Mapping Text to Knowledge using Natural Language Processing

A Major Qualifying Project
Submitted to the faculty of
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Bachelor of Science

Submitted by:

Andreea Bodnari

Submitted to:

Project Advisor, Professor Carolina Ruiz
Project Co-Advisor, Professor Gary Pollice

Date: 15 March 2010

Abstract

The goal of this project was to design and implement a system that analyzes text corpora. This system uses natural language processing techniques to extract knowledge from written text and represents this knowledge as a network. The system displays this network to the user and allows the user to interactively explore the network. The accuracy of the knowledge extraction process and the overall performance of the developed system were assessed. Possible applications are in social networks and text simplification.

Acknowledgement

The author would like to first thank professors Carolina Ruiz and Gary Pollice for their time, guidance and support while advising this project. A special recognition is expressed to Myank Jain, Mike Tidd and Yi Wang.

To my beloved family,
who has taught me to have faith, live and work.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Artificial Intelligence (AI) is the design and study of computer programs that behave intelligently [1]. As a subfield of artificial intelligence, Natural Language Processing (NLP) aims at giving machines the ability to read and understand human languages. Researchers hope that a sufficiently powerful natural language processing system would be able to acquire knowledge on its own by reading the existing text available over the Internet and build logical connection among concepts based on available explanatory information. Natural Language Processing toolkits are available from academic and industry research groups, but each toolkit has a restrictive implementation and does not allow for integration with other systems/toolkits.

The objective of this project was to implement an extensible framework for natural language processing and knowledge visualization. The system would handle any document format (e.g., Microsoft Word 2003/2007, PDF, HTML web pages), extract textual knowledge, and exhibit it to the user in a network-like display. Functionality provided by already existing NLP libraries (e.g., Stanford [2] and LingPipe [3]) was employed in a concurrent system design. The performance of the system was improved by parallel processing of tasks.

The developed system consists of an extensible software framework and a user-friendly interface. The software framework can be integrated with existing NLP Java libraries and allows for addition of different data visualization techniques.

2 Background

2.1 Communication

If you wish to converse with me, define your terms.
~ Voltaire[1]

Communication represents the exchange of information between at least two agents through a common repertoire of signs and semiotic rules. Communication is classified as verbal, non-verbal, written, and visual, depending on the means used for information exchange.

Humans have developed a superior communication tool, natural language, which is a complex system of messages based on an alphabet, through which an unbounded number of qualitatively different messages can be conveyed [4]. The speech act (the act of producing language) is motivated by the need to obtain and share information about the world or personal experiences. Information is sent via a medium, received by an agent, decoded and understood. Understanding the speech act is problematic due to potential ambiguity of the sent information. In order to avoid ambiguity, communication is governed by three major semiotic rules: syntactic (formal properties of signs and symbols), pragmatic (concerned with the relations between signs and expressions and their users), and semantic (study the relationships between signs and symbols and what they represent).

2.1.1 Formal Language

In contrast to natural languages, which have evolved freely, the fields of computer science and linguistics have introduced the concept of formal languages. A formal language is defined as a set of strings[2] made of terminal symbols. The set from which the symbols are extracted is called the alphabet over which the formal language is defined.

The combinations of alphabet symbols accepted by a language's syntax are specified through a formal grammar. Grammars are used as both language generator and language recognizer – computing the probability of a string belonging to a certain language through a process called parsing. As defined by Noam Chomsky in the 1950s [5, 6], a grammar consists of a finite set N of nonterminal symbols, a finite set $\sum$ of terminal symbols that is disjoint from set N, a finite set P of production rules,

[1] Voltaire (François-Marie Arouet, 1694-1778): French Enlightenment writer, essayist and philosopher; one of the greatest of all French authors.
[2] The linguistics equivalent for strings is words, where the alphabet characters represent the terminal symbols.

and a distinguished symbol S ∈ N that is the start symbol. Most grammar rules are designed around the idea of phrase structure, according to which strings are composed of substrings called phrases. For example in linguistics, the string "The train is late." consists of a noun phrase (NP- "The train") and a verb phrase (VP- "is late"), both generically addresses as nonterminal symbols.

Grammars can be classified by the set of languages they can represent[3]. Noam Chomsky [6] describes four hierarchical classes of grammatical formalisms that differ in terms of the production rules used (see Table 1).

Table 1: Classification of grammatical formalism based on generative capacity

Grammar Type	Description	Production Rule	Example
Recursively enumerable grammar	Unrestricted rules	LHS, RHS: Any number terminals and non terminals	A B -> C
Context-sensitive grammars	Any production rule can be surrounded by a context of symbols	LHS, RHS: Same number of symbols	A-> B A B-> C D
Context-free grammar (CFG)	CFG are commonly used for natural language processing and programming language grammar [4]	LHS: one single nonterminal symbol RHS: any combination of terminal and non-terminal symbols	A -> AA
Regular grammar	The most restrictive grammars.	LHS: single nonterminal RHS: terminal optionally followed by a nonterminal	B-> aB

2.2 Natural Language Processing

Natural Language Processing (NLP) is an AI sub-discipline, concerned with the computerized investigation and evaluation of claims about human language. The history of NLP dates back to the late 1940s, when machine translation (MT) was developed as the first computer-based application related to natural language [7]. The initial MT systems assumed that the only difference between languages is represented by vocabularies and a predefined word order, thus ignoring the lexical ambiguity inherent in natural language. This issue was addressed by Noam Chomsky's in his publication *Syntactic Structure* [6]. Chomsky introduced the idea of transformational grammar, a theory of how grammatical knowledge is represented and processed in the brain. Transformational grammar was defined through two main features [8]:

[3] Property addresses in literature as generative capacity.

1. A sentence has two forms of representation: ‘deep structure’- an underlying, more abstract form comprising the semantics of the sentence, and 'surface structure'- the actual form of the sentence produced. ‘Deep structure’ is represented in the form of a hierarchical tree diagram, or "phrase structure tree," depicting the abstract grammatical relationships between the words and phrases within a sentence. ‘Deep structure’ is assembled using a lexicon, and a series of transformations converts it into the ‘surface structure’[4].

   For example, the occurrence of a girl running down the stairs can be represented at a semantic level by three elements, GIRL, STAIRS, RUNNING, together with logical connectors showing which is the verb, subject and object. These elements are transformed at a spoken level to convey verb tense (“is running”), temporal condition (“now”) and spatial location (“down”). The result is a sentence of the form “The girl is now running down the stairs”, which can be enhanced by phonetic articulation (e.g., voice tone, intonation, etc).

2. A system of formal rules specifying how deep structures are to be transformed into surface structures

The NLP area has branched into on two main subfields, basic language processing and language generation [9]. The first of these refers to the analysis of language for the purpose of producing a meaningful computer representation (equivalent to the role of reader/listener), while the latter refers to the production of language from a computer representation (equivalent to the role of writer/speaker). NLP processing, which we will focus on throughout this project, consists of a four steps design: speech recognition, syntactic analysis, semantic interpretation, and pragmatic analysis.

Figure 1 presents the processes involved in communication and their equivalency to subfields of NLP [4] .

[4] Surface structure is converted into a phonetic form using the rules of phonology of the selected language.

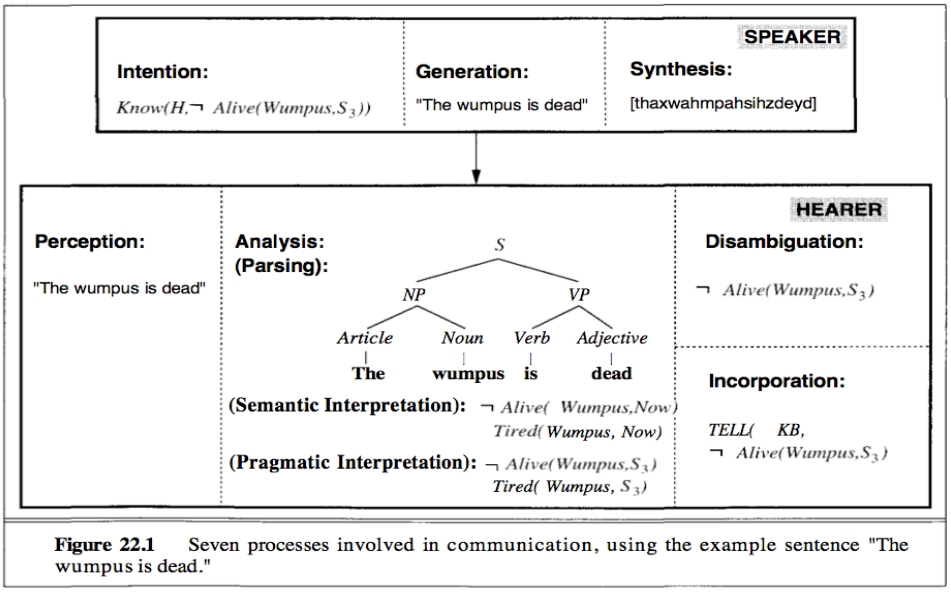

Figure 22.1 Seven processes involved in communication, using the example sentence "The wumpus is dead."

Figure 1 Seven processes involved in communication, using sample sentence "The wumpus is dead" [4]

Speech recognition represents the foundation of NLP understanding. Similar to the hearer having to percept the presence of a communication flow, the computer has to identify the given input as either written or spoken speech (text or verbal). This project is concerned with written speech recognition only. Text recognition can be reduced to the identification of tokens (words), which are further validated within a specific language. At a further level the identified speech is interpreted in order to extract the factual knowledge.

The main NLP tasks Syntactic Analysis Section 2.2.1, Semantic Interpretation Section 2.2.2, and Pragmatic Analysis Section 2.2.3, are further explained with reference to the model sentence “The girl is now running down the stairs”.

2.2.1 Syntactic Analysis (Parsing)

Sentence analysis is performed in terms of its syntactic composition. Syntactic interpretation is performed as a two-step process by a lexical tokenizer and a lexical parser.

The lexical tokenizer divides a given content into tokens (formally referred to as lexema) and categorizes them according to their function inside the sentence. There are potential issues that the tokenizer has to overcome. One class of issues is the morphological variations a token can present:

1) Inflectional morphology: a token changes its form and gains additional meaning (e.g., “The girl is running” is transformed into “The girls are running”);

2) Derivational morphology: a token changes meaning through derivation from initial form (e.g., “down” becomes “downer”);

3) Compounding: a token changes meaning by being compounded with another token (e.g., “girl” is compounded with “friend” and results in “girlfriend”).

The tokenizer detects variations and performs a dictionary look-up to identify the meaning of the word, while also handling error-recovery. Thus errors introduced unintentionally by the speaker are filtered through error-spelling correction, word syntactic class guessing (e.g., “smarply” can be classified as an adverb due to the “-le” suffix), and specialized formats detection (e.g., SSN: “ddd-dd-dddd”, date: “mm/dd/yyyy”, time: ”hh:mm:ss”).

The lexical parser combines the identified tokens into a structural form defined within a chosen formal grammar. The structural form is referred to as a parse tree, where the interior nodes represent the function of a given toke inside the sentence, and the leaf nodes represent the actual tokens.

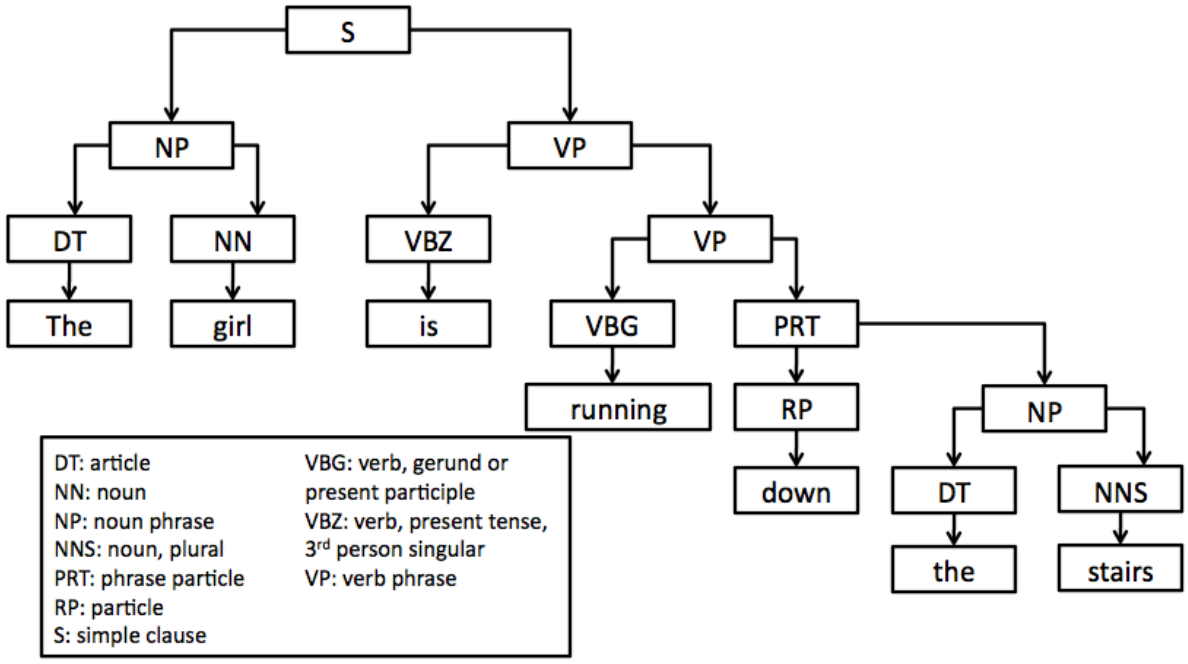

Figure 2 Sample parse tree using the test sentence "The girl is running down the stairs."

2.2.2 Semantic Interpretation

Every token has a context-independent aspect (referred to as the meaning of a token) and a context-dependent aspect (referred to as the usage of a token). Semantic interpretation is the process of combining context-independent aspects to get all possible interpretations.
Disambiguation is responsible with choosing the interpretations that convey the contextual usage. Disambiguation tackles lexical ambiguity (e.g., depending on the context "back" can be an adjective "back door" and an adverb "go back"), syntactic ambiguity (e.g., "Bear left at zoo."), semantic ambiguity ("There was not a single woman at the party."), and figures of speech (metonymy and metaphor).

2.2.3 Pragmatic Analysis

The semantic analysis discussed previously can provide a set of possible semantic interpretation. Yet, only one possible interpretation should be accepted and additional information is required to make an eliminatory decision. The pragmatic interpretation process takes into account information about the current situation when the speech occurred, as well as other data about external factors that could influence the speech process. Some of the issues solved by pragmatic analysis are indexical and anaphora/cataphora resolution. Indexicals are phrases that refer to the current situation (e.g., in a personal diary entry, the entire content entered during a day refers to that given time scene and author). Anaphora/cataphora is the occurrence of phrases that refer to object introduced previously/subsequentially.

2.3 Text Mining

Text mining (text data mining) represents the discovery of interesting and non-trivial patterns or knowledge from text documents through the use of computers [10]. It is a variation of the field data mining, which tries to extract interesting patterns from large databases. Text mining differs from data mining as patterns are extracted from natural language text rather than structured databases of facts.

The term "text visualization" emerges from the output of text mining and has been used to describe a variety of techniques for depicting the semantic characteristics of the free-text components of documents in large document collections. The so-called semantic mapping methods also typically strive to depict detailed inter- and intra-set similarity structure [11]. Text visualization varies in the degree of complexity based on the focus of interest expressed in analyzing text subcomponents (word, sentence, paragraph, content).

The simplest level of visualization is concerned with analyzing individual words only and results in a form of text visualization called "tag cloud". A tag cloud displays words in terms of their frequency inside the analyzed text, thus emphasizing on word commonality [12, 13]. The challenge presented by tag clouds is the

inconsistency between good readability and careful layout (in this case good readability results in almost random special layout).

A more complex text visualization process makes use of contextual semantic information. Theme River [14] uses semantic information in order to construct a graphical representation of the dynamics of themes and ideas encountered in large documents or across collections of documents (see Figure 3). The themes in the collection of documents under analysis are represented by a "river" that flows left to right through time [15], where each individual theme is depicted through a colored "current"" flowing within the river. The width of the river represents the collective strength (frequency of occurrence, given by number of words relating to the theme of interest) of the selected themes in the underlying documents. Existing ontologies[5] can be used to primarily group words with similar meaning and secondarily build summaries of the text under analysis.

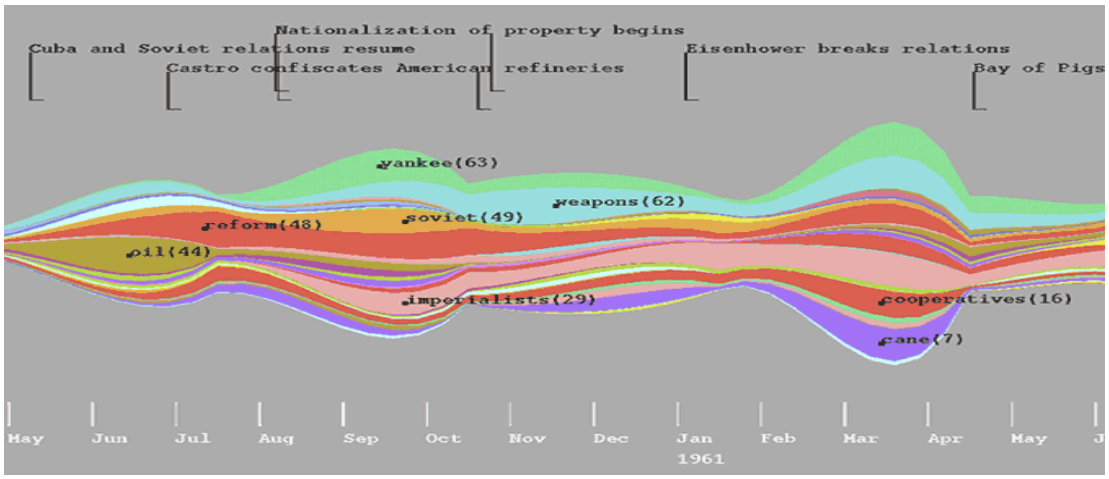

Figure 3 ThemeRiver showing Castro data from November 1959 through June 1961

Textual semantic relationships can be displayed under the form of a network [17], often constructed on co-occurrence relations (e.g., two words are connected if they occur in all documents under consideration; different pre-conditions can be used, based on the goal of the analysis). This approach presents two drawbacks: typical graph layouts may result in a jumbled text and co-occurrence is not giving enough focus on a type of relationship [18].

[5] Ontology : a formal representation of a set of concepts within a domain and the relationships between those concepts. The most extensively used ontology in NLP research is WordNet [16] C. Fellbaum, *WordNet. An electronic lexical database*, Cambridge, MA: MIT Press, 1998.

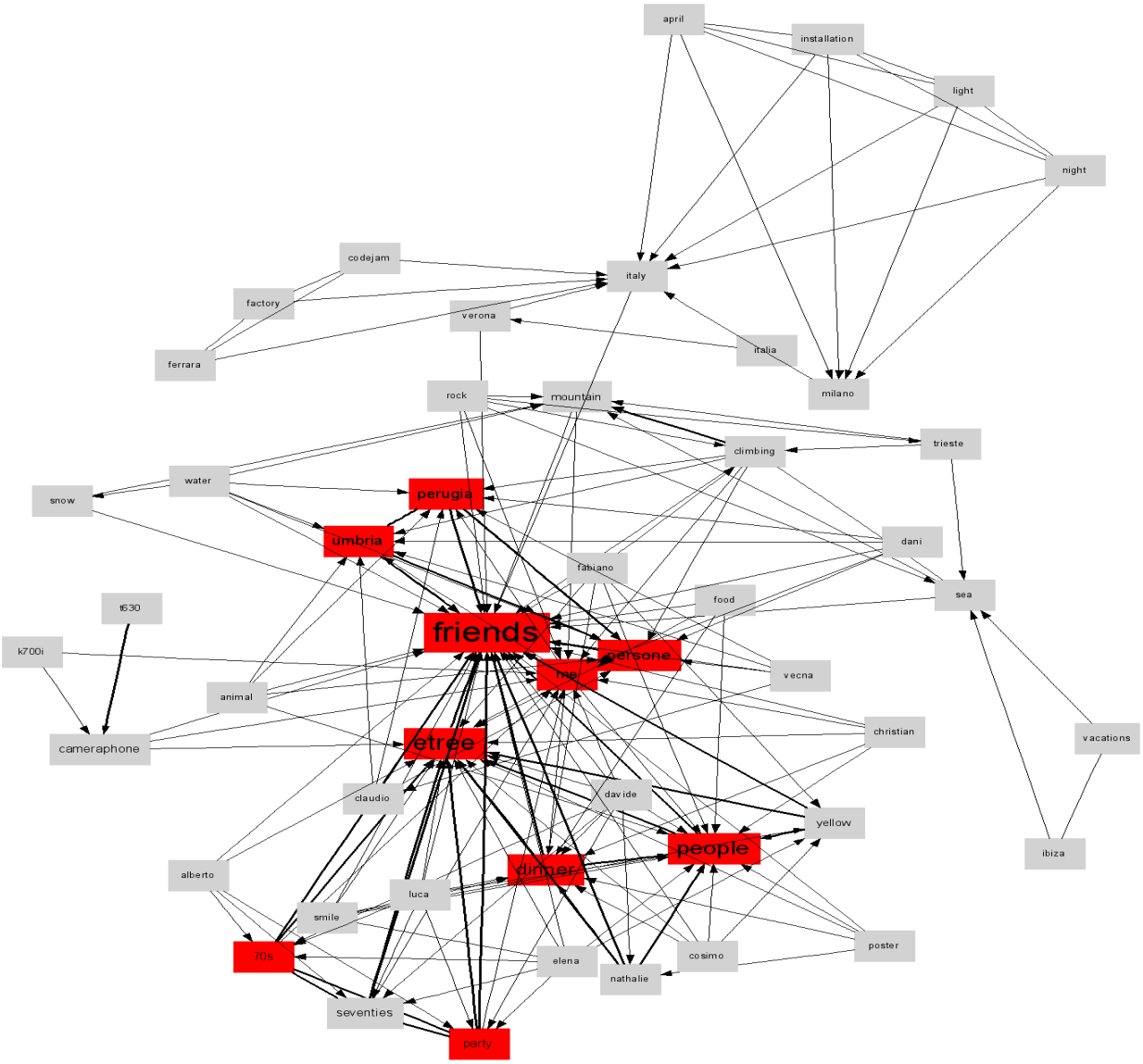

Figure 4 Semantic Network Sample [6]

2.3.1 Existing NLP Systems

Many of the NLP techniques have been implemented as a suite of libraries using specific programming languages (e.g., Java, C++, Ruby). A description of the most frequently used NLP libraries is included in Table 2.

[6] Graph taken from www.indiana.edu/~clcl/index.html

Table 2: NLP Libraries

Name	Programming Language	Examples of Featured Tools
LingPipe[3]	Java	• Track mentions of entities • Link entity mentions to database entries • Entities-actions relationship discovery • Spell correction
Stanford NLP [2]	Java	• Text parser • Part of speech tagger • Named entity recognition • Classifier
Alchemy API [19]	Java	• Entity extraction • Text categorization[7] • Language detection[8]
MontyLingua [20]	Java Python	• Entity extraction • Text categorization • Language detection
Natural Language Toolkit (NLTK) [21]	Python	• Entity extraction • Text categorization • Language detection • Extensible framework

[7] Text categorization: assigning the most likely topic category (e.g., news, sports, business, etc) to a document under analysis.

[8] Language detection: process of identifying the natural language used inside a document under consideration.

3 Our Approach

This project work consists of two parts: firstly, a NLP application was created for analyzing any file format, and secondly, software framework was designed for further development of NLP functionalities. The NLP application consists of a graphical user interface, along with NLP algorithms. The NLP framework allows for extensibility of existing feature, as well as for implementation of new features.

For simplicity, our system will be further referred to as "Text Visualization and Knowledge Extraction (TVKE)".

3.1 NLP Application

The TVKE application presents four progressive stages: content extraction, data analysis, data storage, and display of analysis results. Multiple files are usually given as input to the system at the same time; this file bundle is referred to as a "project". The four stages of TVKE are applied to each of the file given as input. The below sections discuss the procedures involved by each TVKE stage with respect to one file.

3.1.1 Content extraction

The content extraction stage consists of the identification of the file type, extraction of the readable content and metadata, and detection of the language of the readable content. A wide range of data files exists[22]. For the purpose of this project, only the most commonly used[23] human-readable file formats are considered (see Table 3). Any other file type can be added for consideration to the application, by implementing the required functionality.

The file format is determined by inspecting the extension and the metadata of the given file. If the file format is not supported by the TVKE the application halts.

Table 3: File types handled by system application

Currently Supported File Types
Microsoft Office (Word, PowerPoint, Excel)
.txt, rtf
Adobe PDF
HTML
E-mail Inbox[9]

[9] STMP email servers are the only e-mail servers currently supported by the TVKE system

A given file contains a "readable" version, or the version usually rendered by the default application, which is wrapped into a "raw" version by including additional information (metadata) both about the file and computer instructions for file manipulation. Metadata contains features not always readily available to the human reader that provide supplemental information useful in interpreting the "readable" content. Common metadata fields not dependent on the file format are: author, file creation date, date file was accessed, file size etc. A full list of metadata retrieved for analysis is included in Table 4.

Table 4: Metadata fields considered in the data analysis process

Metadata	**Type**	**Notes: file type**
Author	UserName	all
Created	Date	all
Description	Text	all
Title	Text	all
Content status	Text	Microsoft Office
Content Type	Text	Microsoft Office
Keywords	Text	Adobe, Microsoft Office
Last Modified	Date	all
Last Modified By	UserName	all
Last printed	Date	Microsoft Office
Last Printed By	User	Microsoft Office
Revision	Number	Microsoft Office
Subject	Text	Microsoft Office
Sent By	UserName	Email
Sent To	UserName	Email
Attachments	File	Email

Language of the readable content is inferred through decision making trained models. The models are run on the content of the file under analysis and they output a string representing the identified language. Supported languages and their respective output strings are included in Table 5: Natural Languages Supported by TVKE. The language identification algorithms were implemented at Stanford University, California, as part of the NLP Research Group.

Table 5: Natural Languages Supported by TVKE

	Natural Language	*Output String*
1.	Catalan	"cat"
2.	Danish	"dk"
3.	English	"en"
4.	Estonian	"ee"
5.	Finnish	"fi"

6.	French	“fr”
7.	German	“de”
8.	Italian	“it”
9.	Japanese	“jp”
10.	Korean	“kr”
11.	Norwegian	“no”
12.	Serbian	“sorb”
13.	Swedish	“se”
14.	Turkish	“tr”

3.1.2 Data Storage

Information is the key to the analysis process carried on by the TVKE application. Thus, available data is consistently stored either on the file system or inside a local database, depending on the type of data targeted for storage. For example, once a file is selected for analysis within the TVKE application, a copy of its raw content is saved for further reference on the disk[10].

A complete list of stored data and respective storing location is included in Table 6: Stored Information inside TVKE. The storing location is not pre-set and can be change upon convenience (e.g., local database vs cloud computing), as the storage process is implemented as a stand-alone part of the TVKE application. A database has been employed due to the easiness to retrieve data that meets specified conditions without having to iterate through an entire file to find the result of the search.

Table 6: Stored Information inside TVKE

Stored Information	**Stored location**
Raw file content	File system
Readable file content	File System
File Metadata	Database
NLP Analysis results	Database
Generated Knowledge Graph	File System

3.1.3 Data Analysis

The readable file content and the file metadata are sent to a processing engine that analyzes them based on NLP libraries and other implemented decision-making

[10] The file can have a big size and slow down the performance of the application if stored inside a database

algorithms. The data analysis process is a multi-step operation further described using the sample text, where the first sentence[24] is an example of sentence with correct grammar but no logical meaning:

"Colorless ideas sleep furiously. There is nothing to be done about them. They have chosen this deep sleep."

Step 1: The methods and algorithms used for this step vary based on the NLP library employed. For a given NLP library, the algorithms to be used are dynamically determined based on the file type and language of the file content retrieved from the content extraction stage.

A sentence parser is used to extract the sentences present inside a file. The parser takes as input the entire text and outputs the most likely sentence content. The output of the parser for the text under consideration is included in Table 7: Sentence parser sample output.

Table 7: Sentence parser sample output

Index	Sentence Content
1	Colorless ideas sleep furiously.
2	There is nothing to be done about them.
3	They have chosen this deep sleep.

Each identified sentence is first stored inside the database and then further analyzed. Sentence analyses consist of named entity recognition, parse tree generation and typed dependencies retrieval. Named entity recognition is performed independently, while parse tree generation and typed dependencies retrieval are performed at the same time. All analyses take as input the sentence content (e.g., "Colorless ideas sleep furiously.") and have been implemented using the Stanford NLP library.

The named entity recognition identifies words such as persons (e.g., Sam, Abraham Lincoln, Norah, Pierre etc), organizations (e.g., U.N., E.U., Health Care Organization etc), and locations (e.g., Mount Kilimanjaro, Paris, Black Sea etc). In the case of the text given for analysis above, the named entity recognition stage does not provide with any output, as there is no mention of a person, organization, or location. The parse tree generation stage supplies the syntactic structure of the sentence, as presented in Table 8: Sample output for parse tree. The tokens used for annotation are part of the Penn Treebank tagset (refer to

Appendix B: Treebank Tags). The typed dependencies stage takes as input the sentence content and it identifies the grammatical constraints between pairs of tokens present inside the sentence. It outputs the dependency type (refer to 6.2 Function Tags), the pair of tokens involved in the dependency, as well as the position inside the sentence for each of the two tokens (e.g., pair of tokens (ideas-2, Colorless-1), with index 2 for "ideas" and respectively 1 for "Colorless").

Table 8: Sample output for parse tree

Sentence 1: Colorless ideas sleep furiously.	
Named Entity	Colorless/O ideas/O sleep/O furiously/O ./O
Parse Tree	(ROOT (S (NP (JJ Colorless) (NNS ideas)) (VP (VBP sleep) (ADVP (RB furiously))) (. .)))
Typed dependencies	amod(ideas-2, Colorless-1) nsubj(sleep-3, ideas-2) advmod(sleep-3, furiously-4)]
Sentence 2: There is nothing to be done about them.	
Named Entity	There/O is/O nothing/O to/O be/O done/O about/O them/O ./O
Parse Tree	(ROOT (S [46.259] (NP [5.132] (EX [1.046] There)) (VP [39.985] (VBZ [0.144] is) (NP [35.212] (NN [7.218] nothing) (S [21.831] (VP [21.736] (TO [0.011] to) (VP [21.706] (VB [0.002] be) (VP [19.008] (VBN [4.987] done) (PP [12.559] (IN [4.351] about) (NP [7.535] (PRP [3.242] them))))))))) (. [0.002] .)))
Typed dependencies	expl(is-2, There-1) nsubj(is-2, nothing-3) aux(done-6, to-4) auxpass(done-6, be-5) infmod(nothing-3, done-6) prep(done-6, about-7) pobj(about-7, them-8)

Sentence 3: They have chosen this deep sleep.	
Named entity	They/O have/O chosen/O this/O deep/O sleep/O ./O
Parse Tree	(ROOT (S [51.235] (NP [4.914] (PRP [3.664] They)) (VP [40.152] (VBP [0.090] have) (VP [36.087] (VBN [6.892] chosen) (NP [25.449] (DT [3.859] this) (JJ [8.075] deep) (NN [10.744] sleep))))))
Typed dependencies	nsubj(chosen-3, They-1) aux(chosen-3, have-2) det(sleep-6, this-4) amod(sleep-6, deep-5) dobj(chosen-3, sleep-6)

Coreference resolution is performed on the given readable content. The coreference resolution algorithm takes as input the entire readable content and outputs the existing coreferences of each word. The output is based on the algorithm designed by Aria Haghighi and Dan Klein, 2009 [25].

Table 9: Sample output coreference resolution. Each digit represents one entity potentially coreferenced further inside the text; repeated digits represent repeated coreferences.

<1>Colorless ideas</1> sleep furiously. <2>There</2> is nothing to be done about <1>them</1> . <1>They</1> have chosen <3>this deep sleep</3> .

The same output format is expected for ***Step 1*** regardless of the used library. The analysis output should contain the content's sentences (e.g., "Colorless ideas sleep furiously.") and tokens (e.g., "Colorless", "ideas" etc), token relationships (e.g., amod(ideas-2, Colorless-1)), part of speech (Colorless/JJ), and named entities (e.g., "Joshua", "UMass").

Step 2: Output of ***Step 1*** is stored inside a relational database. The storage process is implemented as a parallel event that takes place at the same time with the analysis process. Analysis of the content has a higher priority and can cause postponement of storage process in the case of lack of sufficient computing resources (e.g., when processing a large novel, the storage step can be delayed until the entire novel has finished processing).

3.1.4 Analysis Results Display

The output of the NLP analysis is used for generating a visual representation of the knowledge present inside the files under consideration. Graph data is stored on the

file system in a XML format. Sample information stored for a graph is the graph nodes and edges, and node and edge properties (e.g., color, size, name etc).

For each file, there are multiple graphs stored on the file system. The default graph constructed for a file contains coreferencing words with a frequency higher than one. The base form of each coreferencing word is selected as a node inside the graph (e.g., from the list of coreferences "Colorless ideas-them-they", "Colorless ideas" is selected as the base form). The typed dependencies between the remaining coreferences of the same word (e.g., "them-they") and other words inside the text result in the edges of the graph. Only the typed dependencies involving a noun, verb, adjective or adverb (e.g., amod, dobj) are represented inside the graph. In case the typed dependency involves a pronoun, the noun referred to by the pronoun is used. The second graph type contains the named entities identified inside the text file and the typed dependencies of the named entity. The last graph type is generated at user request based on a noun present inside the text file. The graph contains the selected noun as the most central node as well as the typed dependencies of the selected node.

The user can interact with the generated graphs and removed edges in order to obtain a better graph readability. There are up to 10 stages to remove graph edges, where each stage implies removing 10 percent of the total number of graph edges. Edges are assigned a priority (priority is represented by the sum of frequencies of occurrence inside the text of the two edge nodes), and the first to be removed are edges with a lower priority (random selection for edges with same priority). If a node is not connected by any edge it is removed from the graph.

3.2 Software Architecture[11] Development

This section provides a high level overview of the architecture for the NLP system developed.

3.2.1 Introduction

TVKE is an extendable application for universal natural language processing that supports any type of natural language and allows for multiple knowledge visualization perspectives. The targeted users of the application are researchers in the field of NLP and linguistics, software developers who are designing NLP applications, and regular computer users. This document provides a high level description of the application architecture: the goals of the application, the use cases supported by the system, and architectural styles and components.

The TVKE architecture is represented in this document as a series of architecture views

[11] The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

(use case view, logical view and process view) and the architecturally significant decisions made during the TVKE development process. There is no separate implementation view described in this section. The architectural views are based on the Unified Modeling Language (UML) Model diagrams.

3.2.2 Definitions, Acronyms, and Abbreviations

Table 10: Definitions, acronyms and abbreviations used for describing the architecture of the TVKE system

Term	Explanation
Hot Spots[26]	Parts of the TVKE used by the programmer for extension and addition of system specific functionality (see Table 11).
Programmer	The software developer of the TVKE; the programmer's main responsibility is to extend hot spots, and run system tests as needed.
NLP Library	Collection of classes used to develop NLP applications. The classes available in a library represent tools of various degree of complexity, ranging from token extraction to anaphora resolution.

3.2.3 Architectural Goals and Constraints

The overall goal of TVKE is to create an extensible NLP framework that can support any Java-based NLP library (e.g., Ling Pipe, Stanford NLP, Monty Lingua[27]) and text visualization perspective, and to provide the end-user with a user-friendly text analysis interface. TVKE was designed to improve performance time and flexibility in the usage of current NLP libraries.

3.2.3.1 Constraints

The main constraint of the system is the limitation to the Java run time environment. An additional constraint is created by the design and implementation of given NLP libraries. Each NLP library presents with algorithms or resources that do not allow for concurrency, and have to be executed sequentially (e.g., the Stanford part of speech library uses the bidirectional-wsj-0-18.tagger classification model does not allow for parallel processing).

3.2.3.2 Design Decisions

The major design decisions encountered during the system development process are explained in this section. Extensive design decisions were required by the following concepts:

1. **Hot Spots**

A number of hot spots were determined during the development process. The hot spots allow for extensibility of the system and are targeted at the language processing as well as the data visualization level. The explicit hot spots are included in Table 11: Hot Spots for the TVKE System.

Table 11: Hot Spots for the TVKE System

Hot Spot	Description
NLP Library	• Different NLP libraries can be used for text processing (e.g.: sentence extraction, named entity and part of speech recognition etc.) • Individual text processing tasks can be executed using a different NLP library, provided the task is not depended on other tasks
Data Visualization Frame	• The output of the NLP processing step can be displayed using any graph visualization application.

3.2.4 Use-Case View

The use-case view presents the main actors of the system and their perception on the functionality provided by TVKE. This section presents the context for the remaining section. Main use-cases of the TVKE system are included in Figure 6.

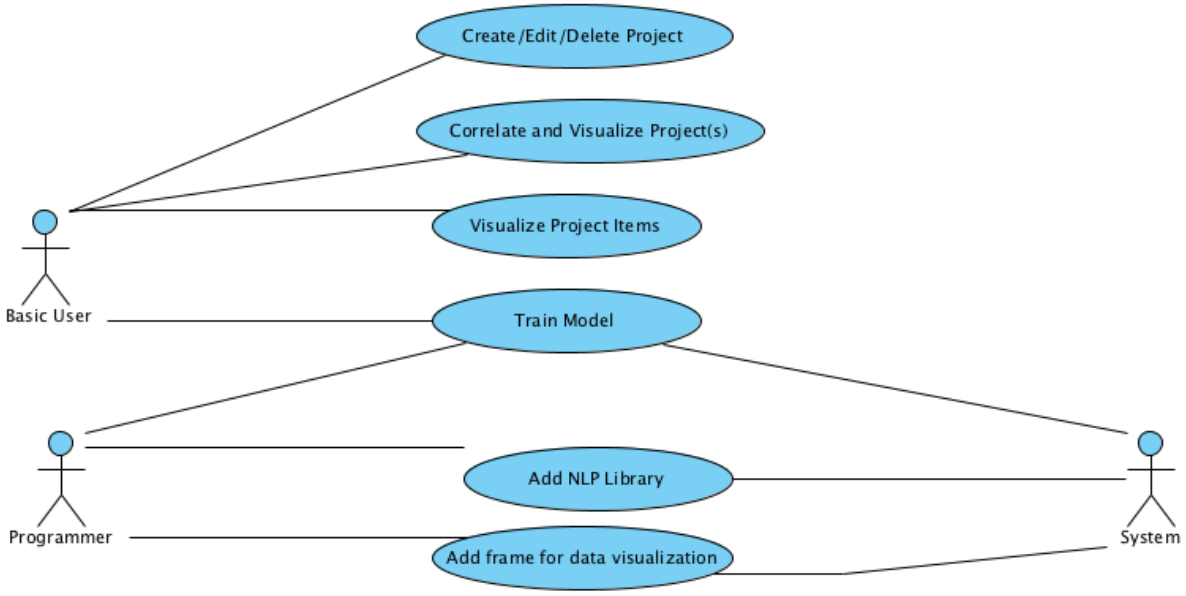

Figure 5 Main use cases of the TVKE system

3.2.5 Logical View

This section describes the logical view of the architecture. The following components are described:

- The most important classes of TVKE depicted through UML Class Diagrams
- Packages organization depicted through UML Package Diagram

3.2.5.1 Architecturally Significant Design Packages

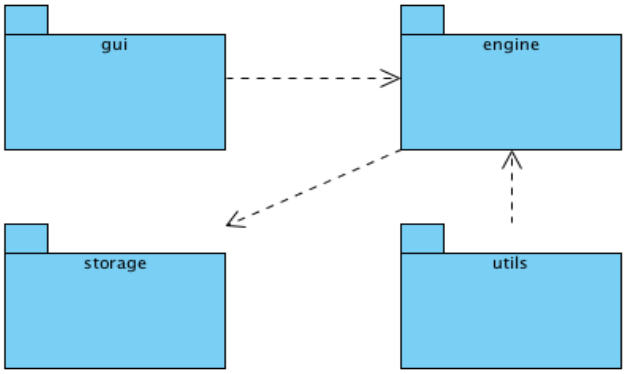

Figure 6 General overview of system packages

I. gui: Package designed to hold the data display and graphical user interaction frame.

II. engine: Package designed for handling NLP tasks and the logic of the overall system.

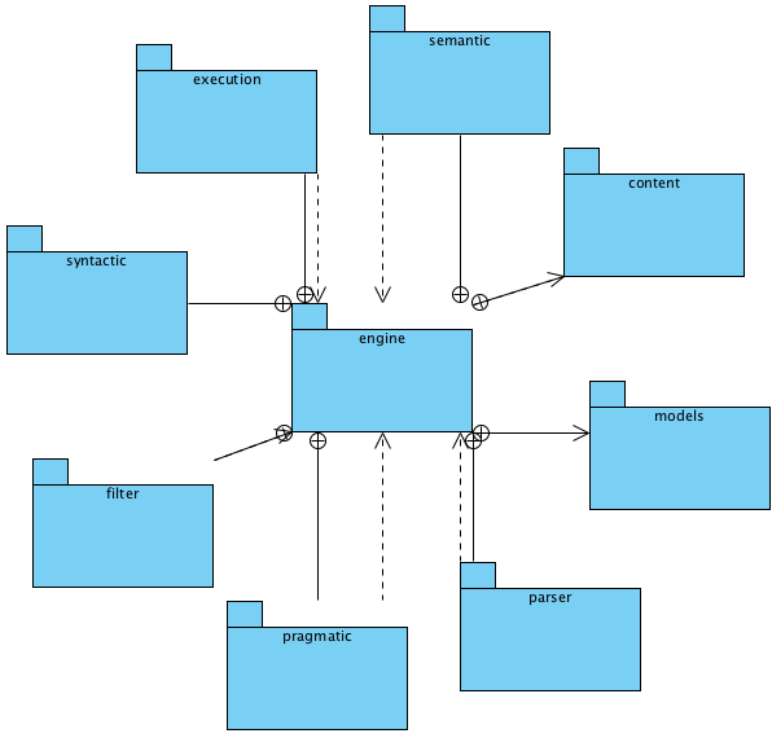

Figure 7 Engine Sub-package distribution

III. storage: package designed for storing the output of text processing.
IV. utils: package providing generic functionality for the system.

3.2.6 Process View

TVKE runs within a user process. Additional threads are created during the processing of each text file. Technically, each stage inside NLP results in a unique thread. The execution framework of the system manages all generated threads.

3.2.7 Data View

The current framework design does account for persistent storage. Data is stored both inside a relational database and on the file system. The data storage location can be changed based on available storage resources (e.g., the relational database can be replaced with a flat-file).

3.2.8 User Interaction overview

The main window of the TVKE system contains a short description of the system and allows for initial system setting:

1. Selection of NLP library
2. Enabling/disabling user prompts.

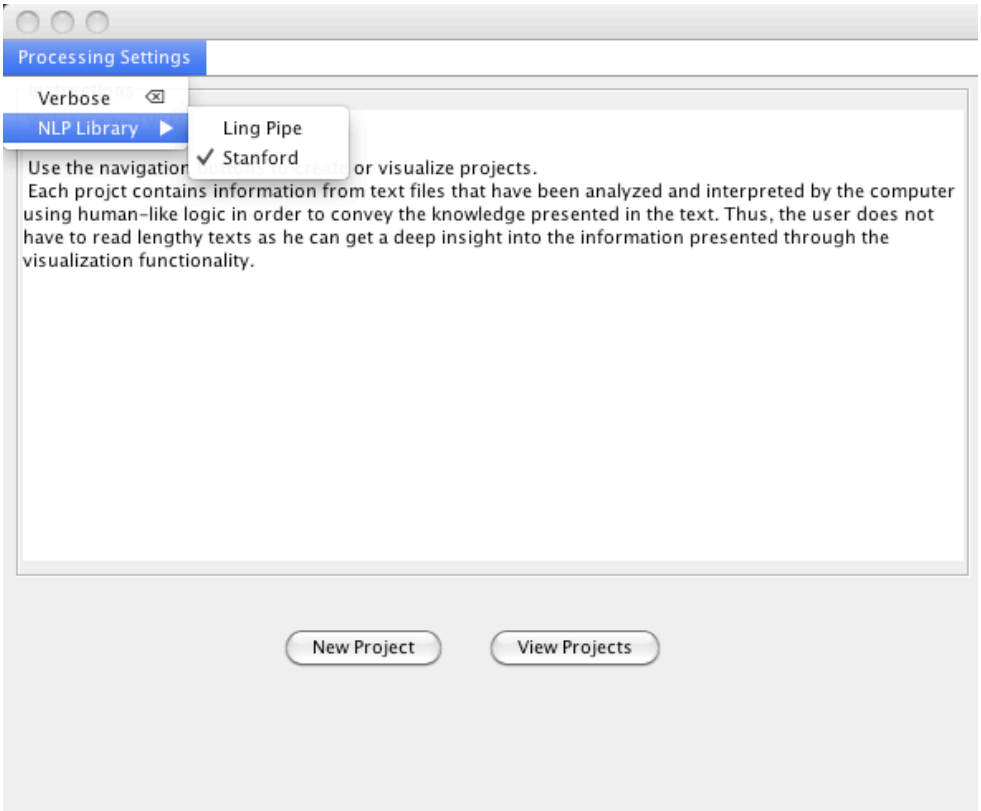

Figure 8 TVKE Welcome window

A TVKE project is defined as a collection of files interpreted together by the system and stored for further reference. The user has the option to visualize previously created projects (see Figure 10), or to create a new project.

The "New Project" window (see Figure 9) requires user input for the name and description of the projected to be created (in the depicted case the project name is "miserables_novel" and the project description is "test"). The user has to include at least one file for analysis, either a file stored on the file disk, a URL, or an email address (in this case the "Miserable ch1.doc", "Miserables_ch2.doc", "Miserables_ch3.doc", and "Miserables_ch4.doc" were given for analysis). The "Create and Analyze Sources" button creates a new project with the given specifications and starts the analysis of the selected sources. Once the project has been created the user is taken to the "View Project" window (see Figure 10 TVKE "View Project" window), where the list of projects and their respective sources are displayed.

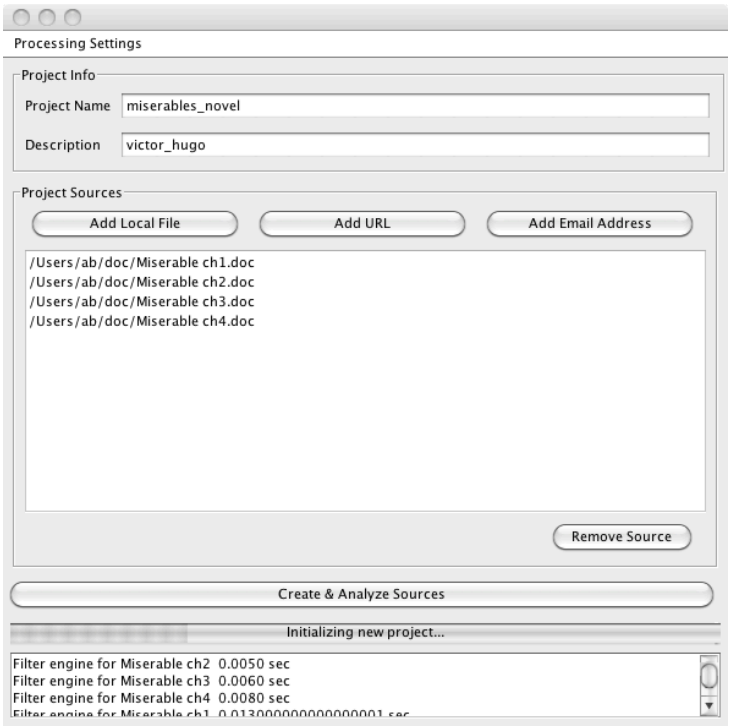

Figure 9 TVKE "Create Project" window

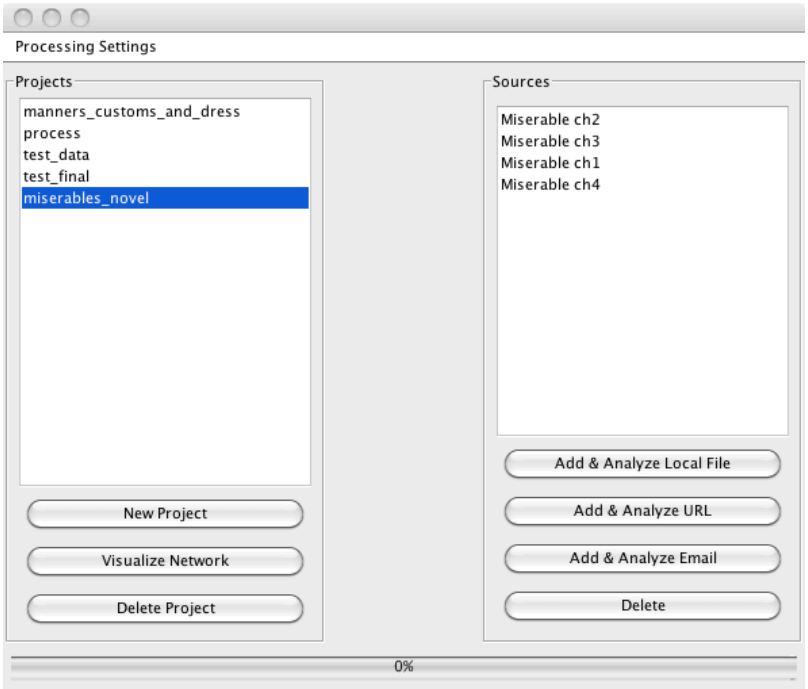

Figure 10 TVKE "View Project" window

The user can choose to visualize the results of the analysis of the project sources. The default visualization of a project contains all named entities present in the project (persons, locations, organizations). For the created project

"miserables_novel" the user can see named entities like "Madam Magloire", "Myriel", "Paris", "Italy" etc.

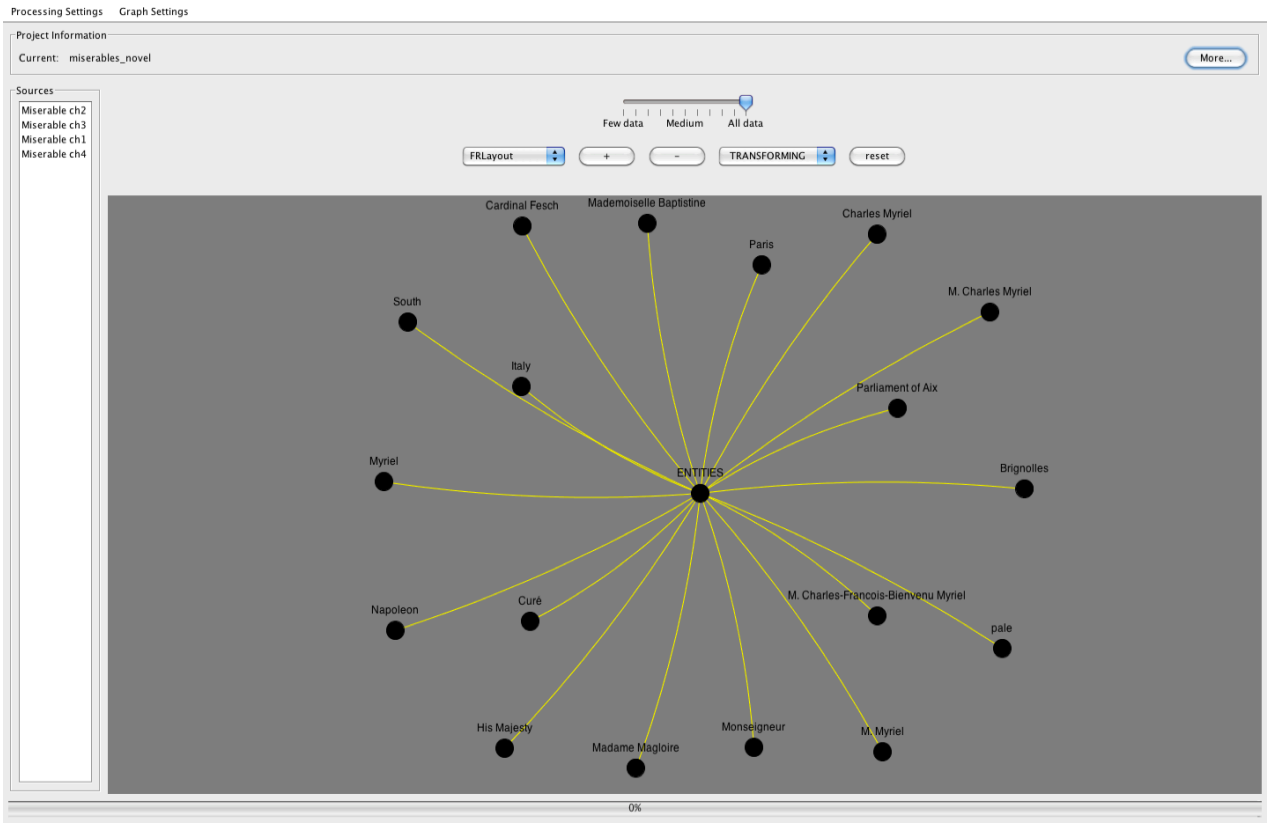

Figure 11 TVKE overview of a default graph for a project

The user can also choose to visualize the content of a single project source (in this case "Miserables ch1.doc"). The result will be a network display of the most important information presented inside the analyzed files. The graph presented in Figure 12 TVKE Project Visualization for one project source informs the user that "M. Myriel" was a bishop, more specifically the "Bishop D." and that the "worthy Cure was waiting anteroom". For each network node, the user can visualize the occurrences of the node data inside the text file (see Figure 13 TVKE Word occurrences display). The user can chose to display information from multiple sources, by selecting more than one source from the "Sources" list displayed on the left side of the "Project Visualization" window (see Figure 14 TVKE Project Visualization for multiple project sources). More interesting networks can be displayed, depending on the content of the analyzed files.

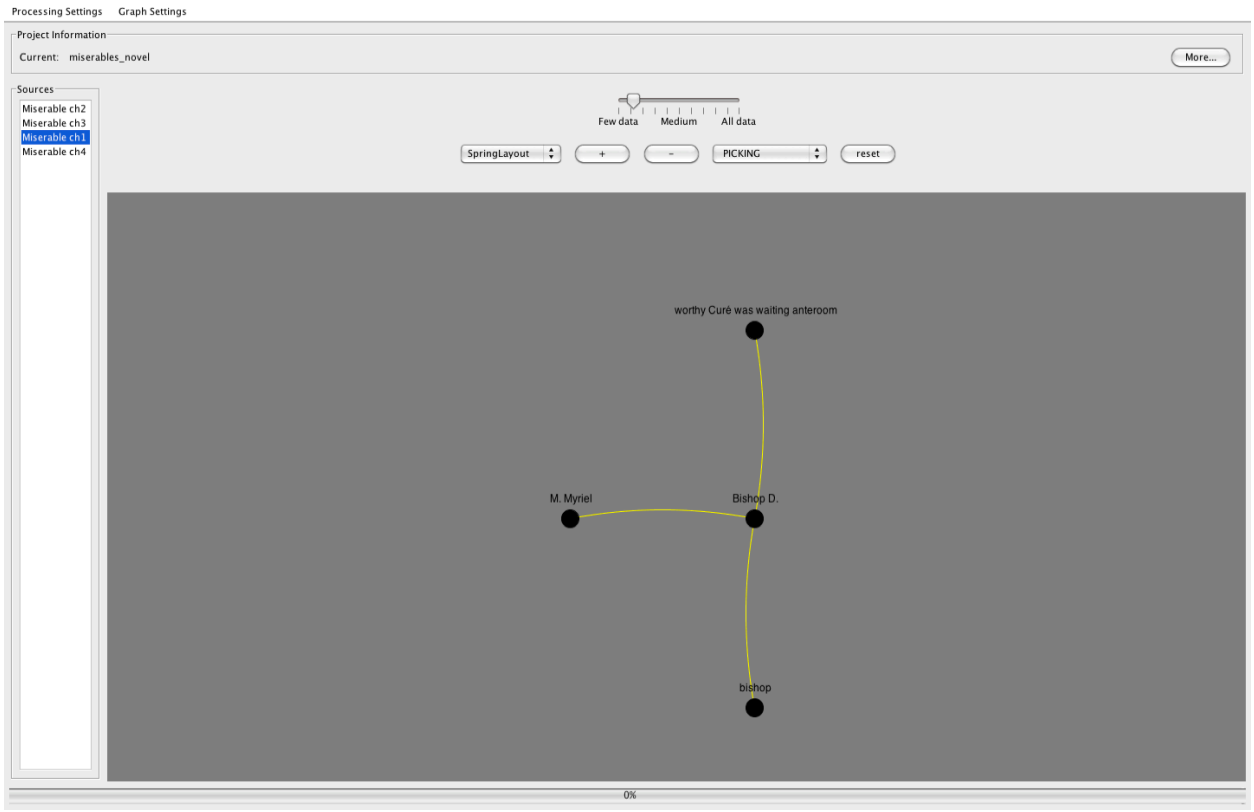

Figure 12 TVKE Project Visualization for one project source

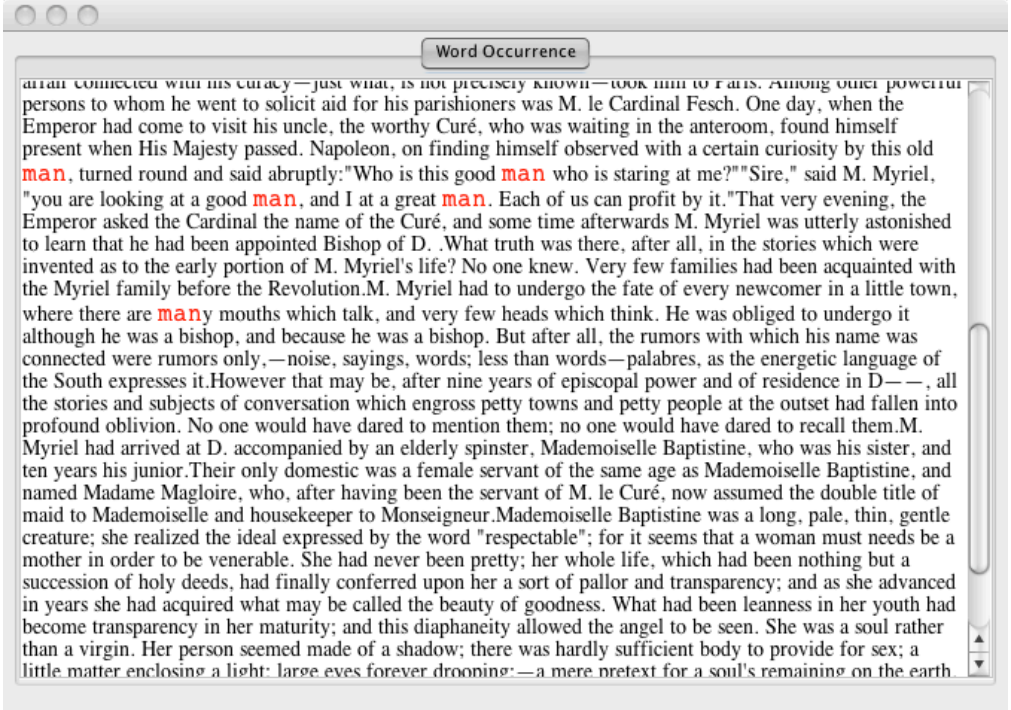

Figure 13 TVKE Word occurrences display

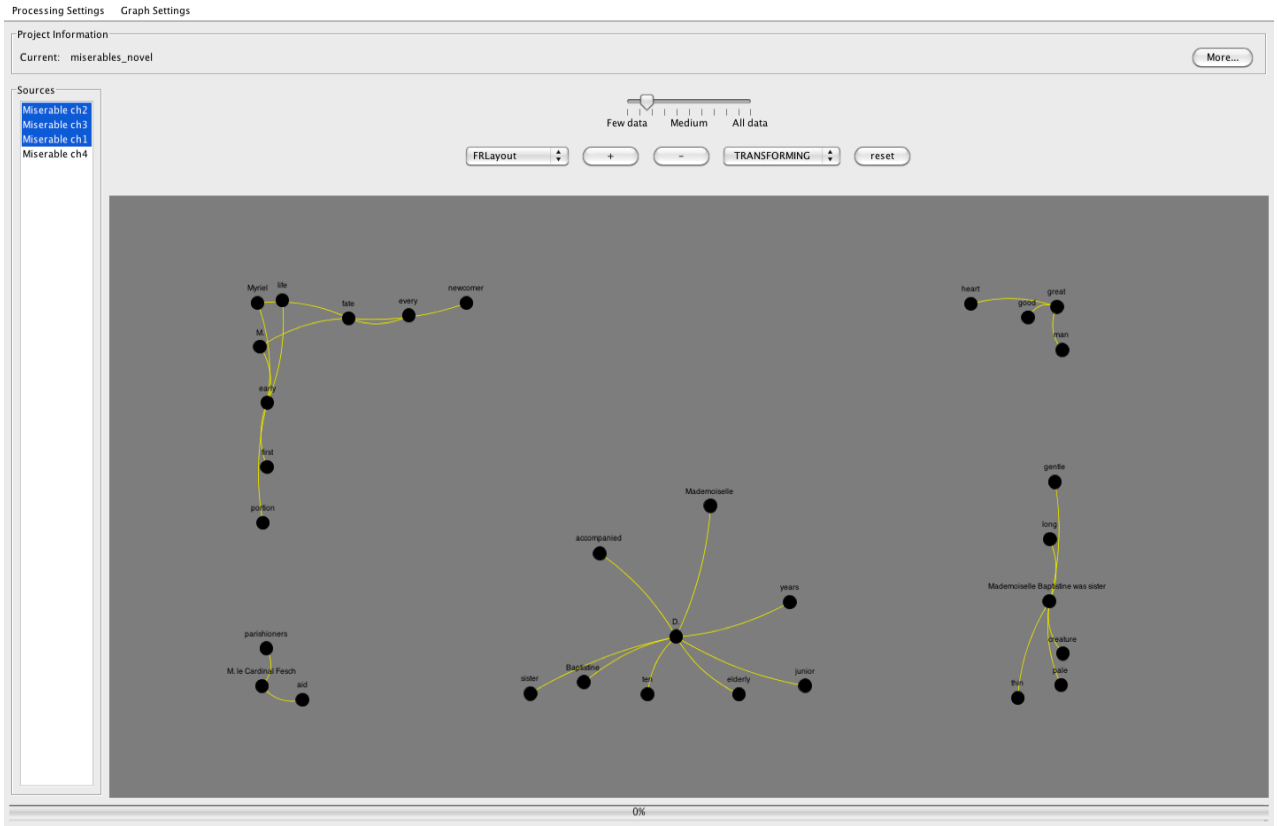

Figure 14 TVKE Project Visualization for multiple project sources

The system can display more information on a project: date of creation, the number of named entities present inside all the analyzed files, and the number of specific parts of speech (nouns, verbs, and adjectives). The user can see whether a project contains more descriptive files, or technical files, depending on the number of nouns/verbs encountered. In the case of the analyzed file, there are 54 named entities, 267 nouns, 26 verbs, and 85 adjectives.

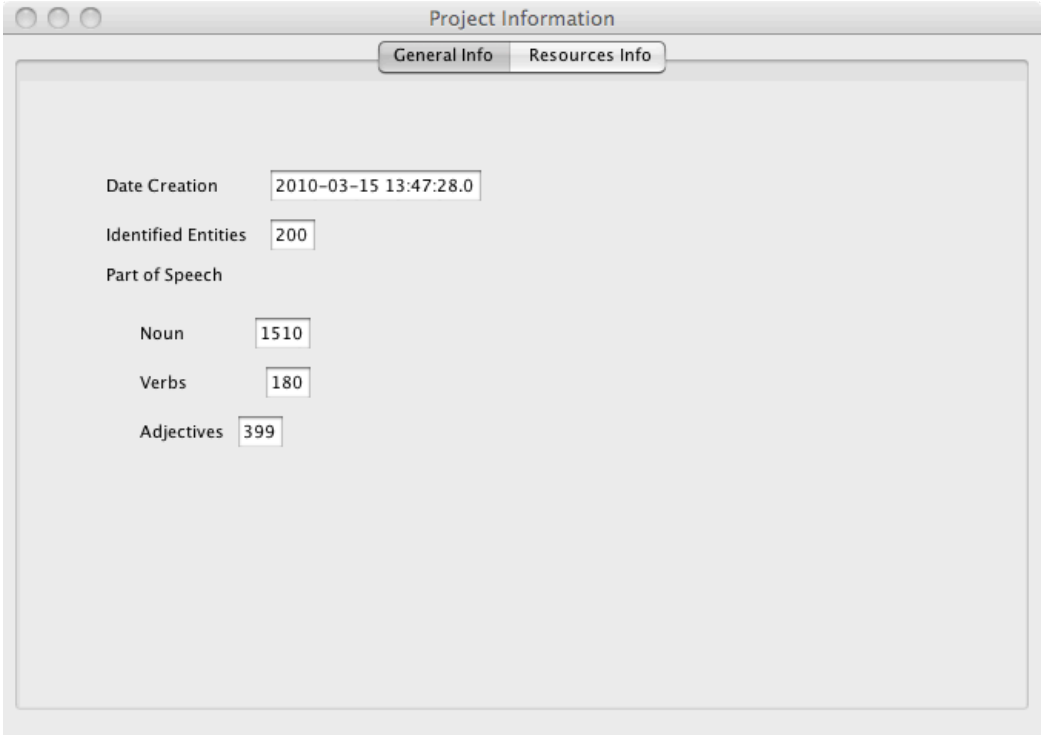

Figure 15 TVKE Project information

Information about individual resources is also displayed. The user can visualize all nouns present inside a selected file, and their corresponding frequency. On selecting a specific noun the file network is redisplayed to emphasize the occurrence of the selected noun (if the noun was part of the graph, otherwise no change is made). The new displayed graph generated after the selection of the noun "Mademoiselle" is displayed in Figure 17.

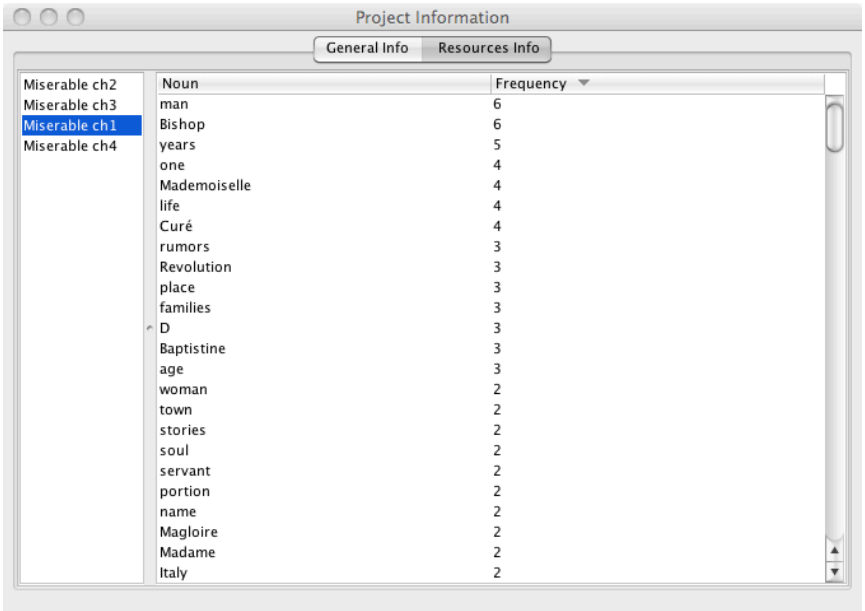

Figure 16 TVKE Project source information

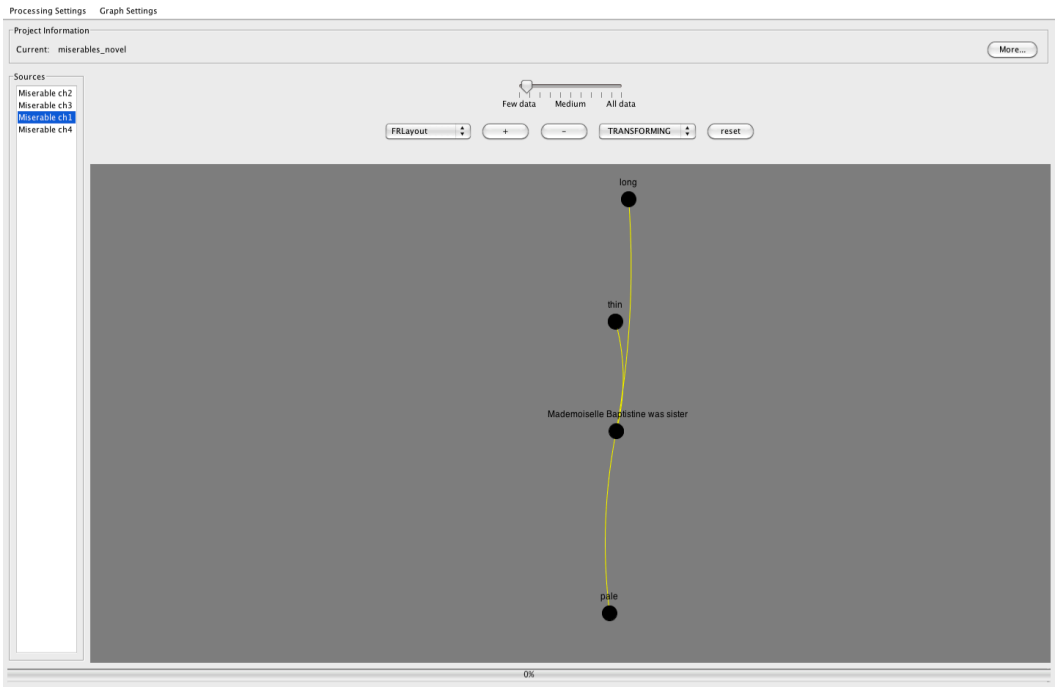

Figure 17 TVKE Graph generated after selection of specific noun encountered in analyzed sources

4 Conclusions and Future Work

This MQP has greatly benefited from the feedback of the Knowledge Discovery and Data Mining Research Group (KDDRG) and Artificial Intelligence Research Croup (AIRG) at Worcester Polytechnic Institute. Periodical demonstrations were given and this gave a good indication whether the project satisfied the goal of usability, thus prioritizing the development of certain system features.

Unlike most currently available toolkits for NLP analysis and graph visualization for text, TVKE performs semantic and pragmatic analysis in addition to syntactic analysis. The TVKE system generates graphs with emphasis on syntactic and semantic features. The concurrent characteristic of the system allows for faster processing (analysis and interpretation) of given text files, considering the complexity of the analysis undertaken. TVKE is an extensible framework and thus allows for integration of different NLP libraries or text visualization features. The system was implemented with specific hot spots, to assist the programmer with the extensibility task.

The TVKE has been evaluated in terms of time performance. The results are included in Table 12: TVKE time performance. Observational evaluation of the system was performed during the project implementation to determine the accuracy of the NLP analysis results. Based on observations made during the implementation it was concluded that the best NLP library to employ is Stanford (previously employed libraries like Ling Pipe did not correctly identify the sentence structure all the time).

Table 12: TVKE time performance (seconds)

Input	**# words**	**NLP analysis type**		
		syntactic	**parser**	**pragmatic**
Sentence	25	0.016	2.344	0.16
Paragraph	108	0.017	7.575	0.208
Book chapter-1	1424	0.026	90.542	0.746
Book chapter	8540	0.099	511.565	11.59
Novel	116 004	63	26599.673	18700.457

Future work should include more specific applications of TVKE, like dynamic representation of named entities inside the text under analysis, social network generation from e-mail data, and training decision models for text classification, among others. Another desirable functionality of the TVKE system is making processing results available online, by employing online cloud storage.

5 Appendix A: General NLP Terms

Anaphora: in the field of linguistics, anaphora represents an instance of an expression referring to another.
Anaphora resolution: represents the problem of determining in respect to a text given for analysis what a pronoun or a noun refers to.
Co-occurrence: frequent occurrence of two terms from a text corpus alongside each other in a specific order.
Language Detection- process of identifying the natural language used inside a document under consideration.
Nonterminal and Terminal symbols- symbols used to construct production rules in a formal grammar.
Production rules- defines a nonterminal symbol in terms of terminal and nonterminal symbols. A production rule has the form left side= right side.
Terminal symbols- see nonterminal symbols.
Text categorization- assigning the most likely topic category (e.g., news, sports, business, etc) to a document under analysis.
Text corpus (pl. text corpora): large and structured set of texts, used in linguistic analysis.
Transformational grammar - a concept used in linguistic when referring to a formal grammar that enables natural and computer language generation.

6 Appendix B: Treebank Tags

6.1 Bracket Labels

6.1.1 Clause Level

S	simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a *wh*-word and that does not exhibit subject-verb inversion.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a *wh*-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.
SINV	Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ	Inverted yes/no question, or main clause of a *wh-phrase in SBARQ.*

6.1.2 Phrase Level

ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection. Corresponds approximately to the part-of-speech tag UH.
LST	List marker. Includes surrounding punctuation.
NAC	Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to N-bar level but used quite differently.
PP	Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle. Category for words that should be tagged RP.
QP	Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Verb Phrase.
WHADJP	Wh-adjective Phrase. Adjectival phrase containing a wh-adverb, as in *how hot*.
WHAVP	Wh-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a wh-adverb such as *how* or *why*
WHNP	Wh-noun Phrase. Introduces a clause with an NP gap. May be null

	(containing the 0 complementizer) or lexical, containing some wh-word, e.g. *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.
WHPP	Wh-prepositional Phrase. Prepositional phrase containing a wh-noun phrase (such as *of which* or *by whose authority*) that either introduces a PP gap or is contained by a WHNP.
X	Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing the...the-constructions.

6.1.3 Word Level

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun (prolog version PRP-S)
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun

WP$	Possessive wh-pronoun (prolog version WP-S)
WRB	Wh-adverb

6.2 Function Tags

6.2.1 Form/Function Discrepancies

ADV (adverbial)	marks a constituent other than ADVP or PP when it is used adverbially (e.g. NPs or free ("headless" relatives). However, constituents that themselves are modifying an ADVP generally do not get -ADV. If a more specific tag is available (for example, -TMP) then it is used alone and -ADV is implied.
NOM (nominal)	marks free ("headless") relatives and gerunds when they act nominally.

6.2.2 Grammatical Role

DTV (dative)	marks the dative object in the unshifted form of the double object construction. If the preposition introducing the "dative" object is for, it is considered benefactive (-BNF). -DTV (and -BNF) is only used after verbs that can undergo dative shift.
LGS (logical subject)	is used to mark the logical subject in passives. It attaches to the NP object of *by* and not to the PP node itself.
PRD (predicate)	marks any predicate that is not VP. In the *do so* construction, the *so* is annotated as a predicate.
PUT	marks the locative complement of *put*.
SBJ (surface subject)	marks the structural surface subject of both matrix and embedded clauses, including those with null subjects.
TPC ("topicalized")	marks elements that appear before the subject in a declarative sentence, but in two cases only: 1. if the front element is associated with a *T* in the position of the gap. 2. if the fronted element is left-dislocated (i.e. it is associated with a resumptive pronoun in the position of the gap).
VOC (vocative)	marks nouns of address, regardless of their position in the sentence. It is not coindexed to the subject and not get -TPC when it is sentence-initial.

6.2.3 Adverbials

BNF (benefactive)	marks the beneficiary of an action (attaches to NP or PP).
DIR	- marks adverbials that answer the questions "from where?" and "to

(direction)	where?" It implies motion, which can be metaphorical as in "*...rose 5 pts. to 57-1/2*" or "*increased 70% to 5.8 billion yen*" -DIR is most often used with verbs of motion/transit and financial verbs.
EXT (extent)	marks adverbial phrases that describe the spatial extent of an activity. -EXT was incorporated primarily for cases of movement in financial space, but is also used in analogous situations elsewhere. Obligatory complements do not receive -EXT. Words such as *fully* and *completely* are absolutes and do **not** receive -EXT.
LOC (locative)	marks adverbials that indicate place/setting of the event. -LOC may also indicate metaphorical location. There is likely to be some varation in the use of -LOC due to differing annotator interpretations. In cases where the annotator is faced with a choice between -LOC or -TMP, the default is -LOC. In cases involving SBAR, SBAR should not receive -LOC. -LOC has some uses that are not adverbial, such as with place names that are adjoined to other NPs and NAC-LOC premodifiers of NPs. The special tag -PUT is used for the locative argument of *put*.
MNR (manner)	marks adverbials that indicate manner, including instrument phrases.
PRP (purpose or reason)	marks purpose or reason clauses and PPs.
TMP (temporal)	marks temporal or aspectual adverbials that answer the questions *when, how often*, or *how long*. It has some uses that are not strictly adverbial, auch as with dates that modify other NPs at S- or VP-level. In cases of apposition involving SBAR, the SBAR should not be labeled -TMP. Only in "financialspeak," and only when the dominating PP is a PP-DIR, may temporal modifiers be put at PP object level. Note that -TMP is not used in possessive phrases.

6.2.4 Miscellaneous

CLR (closely related)	marks constituents that occupy some middle ground between arguments and adjunct of the verb phrase. These roughly correspond to "predication adjuncts", prepositional ditransitives, and some "phrasel verbs". Although constituents marked with -CLR are not strictly speaking complements, they are treated as complements whenever it makes a bracketing difference. The precise meaning of -CLR depends somewhat on the category of the phrase.
CLF (cleft)	marks it-clefts ("true clefts") and may be added to the labels S, SINV, or SQ.
HLN (headline)	marks headlines and datelines. Note that headlines and datelines always constitute a unit of text that is structurally independent from the following sentence.
TTL (title)	is attached to the top node of a title when this title appears inside running text. -TTL implies -NOM. The internal structure of the title is

	bracketed as usual.

7 Bibliographical References

[1] T. Dean, J. Allen, and Y. Aloimonos, *Artificial Intelligence Theory and Practice*: The Benjamin/Cummings Publishing Company, INC, 1995.

[2] Stanford. "The Stanford Natural Language Processing Group," 20 January, 2010; http://nlp.stanford.edu/.

[3] Alias-i. "LingPpipe," 20 January, 2010; http://alias-i.com/lingpipe/web/about.html.

[4] S. J. Russell, P. Norvig, and J. Canny, *Artificial intelligence : a modern approach*, 2nd ed., Upper Saddle River, N.J.: Prentice Hall, 2003.

[5] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory,* vol. 2, no. 3, pp. 113-124, September 1956.

[6] N. Chomsky, *Syntactic Structure*, 1st ed.: Mouton & Co., 1957.

[7] E. F. K. Koerner, R. E. Asher, and W. J. Hutchins, "Concise history of the language sciences: from the Sumerians to the cognitivists," pp. 431-445: Oxford: Pergamon Press, 1995.

[8] C. S. I. L. Lexicon. "Tranformational Grammar," 15th January, 2010; http://www.class.uh.edu/cogsci/Lang/Entries/transformational_grammar.html.

[9] A. Cawsey, *Natural Language Processing*: Prentice Hall Europe, 1998.

[10] M. Hearst. "What is Text Mining?," 12 February, 2010.

[11] J. Risch, A. Kao, S. R. Poteet *et al.*, "Text Visualization for Visual Text Analytics," *Lecture Notes in Computer Science,* vol. 4404/2008, pp. 154-171, 2008.

[12] J. Feinberg. "Wordle," 20 January, 2010; http://www.wordle.net/.

[13] J. Clark. "Neoformix Blog," 20 January 2010, 2010; http://www.neoformix.com/.

[14] S. Havre, E. Hetzler, P. Whitney *et al.*, "ThemeRiver: Visualizing Thematic Changes in Large Document Collections," *IEEE Transactions on Visualization and Computer Graphics,* vol. 8, no. 1, pp. 9-20, 2002.

[15] P. N. N. Laboratory. "Information Visualization," 3rd February, 2010.

[16] C. Fellbaum, *WordNet. An electronic lexical database*, Cambridge, MA: MIT Press, 1998.

[17] S. C. Shapiro, "A Net Structure for Semantic Information Storage, Deduction and Retrieval," *Second International Joint Conference on Artificial Intelligence*, pp. 512-523, 1971.

[18] F. H. Ham, W. Martin, and V. B. Fernanda, "Mapping Text with Phrase Nets," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1169-1176, November/December 2009.

[19] Alchemy. "AlchemyAPI Transforming Text into Knowledge," 20 January, 2010; http://www.alchemyapi.com/.

[20] H. Liu. "MonthyLingua 2.1," 20 January, 2010; http://pypi.python.org/pypi/MontyLingua/2.1.

[21] NLTK. "Natural Language Toolkit," 20 January, 2010; http://www.nltk.org/.

[22] FileInfo.com. "Data Files," 22nd February, 2010; http://www.fileinfo.com/filetypes/data.

[23] FileInfo.com. "Common File Types," 22nd February, 2010; http://www.fileinfo.com/common.php.
[24] N. Chunsky, *Syntactic Structure*, 1st ed.: Mouton & Co., 1957.
[25] A. Haghighi, and D. Klein, "Simple Coreference Resolution with Rich Syntactic and Semantic Features."
[26] M. E. Markiewicz, and C. J. P. Lucena, "Object Oriented Framework Development," *ACM Crossroads,* vol. 7, no. 4, pp. 3-9, 2001.
[27] H. Liu. "Montylingua," 23rd February, 2010; http://web.media.mit.edu/~hugo/montylingua/#research.