INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Natural Language Processing on Ambiguous Sentence Using NLP Tools: Core NLP, Apertium and PRAAT

Taranpreet Singh Saini
Student, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India
Priyanka Lokhande
Student, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India
Renuka Deshmukh
Student, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India
Dipali Baviskar
Student, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India

Abstract:
Natural Language Processing (NLP) is the computerized approach to analyzing text that is based on both set of theories and a set of technologies. It can take the input in any form like text, audio and in any language. Pattern modules help us in understanding the natural language processes and how easily we can use it. In this paper, we are focusing on three NLP tools Core NLP, Apertium and PRAAT and using them on single ambiguous sentence.

Keywords: *Natural Language Processing (NLP), Pattern Modules, NLP Tools, Core NLP, PRAAT, Apertium.*

1. Introduction

Natural Language Processing (NLP) is a region of research and application that analyzes how computers can be used to understand and influence natural language text. NLP researchers aspire to collect knowledge on how human beings understand and make use of language so that relevant tools and techniques can be refined to make computer systems understand and influence natural languages to achieve the preferred tasks. Applications of NLP comprise a number of fields of studies, like machine translation, natural language text processing and natural language text summarization, user interfaces, speech recognition, artificial intelligence and expert systems, and many more.

"Pattern" is a package for natural language processing web mining, network analysis, and machine learning, with an aim on easy to use. It offers a mash-up of tools often used when tackling the Web as a corpus, which requires numerous independent toolkits attached together in a practical application. The package desires to be useful to both a scientific and a non-scientific audience. Parameters and Function names were chosen to make the commands self-explanatory.

English text is used almost everywhere. It would be best if system can understand and generate it automatically. However, natural language is not easy to understand. Nowadays, a lot of tools have been used to do natural language processing jobs. These tools give almost accurate result of analysis. In this paper we will discuss these tools like Core NLP, PRAAT and Apertium in detail.

2. Natural Language Processing

Natural language processing assigns to computer systems that attempt to understand, analyze or produce one or more human languages. The input might be text, keyboard input or spoken language. The task might be to translate to another language, to build a database or generate summaries. It is extremely difficult to define how we would get to know that a system actually "understands" language.

The principal difficulty in processing natural language is the inescapable ambiguity found at all levels of the problem. For example, all natural languages involve:

- Simple lexical ambiguity (e.g. "duck" can be a noun [the animal] or a verb [to avoid something thrown]).
- Structural or syntactic ambiguity (e.g. in "I saw the man with a telescope," the telescope might be used for the viewing or might be held by the man being observed).
- Semantic ambiguity (e.g. "go" as a verb has well over 10 distinct meanings in any dictionary).
- Pragmatic ambiguity (e.g. "Can you lift that rock?" may be a yes/no question or a request to lift the rock).

- Referential ambiguity (e.g. in "Jack met Sam at the station. He was feeling ill. . .," it is not clear who is ill, although the remainder of the sentence might suggest a preferred interpretation).

As computers play a larger role in the preparation, transmission, monitoring, acquisition, storage, analysis, and transformation of information, enabling them with the ability to generate and understand information expressed in natural languages becomes more necessary. Some tasks performed by humans cannot be automated without enabling computers with natural language processing efficiency, and these provide two major challenges to NLP systems:

1. Reading and writing text – This applied to tasks such as message routing, abstracting, summarizing, monitoring, and entering information in databases.
2. Translation - Computers should be capable to understand input in more than one language, able to provide output in more than one language, and able to translate between languages.

3. Pattern Modules

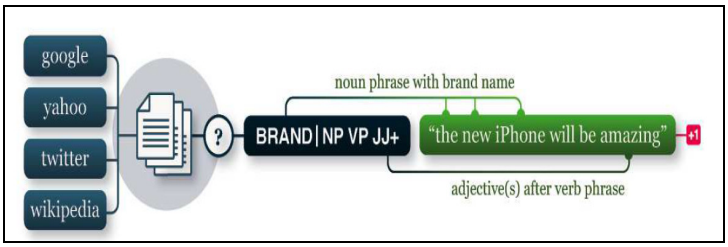

Figure 1: Pattern Modules

PATTERN is formulated in separate modules that can be attached together, as shown in Figure 1. For example, text from Wikipedia (pattern.web) is parsed for part-of-speech tags (pattern.en), queried by syntax and semantics (pattern.search), and then it is used to train a classifier (pattern.vector).

- pattern.web Tools for web data mining, using a download mechanism that supports caching, asynchronous requests and redirection. A Search Engine class provides a uniform API to multiple web services like Google, Yahoo!, Bing, Twitter, Wikipedia etc. The module includes an HTML parser, a PDF parser , a webmail interface and a web crawler.
- pattern.en Fast, regular expressions-based shallow parser for English (identifies nouns, verbs), using a finite state part-of-speech tagger. A parser with higher accuracy can be used. The module has a functions for singularization/pluralization conjugation, modality and sentiment analysis, Sentence class for parse tree traversal.
- pattern.nl it is a Lightweight application of pattern.en. Contributors are determined to read the developer documentation on how to support for other languages.

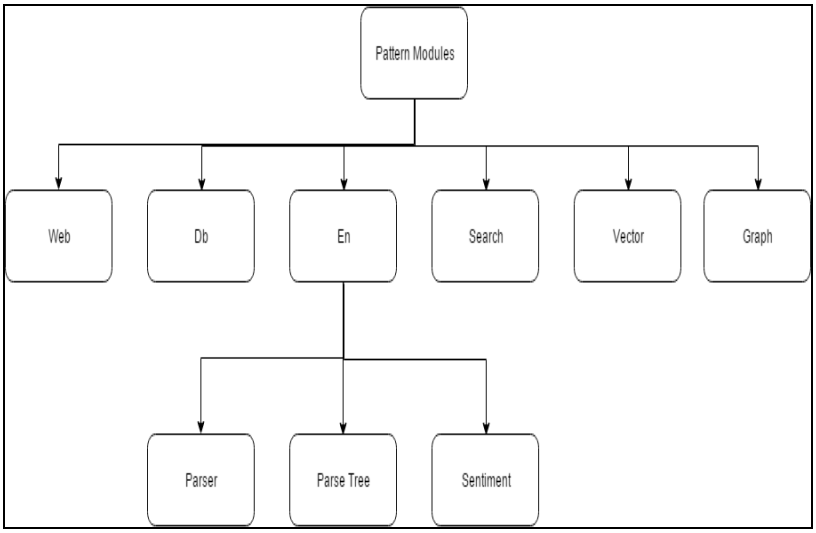

Figure 2: Flow of pattern.en

- pattern. search The algorithm uses an approach similar to regular expressions. Search queries include a collection of words, phrases, part-of-speech-tags to extract relevant information.

4. Natural Language Processing Tools

4.1. Core NLP

The design and development of Stanford Core NLP, a Java (or at least JVM-based) annotation pipeline framework, which contribute most of the common core natural language processing (NLP) steps, from tokenization to Co-reference resolution. While there are different good natural language analysis toolkits, Stanford Core NLP is one of the most used.

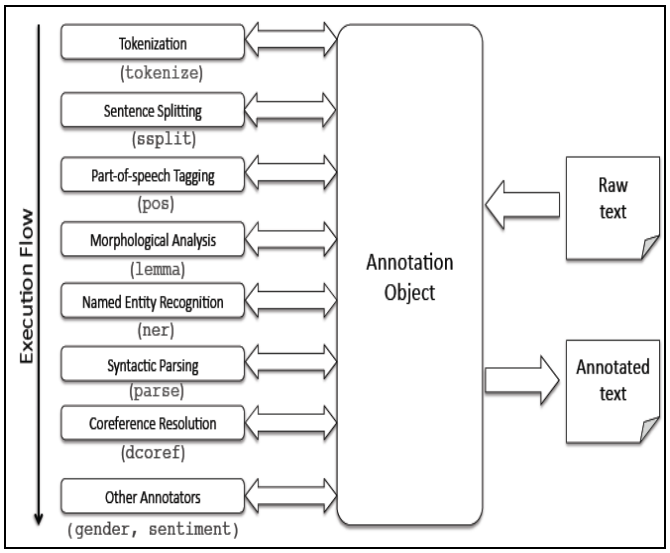

Figure 3 : Annotation Pipeline

Annotation pipeline system was initially designed for internal use. The initial version of the Overall system architecture: Raw text is put into an Annotation object and then a sequence of Annotators add information into an analysis pipeline. The final combined Annotation, containing all the analysis information added by the Annotators, can be output in XML or plain text forms.

The motivations were:

- To be able to quickly get linguistic annotations for a text without any pain.
- To cover variations across components behind a common API.
- To have a minimuml conceptual footprint, so the system is easy to learn and understand.
- To provide a lightweight framework, using plain Java objects.

4.2. Praat

PRAAT is a flexible tool for speech analysis. It offers a broad range of standard and non-standard procedures, including spectrographic analysis and neural networks. PRAAT has following functionalities:

1. Spectral analysis - This enables you to organize the spectrogram settings and extract information; the frequency value at the cursor position is specified on the left hand side of the panel in a red font.
2. Intensity analysis - This enables you to control the intensity signal settings and extract information; the intensity signal is shown in a yellow line and the value at the cursor position is specified on the right hand side of the panel in a bright green font.
3. Pitch analysis - This enables you to control the voice i.e. pitch signal settings and extract information; by default, the pitch signal is indicated by a blue solid line and the value at the cursor position is specified on the right hand side of the panel in a dark blue font.
4. Formant analysis - This enables you to control the formant settings and extract information; the formants are indicated by red dotted lines. The size of the window showing formant values can be set in the 'Formant settings' option.

4.3. Apertium

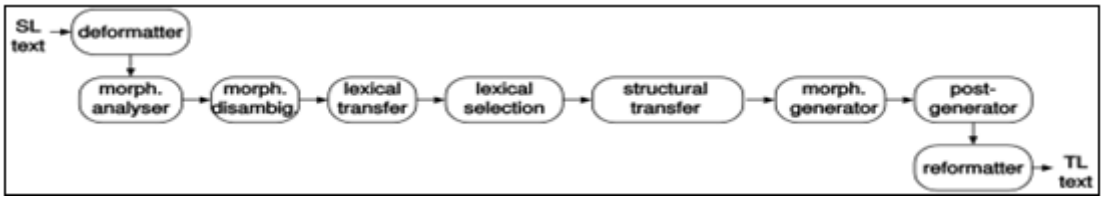

Figure 4: Apertium Flow

This is a step-by-step flow of how Apertium works.

The diagram shows the steps that Apertium takes to translate a language text that has to be translated into a target-language text.

1. Source language text is passed into Apertium for translation.
2. The deformatter removes formatting markup (HTML, RTF, etc) that should be kept but not to translate.
3. The morphological analyser segments the text and finds the segments in the language dictionaries, then returning base form and tags for all matches.
4. The morphological disambiguator resolves ambiguous segments, when there is more than one match, by choosing only one match. Apertium is working on installing more Constraint Grammar frameworks for its language pairs, enabling the imposition of more fine-grained constraints than would be otherwise possible.
5. Lexical transfer finds disambiguated source-language base words to find their resulting-language equivalents. Lexical selection chooses between alternative translations when the source text word has another meaning.
6. Structural transfer is a XML format that allows writing complex structural transfer rules, can consist of a one-step transfer or a three-step transfer module. It points to grammatical differences between the source language and resulting language. It then modifies or reorders chunks in order to produce a grammatical translation in the resulting language.
7. The morphological generator uses the tags to bring the correct resulting language surface form. The morphological generator is a morphological transducer, like the morphological analyzer. A morphological transducer both generates and analyses forms.
8. The post-generator makes any required orthographic changes due to the contact of words.
9. The reformatter replaces formatting markup that was detached by the deformatter in the first step.
10. Apertium gives the target-language translation.

5. Working with Ambiguous Sentence

For an ambiguous sentence, above mentioned NLP tools works as follows:
Example – "Look at the dog with one eye."

5.1. Core NLP

"Look at the dog with one eye."

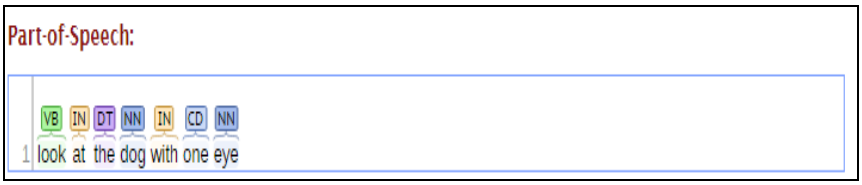

Figure 5: Core NLP Part of Speech

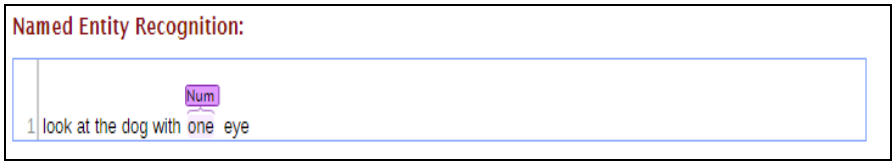

Figure 6: Core NLP Named Entity Recognition

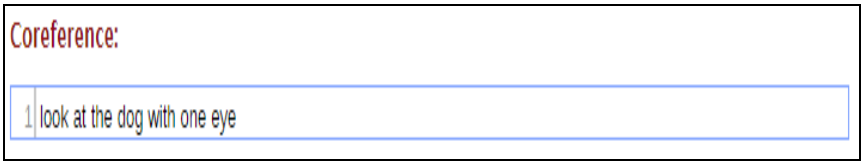

Figure 7: Core NLP Coreference

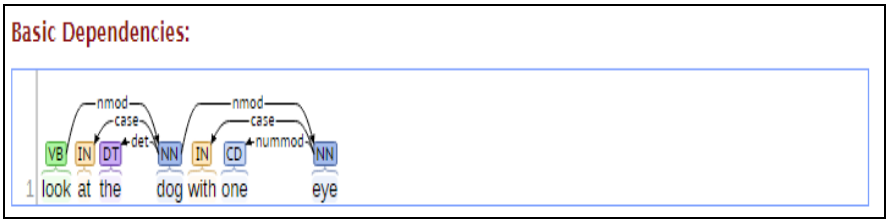

Figure 8: Core NLP Basic Dependencies

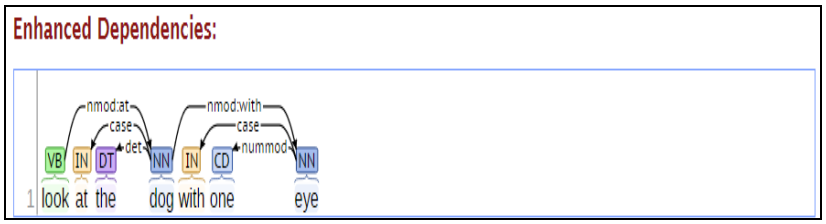

Figure 9: Core NLP Enhanced Dependencies

VB – verb, IN – inter-junction, DT – determiner, NN – noun.

- Nmod - As can be seen in figure, Nmod labels the relations between the three content words of- look, dog and eye, whereas the preposition or the possessive marker is a case depending on its complement. In general, Nmod expresses some form of oblique or adjunct relation which can be further specified by the case or be morphologically marked.
- Det – A determiner is the relation between the head of an NP and its determiner.
  As we can see from above sentence that determiner relation is between "the" and "dog" where "dog" is the head of NP and the determiner is "the".
- NN- A noun compound modifier of an NP is any noun that serves to modify the head noun. We can see from above sentence "eye" is our noun compound modifier which serves to modify the head noun "dog".
- NumMode - A numeric modifier of a noun is any number phrase that serves to modify the meaning of the noun with a quantity. In this example noun is "eye" and quantity is "one".

5.2. Apertium

With the same example above, we can translate this sentence from current language to some other language. Here we are translating English to Spanish language.

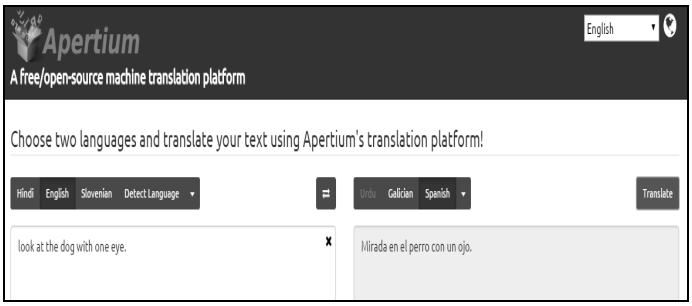

Figure 10: Ambiguous Sentence Translation in Apertium

5.3. PRAAT

PRAAT is a special tool for sound analysis. We take the same sentence in both male and female voice. We will perform analysis on both voices to see different results on the basis of four functionalities, Spectral analysis, Pitch analysis, Intensity analysis and Formant analysis.

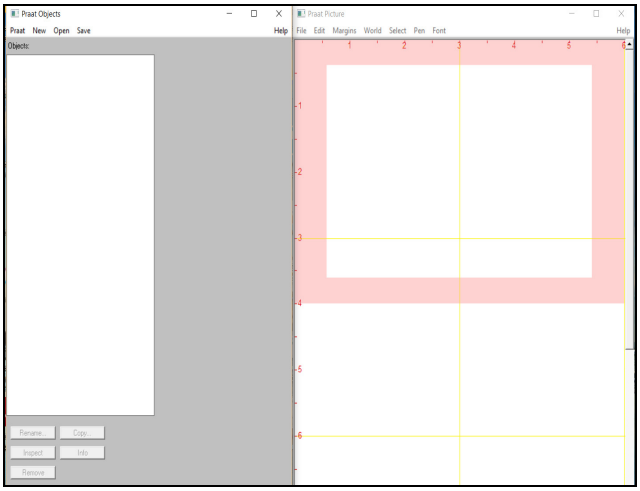

Figure 11: PRAAT Working Window

Initially we will see the above window. We have to record male/female voice and perform analysis.

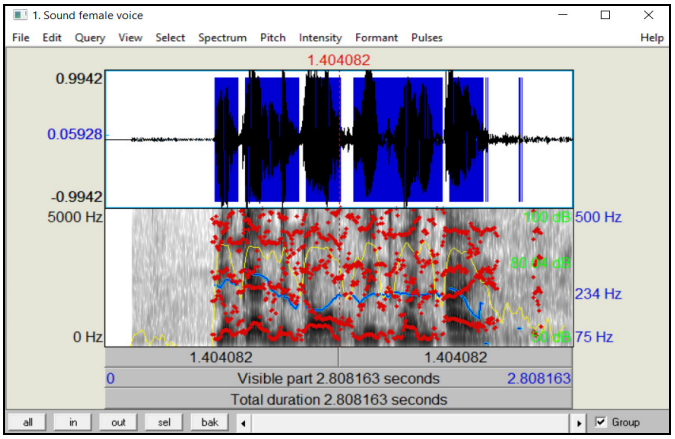

Figure 12: Female Voice Sample

In above figure we can see the analysis for female voice.

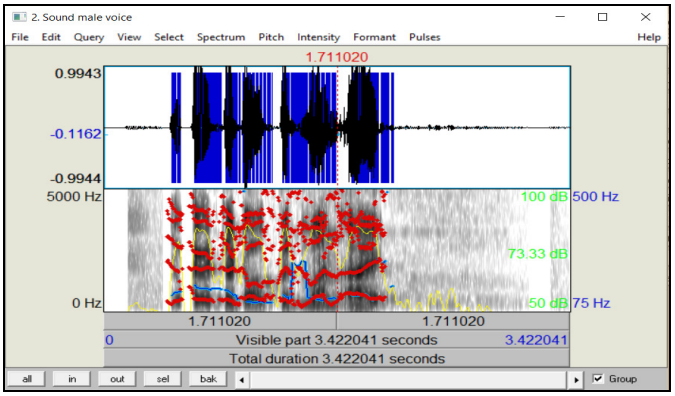

Figure 13: Male Voice Sample

In above figure we can see the analysis for male voice. In both figures we have combined the analysis for all four functionalities which come under PRAAT tool.

- Spectral analysis - Grey background with the frequency from 0HZ to 5000HZ specifies the spectral analysis result.
- Pitch analysis - Blue waves in both figures represents the Pitch analysis for both male and female voice.
- Formant analysis - Red waves in both figures represent the Formant analysis.
- Intensity analysis - Yellow waves in both figures represent the Formant analysis whose sound level ranges from 50db to 100db.

6. Conclusion

In this paper we can conclude that natural language processing tools can be used for the analysis of any sentence or any voice module. Here we have used an ambiguous sentence for the analysis using different tools. Results of all these tools are different. Core NLP gives the result on the basis of the dependencies of the words and differentiating them on the basis of nouns, adjectives etc. Whereas Apertium is just a translation tool which helps in translating sentence to another language. There are official 42 pairs of released versions of languages. PRAAT gives the result on the basis of voice input and here we have compared two voices for male and female.

Acknowledgment

We would like to thank Prof. Dipali Baviskar (Software Architecture and Design) for her excellent support.

7. References

i. T. De Smedt, “Pattern for Python,” vol. 13, pp. 2063–2067, 2012.
ii. T. Publishers and P. Papers, “It is extremely difficult to define how we would ever know that a system actually ‘understands’ language.,” pp. 1218–1222.
iii. P. Boersma, “The use of Praat in corpus research.”
iv. B. P. Boersma and V. Van Heuven, “Speak and unSpeak with P RAAT,” vol. 5, no. 9, pp. 341–347, 2001.

v. P. Objects, "Speech Analysis using PRAAT."
vi. W. Styler, "Using Praat for Linguistic Research," 2013.
vii. P. Van Lieshout and D. Ph, "PRAAT Tutorial," vol. 2003, pp. 1–27, 2003.
viii. C. D. Manning, J. Bauer, J. Finkel, and S. J. Bethard, "The Stanford CoreNLP Natural Language Processing Toolkit."
ix. G. Kundu and D. Roth, "Adapting Text instead of the Model : An Open Domain Approach."
x. M. Lease, "Natural Language Processing for Information Retrieval : the time is ripe (again)."
xi. V. Arnaoudova, S. Haiduc, A. Marcus, and G. Antoniol, "The Use of Text Retrieval and Natural Language Processing in Software Engineering," pp. 949–950, 2015.
xii. G. G. Chowdhury, "Natural Language Processing," no. 1999, pp. 1–38, 1984.
xiii. R. Weiscbedel, J. Carbonell, W. Lehnert, M. Marcus, and R. Perrault, "White Paper on Natural Language Processing," pp. 481–493.
xiv. E. Cambria, "Review Article," no. May, pp. 48–57, 2014.
xv. S. Bird and E. Loper, "NLTK : The Natural Language Toolkit."
xvi. E. Loper and S. Bird, "NLTK : The Natural Language Toolkit," pp. 63–70.
xvii. S. Bird, "NLTK : The Natural Language Toolkit," no. July, pp. 69–72, 2006.
xviii. P. Das, K. K. Baruah, A. Hannan, and S. K. Sarma, "International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS) Rule Based Machine Translation for Assamese-English Using Apertium," pp. 401–406, 2014.
xix. M. D. Brandt, H. Loftsson, and F. M. Tyers, "Apertium-IceNLP : A rule-based Icelandic to English machine translation system," 2009.