of the time, and the decimal frequency for each of those nodes. From these files, I was able to determine the most common syntactic structures surrounding adverb phrases.

3 Results

Complete tabulated results can be found in the three attached .csv files. However, I will summarize and discuss the most important findings which emerged from my analysis.

3.1 Parent/Dominating Nodes

In the majority of cases, adverb phrases are preceded by a VP. In fact, the majority of types of adverb phrases were immediately dominated only by VPs in all sentences in the corpus. One notable exception to this pattern was the WHADVP, which was most often preceded by SBAR. WHADVPs are annotated in the PTB with a number which indicates that it is the nth instance of a WHADVP in a sentence, but in all but the case of WHADVP-n that were not in sentence fragments, the WHADVP was immediately dominated by SBAR over 50% of the time (often over 75% of the time). Another clear exception was TPC/topical adverbs, which are most often preceded by either S or SINV. It is interesting to note that of the 25 different adverb phrase types that included a TPC tag, 23 were most often dominated by SINV(inverted declarative sentence), while only 2 were most often dominated by S. Finally, for some of the adverb phrase types which were most often dominated by a VP, there were other nodes that dominated them in a reasonably large minority. For example, for a simple ADVP, 33% of occurrences were dominated by S rather than VP. Many of the temporal adverb phrases were often dominated by NP, including ADVP-TMP (31% dominated by NP) and ADVP-TMP-PRD (17% dominated by NP).

Manner adverbs (ADVP-MNR) were dominated by NP 12% of the time, and locative adverbs (ADVP-LOC) were dominated by NP 17% of the time [3].

3.2 Child/Dominated Nodes

The node most often dominated by an adverb phrase was, unsurprisingly, the adverb. However, there were many types of adverb phrases which often dominated the -NONE- operator, including temporal, locative, manner, and Wh-adverb phrases, and notably purpose or reason adverb phrases (ADVP-PRP), which dominated -NONE- 93% of the time. This shows us that these types of adverb phrases often modify a node which must undergo a movement operation. Finally, there are many types of adverb phrases which dominate prepositional phrases, including notably the simple ADVP-1, which actually dominates prepositional phrases equally as often as adverbs (30% of the time), and the various adverb phrases which mark the locative complement of put (-PUT tag), which dominate PP 21% of the time. Finally, several types of adverb phrases dominated prepositions or subordinating conjunctions (IN) over 10% of the time, including ADVP-DIR (direction), ADVP-CLR (closely related), and ADVP-PRD-LOC (predicate locative)[3].

3.3 Sister Nodes

It is more difficult to identify patterns in the types of sister nodes that certain types of adverbs tended to have. This makes sense since there are many types of adjuncts which may be modified by an adverb and many types of nodes which may modify the same adjunct as the adverb. In this data set, patterns are particularly difficult to identify since the PTB categorization of each syntactic position is so fine-grained. Thus, for example, types of adjective phrases like "ADVP-TMP-CLR-TPC-1" occur very few times even in such a large corpus, so there is not enough information to tell what sister nodes such a phrase would usually have. This phenomenon is discussed further in the next section regarding future research.

4 Conclusions and Future Work

The main takeaway from this experiment is that, while many types of adverb phrases have a certain syntactic environment in which they usually appear, there are a number of types of adverb phrases which do not have a "standard"

[3]Penn Treebank Tags. Massachusetts Institute of Technology, 30 Oct. 2017, web.mit.edu/6.863/www/PennTreebankTags.html.

[3]Penn Treebank Tags. Massachusetts Institute of Technology, 30 Oct. 2017, web.mit.edu/6.863/www/PennTreebankTags.html.

syntactic environment and instead have several different potential environments that occur significantly often. For example, while Wh- adverb phrases almost always come after an SBAR, basic ADVPs are often dominated by either S or VP. However, based on these observations, there is little we can do to actually train an AI to identify the syntactic position of a given adverb without further information. Thus, the clear next step would be to break down the types of adverb phrases with highly ambiguous potential syntactic position to see if there are certain specific adverbs which occur in one of the potential positions more frequently than the others. Thus, if the AI were to encounter an adverb phrase type that has several potential syntactic environments, it would check the actual adverb itself to make the decision on which syntactic position to choose when building its tree.

Another potential next step would be to create broader categories for classifying the types of adverb phrases than the PTB uses. As mentioned earlier, the PTB has a highly specific tagging system which often means that certain types of adverbs only occur a few times in the entire corpus, which is not enough information to produce a definitive conclusion. Thus, if we were to group all the adverb phrase types into a small amount of broader categories, we would have more data for each category and thus would be able to see more patterns. It would also be effective to experiment with several parameters for defining the broad categories, since that would tell us which qualities of an adverb phrase are most important for defining its syntax.

The long-term goal of this research would be to actually use these results to train an AI. That is, with a more refined statistical analysis of the relationship between an adverb phrase and its syntactic position, an AI would be able to "read" a line of (untagged) text and place the adverb in the correct position in a syntax tree. With an accurate syntax tree, the AI would be able to parse how each word functions in the sentence, which would bring it closer to "understanding" natural language.