Introduction to Natural Language Syntax and Parsing
Lecture 1: Automatic Linguistic Annotation

Stephen Clark

September 29, 2015

Automatic Linguistic Annotation We would like to automatically annotate linguistic units (typically sentences) with some linguistic structure, in order to facilitate various NLP tasks and applications, such as (semantic) search, question answering, information extraction, machine translation, and so on. We might also want to model some aspects of linguistic structure for linguistic, or cognitive science, reasons, but in this part of the course we'll be using NLP — with more of an engineering focus — as the main motivation.

Sentence Segmention One of the first tasks in any NLP pipeline is often sentence segmentation – breaking the document up into sentences. This may appear trivial — e.g. just split on periods — but the period-splitting heuristic is not going to work. In English, periods serve a number of functions, for example to mark abbreviations, as in the Dr. example. One approach to this problem is to manually write a number of rules, e.g. split on a period unless the period follows Dr. or Mr. or Mrs. or... The difficulty with this approach is that the rule set, in order to cover all the cases, soon becomes unwieldy and difficult to modify and maintain. Hence, as in the rest of NLP, a machine learning approach is often taken, using manually segmented sentences as training data.

One general comment applying to all 8 lectures applies here: in the examples we'll often be using well-edited text such as newspaper text or Wikipedia articles. However, there is currently a lot of interest in processing less well-edited text, such as tweets or other postings on social media, for example to determine whether customers are saying positive things about a particular product, or to predict stock market prices or influenza outbreaks. Hence one question you should keep asking yourself is: would this proposed method work on Twitter data?

Tokenisation (What's a Word?) The next task in the canonical NLP pipeline is often tokenisation, the task of breaking the sentence into tokens. This is useful because we may have learnt token-level translation rules, for example, or we may be searching for information about Dr. Black, in which case it's useful to know that the sentence contains the token *Black*, as opposed to *Black's*. Having said that, there is currently interest in processing sentences at the *character* level, using neural network models, and not segmenting at the word level at all (or even at the sentence level).[1]

A second general comment about the 8 lectures applies here: the default language we will assume is English. However, there are a number of features of English which are not representative of the world's languages. First, there are many languages, such as Turkish, which are much more morphologically complex than English. A single word in these languages can be used to express a concept which requires many words in English. Hence the questions of how to do tokenisation and parsing are rather different for these languages. Second, there are a number of languages — the canonical example being Chinese — which do not use spaces to separate the words. In fact, the very notion of a word is controversial in Chinese, and native speakers do not exhibit high agreement on where to place the spaces if asked to perform word segmentation. Despite this lack of agreement, there is a large literature on the task of Chinese word segmentation; for a recent paper see [3].

But even in English the question of how to do tokenisation is not always clear-cut. Should *medal-winning* be one token or two? It perhaps depends on the application. If we have a translation for *medal-winning*, then it makes sense to keep it as a single token when doing translation. If we're looking for information about what Dr. Black has won, then splitting it may make sense. These questions arise in particular when processing biomedical text, which uses a lot of characters, such as hyphens, outside of the standard alphabet. Hence two more questions you should keep asking yourself are: will it work for Chinese/Turkish/Swahili, and will it work for biomedical text?

Part-of-Speech Tagging The next stage is part-of-speech tagging, where we begin to add grammatical structure which is not overtly realised in the sentence. You have learnt a lot about possible tagging schemes in the other half of the course. The task of assigning the tags, which can be thought of as a *sequence labelling problem* from machine learning, is a classic task in NLP. For well-edited English text for which there is plenty of manually annotated data to learn from, e.g. newspaper text, and for relatively small tag sets, e.g. the Penn Treebank tagset, POS tagging is close to being a solved problem (although not completely solved). Again, tagging for Twitter and biomedical text is harder. There are a number of freely available POS taggers. The Stanford tagger is one of the most widely used [2].

Syntactic Parsing - Phrase Structure Now we begin to see some hierarchical structure. I will say more about phrase structure, and the resource typically used to build phrase-structure parsers, in the next lecture.

[1]In order to keep the number of readings to a manageable level, I do not always include references when referring to the literature; but if you wanted to find a recent paper about character level parsing, for example, a Google search for "character level chinese dependency parsing" will do the trick.

Semantic Parsing - Logical Form If we wanted to be really ambitious, we could try and construct a logical form. The example is from a semantic analysis tool called Boxer, which builds on the Combinatory Categorial Grammar parser we'll hear more about later in the course. You'll also learn more about semantic analysis and interpretation in the other half of the course. The details aren't too important at this stage, except to say that the example is a pretty-print of a representation which is essentially a piece of first-order logic. The advantage of translating into logic is that there are ready-made inference procedures, and tools, available which are straightforward to implement on a computer. The disadvantage, as is known from decades of work in AI which attempts to translate natural language into some formal language, is that inferences in NLP typically require large amounts of linguistic and world knowledge, even if the translation into logic can be successfully automated.

Syntactic Parsing - Dependency Structure This is the representation that we're going to focus on in this half of the course, for reasons I'll give in the next lecture. One interesting feature of the example is that the dependency links cross – more on this later.

Why is Parsing Difficult? Natural languages exhibit many different structures. For phrase structure grammars in particular, many different grammatical rules are needed to cover all these structures. Obtaining those rules, either manually through an expert linguist writing the grammar, or (semi-)automatically through some learning procedure from corpus data, is a challenging task.

The second reason parsing is difficult is perhaps one of the more surprising properties of natural languages that NLP has uncovered in the last few decades. Natural languages exhibit large amounts of syntactic ambiguity. The reason we don't see it as humans is because our own language processors are extremely effective at using context to perform the disambiguation. Note also that, as grammars become more comprehensive, solving the first problem, this only increases the level of ambiguity, making the second problem even worse.

Syntactic Ambiguity The classic text book example of syntactic ambiguity is *John saw the man with the telescope.* Is John looking through the telescope at the man, or is the man holding the telescope? The two semantic readings result from different syntactic parse trees.

Syntactic Ambiguity: the problem is worse than you think The classic example is useful, because the ambiguity is easy for humans to see, but also misleading. To resolve it would require contextual representations, world knowledge, and general reasoning capabilities currently not available. It's also the case that either reading could be possible. Natural language ambiguity is pernicious, because it's precisely the cases we don't easily perceive, such as *John ate the pizza with a fork*, which cause the problems for current parsing technology. Here only one of the readings is plausible, and it's the job of the parser to decide which one.

Syntactic Ambiguity: the problem is even worse than that It's not just the existence of "hidden" ambiguity which is the problem, it's the fact there is so much of it. Many constructions, including PP attachment, coordination, relative clause attachment, lead to alternative possibilities which multiply when chained together. In the example sentences, the number of possible analyses grows exponentially with the number of PPs, following the Catalan series. Of course natural languages don't contain sentences quite like these, but they do exhibit chains of attachment decisions which have this property. A favourite moment from my own research occurred when calculating, with James Curran, the number of parses for a long newspaper sentence given by our CCG parser (using an efficient dynamic programming technique so the counting can be performed exactly). The number was close to the Avogadro constant you may have encountered in high school chemistry, 6.022×10^{23}.

Readings for Today's Lecture In addition to the references below, Chapters 9 and 10 of Manning and Schutze, and Chapter 5 of Jurafsky and Martin (2nd. Ed.), are useful readings for POS tagging. Only part of the Zhang and Clark reference below is relevant for today's lecture, but the whole article will become relevant as the course progresses. A useful reference which covers much of the course is my book chapter on statistical parsing [1]. All papers are freely available on the web.

References

[1] Stephen Clark. Statistical parsing. In Clark, Fox, and Lappin, editors, *Handbook of Computational Linguistics and Natural Language Processing*, pages 333–363. Blackwell, 2010.

[2] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the HLT/NAACL conference*, pages 252–259, Edmonton, Canada, 2003.

[3] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151, 2011.