# Annotated Bibliography

Dylan Orris

October 13, 2018

## References

[BL04] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

This article discusses the goals of the Natural Language Toolkit (NLTK), a system implemented in Python meant for use in computational linguistics courses. The toolkit aims to make the creation of NLP systems simpler for amateur coders and linguists, who are able to then focus on what they are doing (the actual act of parsing) rather than being bogged down in learning languages - Perl and Prolog are explicitly mentioned here. The toolkit has many modules, including some for mapping mathematical functions.

[Cho03] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

Chowdhury gives an overview of Natural Language Processing and the many areas of its research. The aspects of his discussion which relate to this project are information extraction and natural language interface. Information extraction consists of many different methodologies, with varying levels of success and ease of implementation. Chowdhury discusses issues NLP has with feature tagging, co-reference resolution, and discourse analysis. Feature tagging is finding the important features of phrases, sentences, and paragraphs and indicating where they lie. Co-reference resolution refers to the difficulty that NLP programs typical have with multiple references to the same object, especially when done with different wording. Discourse analysis is essentially the importance of context to semantic meaning. Chowdhury discusses a method called ‘template mining’, in which a sort of template is implemented for the processed data which can then quickly and accurately pull out important information. The difficulty here is that different formats are needed for different fields, so no universal template is truly feasible. However, for this project, template mining could prove useful if claim statements (when properly made) only come in certain structures. This would allow the analysis to first find the structure of the statement by comparing it with several templates, and when a match is found, rapidly pull out the necessary information for testing. This would eliminate more conversational ways of requesting information, however. Another issue for NLP is the ability to expand it to different realms. Many processors which work wonderfully for tasks such as article summaries have great difficulty when expanded to new domains with new, more varied word choices. Chowdhury also discusses natural language interfaces, situations in which a user interacts with the program with speech or conversational writing to request information. Already in 2003, with simple questions, programs could deliver the correct answer over 60% of the time. Most issues with understanding human dialogue comes from difficulties with discourse analysis and a lack of user models. User models create an interesting question for this project; should it allow the less mathematically able to query it just as effectively? Ideally the answer is of course yes, but this will require greater interpretive power, as language will be less precise and more ambiguous. It may be possible to create links between common phrases and their proper references, such as a question like 'Computer, please prove FOIL for me'. Here, we must know that the user is referencing First Outside Inside Last, a common saying when referring to distributive multiplication. By understanding the different ways users will phrase questions, we can expect a more compelling program and better responses. Memory also becomes an important part of the program, as it should be able to respond to requests such as 'Show me how $x$ step works'. The program must then find what it labelled as the xth step in the proof, and then display the axioms and other pertinent information to the user.

[Lyt86] Steven L Lytinen. Dynamically combining syntax and semantics in natural language processing. In *AAAI*, volume 86, pages 574–587, 1986.

Lytinen's article discusses issues with what were, at the time, the two main paradigms for natural language processing. One camp felt that syntactic structures should be examined first, with ambiguity later cleared up using semantics. The other felt that these concepts could not be split apart, so semantics and syntactics would be examined simultaneously by the

processor. Lytinen explains that the semantics-second approach leads to a great deal of ambiguity and slow runtime. By ignoring semantics, which could easily clear up some ambiguity present in language, statements may rapidly become n-ambiguous, meaning that there are n points of ambiguity. This ambiguity rapidly multiplies, as two ambiguous points in a row mean there are four possible meanings for the sentence, assuming none are mutually exclusive. Lytinen then describes issues with the view of simultaneous semantic and syntactic analysis. Though this system clears many ambiguities away, making it less computationally expensive, it becomes space inefficient rapidly. This is because its rules are not generalizable, as the focus on semantics leads to many nearly identical rules being applied, but being programmed as though they are different. Finally, Lytinen introduces MOPTRANS, a parser which translates stories about terrorism. This parser uses semantics on statements within recent memory as a method to clear ambiguity early, while allowing for rules to be generalized. The parser works by checking how items in memory compare to rules that are known by the parser, and groups them together early when high percentage likelihood matches are made. This system is storage efficient, while also retaining the benefits of considering semantics. While this may not be particularly applicable to my project, understanding the relationship of syntax with the semantics I work with may allow for more complex statements to be input and understood. Rather than requiring all claims to come in a certain form, it may be possible to program certain rules based on how words apply to their surroundings to better understand user input. The inclusion of multiple subjects or fact bases in a single claim would currently break my parser, so this article was useful in seeing an effective way of dealing with the ambiguity which may arise in these situations.

[MD08] Nitin Madnani and Bonnie J Dorr. Combining open-source with research to re-engineer a hands-on introductory nlp course. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 71–79. Association for Computational Linguistics, 2008.

This article discusses the use of NLTK (described in a separate source) in an introductory computational linguistics course and its effectiveness. The article indicates that the toolkit allows for much better understanding of natural language processing and its intricacies, and allowing for complex task implementation, without requiring knowledge of several languages or several libraries. Students compare NLTK favorably to the previous course system, which used several programs in addition to coding in C or C++, depending on the task. NLTK is also open source, allowing for easy modification for other uses.

[PB93] James Pustejovsky and Branimir Boguraev. Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 63(1-2):193–223, 1993.

This article discusses the benefits of using processing power to determine what part of processed language a word is, rather than encoding every possible way it could be used. By attempting to enumerate every way to use a word or phrase, it is quite possible that the parser becomes confused when words are used in a new and novel way. By using a generative lexicon, we can eliminate this problem. The generative lexicon examines the text at a deeper level, attempting to use relative position and relation to other words to determine how each word behaves. Pustejovsky argues that this method not only captures similarities in meaning that enumerative parsers miss, but can be more storage-effective. By examining the difference between 'forgot to' and 'forgot that', Pustejovsky explains that while whatever was forgot did not occur in the first claim but did in the second, there is a clear underlying relation between the forgots. Rather than ignoring this and building in two separate rules for forgot into the parser, it is possible to look deeper into meaning.

[Res99] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research*, 11:95–130, 1999.

Resnik describes using measures other than edge-counting in order to classify semantic objects. In simple edge-counting, each word belongs to a category, which belongs to a super-category, and so on. Resnik cites the example of coinage, where a nickel would belong to the coin category, which falls under money, which in turn falls under methods of exchange, and so on. By using systems like this, a level of distinction between words can be found, with words requiring few edge transitions being considered closely related, while those with many jumps being distantly related. Resnik attempts to improve on this system by using a categorizer that takes many instances of words, and creates a percentage association with the word and a subsuming concept, which is referred to as WordNet. For example, nurse would have a high association with health professions, but would have a moderate connection with childcare, due to the term ‘wet nurse’. Resnik does encounter some difficulties, as certain words will be too closely related due to the consideration of slang terms. In one example, tobacco is more closely related to horse than alcohol. This is due to horse being a slang term for the narcotic heroin, thus placing them both in the same narcotic category.

[Sme92] Alan F Smeaton. Progress in the application of natural language processing to information retrieval tasks. *The computer journal*, 35(3):268–278, 1992.

Smeaton discusses the now reliable, robust, and efficient techniques in NLP which can be used in information retrieval in text. Previously, information retrieval used statistical methods with many limitations. Among goals that Smeaton places for NLP are the disregarding of ’semantic nonsense’ which are sentences that, while syntactically correct, mean nothing. By understanding these sentences, NLP systems can reduce ambiguity in further readings, by recognizing that certain verbs and descriptions cannot be applied to possible domains. Smeaton also sets his eyes on discourse level analysis, where meaning is greatly tied to context. Smeaton describes TSAs, tree-structured analytics. These differ from parser trees, as TSAs are ‘binary trees whcih encode rather than enumerate structural syntactic ambiguities as markers on non-leaf nodes’. Through using these trees, it is possible to identify instances where ambiguity occurs and maintain the ambiguity for further processing by the system, rather than accepting a faulty understanding. Smeaton ends with a reference to issues with improving NLP from the side of those who are retrieving information, as these groups, at least at the time, did not provide much of anything to the processors. They were simply regarded as a black box, with no updates made by the researchers using them for information retrieval, greatly slowing down NLP progression.