Syntax Preprocessing in Cyberlaw Web Knowledge Base Construction

Wilson Won, Goh Ong Sing, Mokhtar Mohd Yusof
Faculty of Information and Communication Technology,
Kolej Universiti Teknikal Kebangsaan Malaysia, Ayer Keroh,
75450, Melaka, Malaysia
Email: wyswilson@yahoo.com, goh@kutkm.edu.my, drmokhtar@kutkm.edu.my

Abstract
This paper describes a functional model for preprocessing and preparing natural language text from online news documents for constructing knowledge base. The model involves five phases of processing which includes tokenizing free text into segments, identifying syntactic categories (i.e. part-of-speech and phrasal categories) and grammatical relations (e.g. subject, object and etc), tagging noun phrases with named-entity identifiers, analyzing morphological root of verb phrases and resolving pronouns.

1 The Motivation

The functional approach towards syntactic preprocessing in this paper is part of a Cyberlaw knowledge base construction from online news documents [1]. Our focus is directed to the Cyberlaw domain as such effort to extract and transform news related to legal activities in the Information Technology and Internet world in order to create and maintain a knowledge base has never been attempted before. Moreover, as more parts of our life become acquainted to technology, this knowledge base will be a valuable resource for legal practitioners, legal students, academicians and others alike.

Our belief that a more precise knowledge representation cannot be achieved without attempting a deeper understanding of at least parts of the text has also prompted us to embark in the study of approaches in the syntax level. As a subfield of linguistics, syntax deals with the patterned relations that govern the way words in a sentence come together. It concerns how different words which are categorized as nouns, adjectives, verbs and etc are combined into clauses which in turn combine into sentences.

In order to scrutinize how the syntactic structure of a sentence can be computed, one must consider two things namely the grammar, which is a formal specification of the structures permissible in the language and the parsing technique, which is the method of analyzing a sentence to establish its structure according to the grammar.

In the terms of knowledge base construction, syntactic preprocessing lays the ground work required for transforming free text meant for human conception to a machine conceivable format. It involves procedures like segmenting multi-paragraph text, tagging each word with syntactic categories and grammatical relationships, recognizing named-entities, analyzing words for their morphological roots, disambiguating words and etc. And despite these responsibilities, there is no complete mechanism for all these tasks that provides definite solutions of 100% accuracy. But nonetheless, many researchers have produced algorithms and formalisms that provide very promising results as we will see later. These physical works lay a solid foundation for us to outline the various functional requirements and considerations when building syntactic preprocessor tailored for knowledge base construction.
2 Related Research

The work in syntactic preprocessing can be broadly grouped into two solutions. An integrated solution like what has been carried out by LaSIE (Large Scale Information Extraction) [2] and FASTUS (Finite State Automata-based Text Understanding System) [3]. The other class of approach is independent studies like Link Grammar [4], Minipar [5], Brill tagger [6], MUSE [7, 11] and etc which deals with different sections of syntactic preprocessing.

LaSIE is by far the most complete system offering an integrated solution to syntactic preprocessing as part of its information extraction responsibility. The processing consists of splitting and tokenizing the input, part-of-speech tagging morphological analysis and named-entity tagging. FASTUS, a somewhat misleading acronym is actually another information extraction system. The system consists of a cascade of transducers, namely tokenizer, preprocessor, phrase parser and phrase combiner.

Link grammar is a relatively new formal grammatical system for parsing English sentences. The formalism is lexical and does not make use of constituents and categories. Minipar is a principle-based English parser. It represents its grammar as a network where nodes represent grammatical categories and links represent types of syntactic relations.

Brill tagger is a rule-based part-of-speech tagger which automatically acquires its rules as opposed to conventional rule-based tagger and thus, performs as well as those based upon probabilistic models like Markov model.

MUSE is a multi-purpose named entity recognition system consisting of three main processing phases. A tokenizer which splits text into atomic level producing numbers, punctuation, symbols and words, a gazetteer which consists of lists of proper nouns and triggers, and a grammar which constitutes of manually produced rules describing patterns to match.

3 Functional Overview

The processing resources required for outlining an integrated approach in syntax preprocessing varied depending on the requirements of the system to which the approach is attached to. The choice between a part-of-speech tagger and sentence parser is very much dependant on the type of output required by the next processing phase. For example, if the system does not require grammatical categories in the output, then using a sentence parser is just redundant.

The input to the syntactic preprocessor is an online news document which may consist of several paragraphs. The functional model of the preprocessing phases required is shown in Figure 2 as part of a knowledge base construction system in Figure 1.
Fig. 1. Architecture of the Cyberlaw knowledge base system

Fig. 2. Functional view of syntactic preprocessing module

Some of the phases in Figure 2 can be shuffled and even removed depending on the functions of the remaining phases. To put it in other words, the choice of algorithm for a particular phase will determine the relevancy of other phases. This property is called dependant optionality where a particular phase is considered as redundant if the output it provides is already contained in the output of other phases. For example, the morphology analyzer can be put aside if the sentence parser also performs morphology analysis implicitly as part of its function.

3.1 Tokenizer

Tokenization serves two purposes. Firstly, each document, which consists of multiple paragraphs are segmented to determine the boundaries which will retain the meaning of the entire document and yet, minimized processing requirements for the remaining phases. Token boundaries are identified and each token is assigned an identifier in the sequence that makes up the document. These identifiers are preserved throughout the phases. Secondly, since the sentence parser expects one sentence per line and expects each token to be separated by white space, the token stream is changed into this format before it is fed to the parser.

3.2 Sentence Parser

Another task of computational linguistics is sentence parsing, whose main aim is to conceptualize the syntactic structure of a sentence. Two choices of parser were in hand namely Link Grammar and Minipar. We installed the two parsers and performed an extrinsic evaluation as described by Bangalore et al. [8]. Extrinsic evaluation is usually used as an indirect method for comparing parsing systems even if they produce different representations for their outputs.
as long as the output can be converted into a form usable by the system in which the parser is
embedded. We will evaluate it in terms of: performance, accuracy and quality of output.

Up to the time of publishing this paper, we have 322 uncategorized news web documents
extracted and wrapped from ZDNet News Archive [9]. We run the two parsers on the first
sentence only of all the 322 documents, where each sentence has an average of 192 words. At
this point, we are not interested with the composition or syntactic structure of these sen-
tences. It took Link Grammar an average of 10 secs to parse a sentence and Minipar 4 secs.

<table>
<thead>
<tr>
<th>Number of news web documents</th>
<th>Link Grammar</th>
<th>Minipar</th>
</tr>
</thead>
<tbody>
<tr>
<td>322</td>
<td>3366 seconds</td>
<td>1375 seconds</td>
</tr>
<tr>
<td>1</td>
<td>10 seconds</td>
<td>4 seconds</td>
</tr>
</tbody>
</table>

**Fig. 3.** Processing time required for each parser respectively

As for the accuracy test, we will only select documents categorized as Cyberlaw (the cate-
gorizing task is out of the scope of this paper) and run the two parsers on them. There were
33 online Cyberlaw news documents and again, only the first sentence is used. We will look
at one sentence as example:

Oracle has asked a California state judge to throw out an un-
fair business practices lawsuit filed by PeopleSoft, as the
plaintiff seeks to derail a hostile takeover bid by its rival.

**Fig. 4.** Sentence A for sentence parsing

(S Oracle has asked
  (S (NP (NP a California state judge)
    (SBAR (WHNP to)
      ...

**Fig. 5.** Part of the constituents output of Link Grammar on sentence A

It is obvious from the Link Grammar parse output above that the first three words have
been left untagged. Figure 5 shows the constituents output of the parse and Figure 6 depicts
the linkage diagram showing the part-of-speech of each word, if there is any. The word Ora-
cle, has and asked have failed to be tagged due to the null-links feature of the parser.

**Fig. 6.** Part of the linkage output of Link Grammar on sentence A
Minipar perfectly parse every words and the constituents display of the Link Grammar can be easily produce from the grammatical relationships listed in the fifth column.

Next, considerations will be given to the suitability of the output of the parse. Here, we are actually looking at the ease for which the output can be translated to vectors required by the remaining processing phases. From the way Link Grammar produces its output, we can assume that they are mainly meant for visual scrutinization as shown in both Figure 5 and 6. In order to extract the syntactic categories and token offset, it requires around 300 lines of optimized Perl codes in comparison to only 90 lines using Minipar parser. In addition to that, we can even obtain the morphological roots of verbs and nouns from Minipar output, thus eliminating the need for a morphology analyzer. Also, Link Grammar parses sentences in such a way that it generates multiple instances of output which correspond to the possible linkages produced as shown in Figure 8, which generates 20 possible linkages based on sentence A in Figure 4. This implies that there is no single optimized output to work on.

Nonetheless, the choice of the parser will not condition the design of other processing procedures. These evaluations merely provide comparative information to select the suitable parser and do not necessarily identify the best parser.

### 3.3 Named-Entity Tagger

Tagging named-entity is considered as one of the most vital component of computational linguistics. Named-entity tagger identifies proper nouns in text and categorized them into one of the categories of interest. Handling named-entity may appear undemanding but it provides the platform necessary for co-referencing and ontological information. With co-referencing, we can track *Mr. Liaw* to any pronoun that refers to it and vice versa. We can also obtain ontological information like *Mr. Joe Liaw* is an instance of *Person*. Named-entity tagging, like other parts of computational linguistics, does come with its own problems:

- Variations of named-entity: for example, *Joe Liaw, Mr. Liaw* or *Joe* may refer to the same person.
- Ambiguity of named-entity types: for example, *June* can be month or person’s name.
- Ambiguity with common words: for example, is bill a person’s name or a common noun.

There are three approaches for named-entity tagging namely list-lookup approach, shallow parsing approach and shallow parsing approach with context. In list-lookup approach, the tagger recognises only entities stored in its lists known as gazetteers. It is simple and fast but cannot deal with name variants and resolve ambiguity. Shallow parsing uses triggers to identify named-entities. For example, **ABC Corporation** will trigger the **Corporation** word and from that, identifying it as an organization even though we do not have **ABC Corporation** in our gazetteers. The triggering approach also has its problem, being unable to handle ambiguous semantic, structures and capitalized words. As an example, the tagger will not be able to differentiate the proper noun **June** in the phrase **the lawsuit was filed by June** and **the lawsuit was filed in June**. If only the tagger has contextual information about the phrases, then it would be able to know that **June** in the first phrase is a **person()** and **date()** in the second phrase. Another example, the phrase **Microsoft’s Japan unit** will be ambiguous without context information because based on the triggers, the phrase can be either **organization()** or **location()**. With that, the third approach employs context-based rules to assist in solving ambiguous cases. Handcrafted rules like by **<PERSON|ORGANIZATION>**, in **<DATE|LOCATION>** and **<ORGANIZATION>’s <LOCATION> unit** will solve the above problems. But nonetheless, these context patterns can be costly in terms of manual construction, maintenance and portability.

In this paper, we will propose a new variation of named-entity tagging algorithm using finite-state automaton and weighted gazetteers with grammar constraints. Variations of named-entity tagger are distinguishable by three components namely named-entity set, gazetteers and its algorithm.

The first named-entity set had seven types namely organization, location, person, date, time, money and percent expressions. The number of entities was limited then because they were targeted for the business domain. As our focus now is directed to the Cyberlaw domain, new entities like **court, judge, allegation** and etc need to be created. Gazetteers in most taggers are made up of lists of names and triggers as followed:

- **Proper name lists**
  - Organization names: a cleaned up list taken from Fortune 500 lists
  - Human names: obtained from Oxford Dictionary Advanced Learner’s Dictionary
  - Location names: names of major cities in the world, states of countries and countries compiled from the Internet

- **Trigger lists**
  - Currency units such as ringgit, dollar and etc
  - Titles such as Mr., Ms., Dr., and etc
  - Company designator such as Co., Ltd., and etc
  - Organization such as Association, Group and etc
  - Location such as Mount., Lake, Sea and etc
  - Government institutions such as Agency, Ministry and etc

In our gazetteers, each entry has additional information like weight, sub-category and the acceptable preceding/foregoing grammatical relations in addition to the triggering information, category and entity name. For example, returning to the noun phrase **Microsoft’s Japan**
unit, this triggering ambiguity could be solved by just using the weighting mechanism without the need for any hand-crafted rules.

![Fig. 9. Finite-state automaton for noun phrases extraction](image)

The algorithm adopted by our named-entity tagger employs finite-state automaton. The sentence to be named-entity tagged is first parsed for syntactic categories and grammatical relationships using the sentence parser of choice, Minipar. The output of parsing is then fed through the FSA as shown in Figure 9 to extract noun phrases. By inferring named-entity extraction criteria by Collins & Singer [10], all named-entity are subset of noun phrases and not merely proper nouns. This is an important point to note because in Cyberlaw domain, phrase like antitrust law might not be identifiable as a proper noun but still, it has to be tagged as named-entity.

![Fig. 10. Sentence parsing output using Minipar for lawsuit was filed by June](image)

![Fig. 11. Sentence parsing output using Minipar for lawsuit was filed in June](image)

![Fig. 12. Noun phrases accepted by the FSA](image)

Feeding the output of Minipar for the two sentences lawsuit was filed in June and lawsuit was filed by June into the FSA will produce accepted noun phrases like in Figure 12. Using these noun phrases, grammatical relationship of the word in and by as shown in Figure 10 and 11, additional information in the gazetteers and the algorithm in Figure 13, the tagger can decide that the word June in the sentence lawsuit was file by June can only exists in the context of Person() because it was preceded by a word with by-subj as the grammatical relationship.
noun_phrase accepted by FSA
IF noun_phrase exists in gazetteers THEN
    IF preceding_grammar_relation allowed by gazetteers THEN
        get category of noun_phrase
    END IF
ELSE tokenized noun_phrase in trigger THEN
    IF preceding_grammar_relation allowed by gazetteers AND token_weight is max THEN
        get category of token
    END IF
    Use category of token as category of entire noun_phrase
END IF

Fig. 13. Algorithm for named-entity recognition using weighted gazetteers with grammar constraints

Fig. 14. Flow chart representation of the named-entity recognizer

The use of FSA and weighted gazetteers with grammar constraints eliminates the need to manually create and maintain a separate context patterns and can be ported to any domain because context information in the gazetteers is based on grammatical relationships which are general in English sentences and not domain specific.

3.4 Morphology analyzer

After words have been part-of-speech tagged, all nouns and verbs are passed to a morphological analyzer to obtain its root form. There are many different approaches of performing this. One of them is the use of regular expression rules to perform analysis in conjunction with a list rules and irregular exceptions derived from lexicon like WordNet. As morphological analysis has been performed as part of Minipar, thus we do not wish to reinvent the wheel.
3.5 Pronoun resolver

Pronoun resolution involves finding the antecedent of a pronoun like it, he, she and etc. There are several approaches widely practiced which involve the use of syntactic categories like NP, N, VP, V and etc [12]. This paper proposed an algorithm that performs a deeper analysis on sentences through the use of grammatical categories like subject, object and etc. The algorithm consists of segments that handle different classes of sentences. Part of the algorithm is shown in the figure below. This segment of the algorithm handles sentences like “The European Commission has told Oracle that it objects ...” and “Japan's Fujitsu announced on Wednesday that it has initiated ...”.

The algorithm uses the notion of a posgram, that is the combination of both part-of-speech and grammatical relation in the form of <POS:GRAMMAR_REL>.

IF current_token is a pronoun THEN
    IF pronoun's posgram is N:s AND previous_token’s posgram is COMP:c THEN
        find the last token that occur before pronoun that has N:s as posgram
        the last token is the antecedent of the pronoun
    END IF
    OTHER ALGORITHM SEGMENTS
END IF

Fig. 15. An algorithm segment of the pronoun-resolver

The algorithm segment simply states that the antecedent for a pronoun is the last token that occurs before it that has N:s as its posgram given the pronoun’s posgram is also N:s and the previous token’s posgram is COMP:c. That rule segment alone is capable of handling 18% of the text in collection.

4 Output Requirements

Regardless of how the processing phases are placed together and which algorithms are used, the final output that is produced must conform to certain requirements set by the system so that they are compatible with the knowledge base system to which it is attached. In our case, the requirements of the semantic parsing module in the Cyberlaw knowledge base system [1] as shown in Figure 1 is a vector consisting of word tokens, part-of-speech information, grammatical relationship, root form of words, noun phrases and their corresponding named-entity tag and word offset information with respect to current sentence.

Microsoft's Japan unit is being investigated by the country's Fair Trade Commission on suspicion of violating antitrust laws.

Fig. 16. Sentence C for syntactic preprocessing

For example, given the above sentence, the syntactic preprocessor should produce a vector of information as shown below.
Fig. 17. Output of syntactic preprocessing for sentence C that meets system requirements

The first column holds the offset of its corresponding token with respect to the origin of the sentence. The offset information is important so that the tokens’ position is recoverable. The second and third column contains the token and its corresponding root form, if there is any, respectively. The fourth column holds the part-of-speech or grammatical category and last but not least, the fifth column contains grammatical relationships between tokens.

The bottom part of the output consists of noun phrases or proper nouns that are tagged with named-entity categories supported by the Cyberlaw knowledge base system.

5 Conclusion

This paper has presented a unified functional model of the different syntactic processing requirements in preparing natural language text which will later be transformed to knowledge representation. This model segregate the different tasks required in syntactic processing into independent modules for study and implementation. Several main contributions arise from this research:

- Introduction of three new variation of algorithms:
  - non-deterministic finite state automata for recognizing noun phrases.
  - named-entity tagging algorithm using weighted gazetteers with grammar constraints
  - multi-segment algorithm using deeper syntactic analysis for pronoun resolution

- Definition of a model for syntax processing by segregating the tasks into well-defined modules.

The model in this paper is presented as part of a larger Cyberlaw knowledge base system to highlight and demonstrate the functions of the model. But nonetheless, the model can be em-
ployed in any knowledge base system that requires the preparation of natural language text for transformation into knowledge representation.

References


