AUTOMATIC QUESTION GENERATION: A SYNTACTICAL APPROACH TO THE SENTENCE-TO-QUESTION GENERATION CASE

HUSAM DEEB ABDULLAH DEEB ALI

Bachelor of Science in Computer Science, University of Lethbridge, 2004

A thesis

Submitted to the School of Graduate Studies

of the University of Lethbridge

in partial fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science

University of Lethbridge

LETHBRIDGE, ALBERTA, CANADA

I dedicate this thesis to my parents, brothers and sisters and everyone

who contribute to have me achieve knowledge and character which define me today

Abstract

Humans are not often very skilled in asking good questions because of their inconsistent mind in certain situations. Thus, Question Generation (QG) and Question Answering (QA) became the two major challenges for the Natural Language Processing (NLP), Natural Language Generation (NLG), Intelligent Tutoring System, and Information Retrieval (IR) communities, recently. In this thesis, we consider a form of Sentence-to-Question generation task where given a sentence as input, the QG system would generate a set of questions for which the sentence contains, implies, or needs answers. Since the given sentence may be a complex sentence, our system generates elementary sentences from the input complex sentences using a syntactic parser. A Part of Speech (POS) tagger and a Named Entity Recognizer (NER) are used to encode necessary information. Based on the subject, verb, object and preposition information, sentences are classified in order to determine the type of questions to be generated. We conduct extensive experiments on the TREC-2007 (Question Answering Track) dataset. The scenario for the main task in the TREC-2007 QA track was that an adult, native speaker of English is looking for information about a target of interest. Using the given target, we filter out the important sentences from the large sentence pool and generate possible questions from them. Once we generate all the questions from the sentences, we perform a recall-based evaluation. That is, we count the overlap of our system generated questions with the given questions in the TREC dataset. For a topic, we get a recall 1.0 if all the given TREC questions are generated by our QG system and 0.0 if opposite. To validate the performance of our QG system, we took part in the First Question Generation Shared Task Evaluation Challenge, QGSTEC in 2010. Experimental analysis and evaluation results along with a comparison of different participants of QGSTEC'2010 show potential significance of our QG system.

Acknowledgments

There are certain individuals that keep marks on the path of our life, in a way or another, those individuals stay in our memory, and those people are the ones that contribute to shape our present and future.

This study was not to be completed without the great support and help from so many individuals, but words cannot represent the contribution, support and patience of my supervisor Dr. Yllias Chali, thanking his availability, willingness to help, directing in different levels throughout this thesis; his support make any way of saying how much gratitude I have for him, beyond what words can describe.

My supervisors committee (Dr. Yllias Chali, Dr. Sajad Zahir, Dr. Wendy Osborn) who evaluate my progress, the computer science faculty members at university of Lethbridge, and Professor Brian Dobing from the Management Information System faculty had provide lots of support that will always be remembered.

Kathy Schrage and Lorie Peter at the Graduate Department were ones that had a considerable support and kindness.

My family and my wife great support played a key role toward the completion of this thesis, they have been there in the ups and downs even though the distance was far their hearts and minds were always near.

My classmate Sadid A. Hasan has been a great support and a special individual that comes in my life path.

If I did not mention some names, that do not mean they are forgotten because they all belong to my friends circle who are many, Lethbridge communities, were there when needed, I appreciate their support.

The ability to dedicate a time for this graduate degree was by the generosity of ZI Corporation management and Linguistics Department team, where I was given the chance to work from home when needed to be in Lethbridge.

And the first to be thankful for is God (Allah, The Lord …) whom without his help and his will this would have never been achieved.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Humans are curious by nature. They ask questions to satisfy their never-ending quest for knowledge. For instance, students ask questions to learn more from their teachers, teachers ask questions to help themselves evaluate performance of the students, and our day-to-day lives involve asking questions in conversations, dialogues to render a meaningful co-operative society. To be straightforward, questions are the significant constituent of countless learning interactions from one-to-one tutoring sessions to extensive assessments in addition to real life discussions (Heilman, 2011).

One noticeable fact is that humans are not often very skilled in asking good questions because of their inconsistent mind in certain situations. It has been found that most people have trouble to identify their own knowledge deficiency (Rus & Graesser, 2009). This becomes our main motivation to automate the generation of questions with the hope that the potential benefits from an automated QG system could assist humans in meeting their useful inquiry needs.

Question Generation (QG) and Question Answering (QA) became the two major challenges for natural language understanding communities, recently (Yao, 2010; Heilman, 2011). QG has turned into an essential element of learning environments, help systems, information seeking systems, etc (Lauer et al., 1992; Graesser et al., 2001). Considerable interest from the Natural Language Processing (NLP), Natural Language Generation (NLG), Intelligent Tutoring System, and Information Retrieval (IR) communities have currently identified the Text-to-Question generation task as a promising candidate for the shared task (Rus & Graesser, 2009). In the Text-to-Question generation task, a QG system is given a text (such as a word, a set of words, a single sentence, a text, a set of texts, a stretch of conversational discourse, an inadequate question, and so on), and its goal would be to generate a set of questions for which the text contains answers.

The task of generating a question about a given text can be typically decomposed into three subtasks. First, given the source text, a content selection step is necessary to select a target to ask about, such as the desired answer. Second, given a target answer, an appropriate question type is selected, i.e., the form of question to ask is determined. Third, given the content, and question type, the actual question is constructed.

In this thesis, we consider a form of Text-to-Question generation task, where the input texts are sentences. The QG system would then generate a set of questions for which the sentence contains, implies, or needs answers. Since the given sentence may be a complex sentence, the system will generate elementary sentences, from the input complex sentences, using a syntactic parser. A part of speech tagger and a named entity recognizer are used to encode needed information. Based on the subject, verb, object and preposition the sentence will be classified, in order to determine the type of questions that can possibly be generated from this sentence.

Primarily, we conduct extensive experiments on the TREC-2007 (Question Answering Track) (http://trec.nist.gov/data/qamain.html) dataset. The goal of the TREC Question Answering (QA) track is to foster research on systems that directly return answers, rather than documents containing answers, in response to a natural language question. The scenario for the main task in the TREC-2007 QA track was that an adult, native speaker of English is looking for information about a target of interest (Dang et al, 2007). The target could be a person, organization, thing, or event. The user was assumed to be an “average" reader of U.S. newspapers. The main task required systems to provide answers to a series of related questions. A question series, which focused on a target, consisted of several factoid questions, one or two list questions, and exactly one other question. We use these data to act oppositely in this research. That is, using the given target, we filter out the important sentences from the large sentence pool and generate possible questions from them. So, from this perspective we treat our Sentence-to-Question generation system as target-driven.

For this research we consider the factoid type questions only. A factoid question can be any of these types: “What", “Where", “When", “Who", and “How". For example, considering “WWE" as target, we can generate these questions: “Who is the chairman of WWE?”, “Who is the chief executive of WWE?", “Where is WWE headquartered?", “What is “WWE" short for?", “WWE evolved from what earlier organization?", “What cable network airs WWE?", etc.

The TREC dataset provides textual data that contains differently structured sentences along with complex sentences with multiple clauses. To simplify the process of question generation, we extract elementary sentences from the given data using syntactic information. By considering the Part of Speech (POS) tagging and Named Entity (NE) tagging of a sentence, we get the parts of speech related information and entities that are used to generate relevant questions. Once we generate all the questions from the sentences, we perform a recall based evaluation. That is, we count the overlap of our system generated questions with the given questions (in the TREC dataset). For a topic, we get a recall 1.0 if all the given TREC questions are generated by our QG system and 0.0 if opposite.

To validate the performance of our QG system, we took part in the First Question Generation Shared Task Evaluation Challenge, QG-STEC in 2010. QG-STEC-2010 offered two important tasks: Question Generation from Paragraphs (Task A) and Question Generation from Sentences (Task B). We participated in Task B to evaluate our QG system. Experimental analysis and evaluation results along with a comparison of different participants of QG-STEC'2010 are presented in details in the last part of this thesis that shows a promising direction to impact the QG research community.

1.2 Overview of the Thesis

The block diagram in Figure 1.1 depicts the basic architecture of this thesis. Since the sentences might have a complex structure with multiple clauses, it would be difficult to generate accurate questions from the complex sentences. Therefore, using syntactic information we simplify the process by extracting elementary sentences from the complex sentences. In the next phase, based on the sentences subject, verb, object and preposition we classify the sentences to determine the possible type of questions that will be generated. In the final phase we generate questions based on the classified sentences and the output is a set of questions generated.

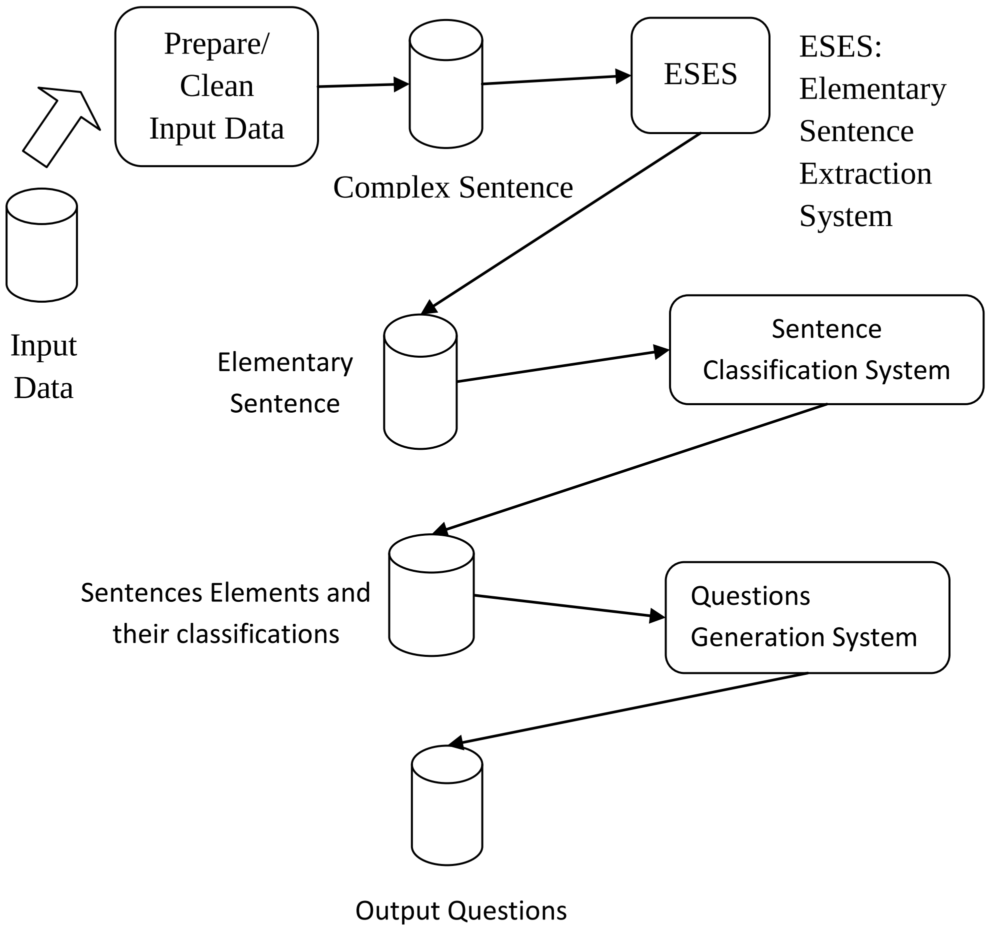

Prepare/ Clean Input Data
Complex Sentence
ESES
ESES: Elementary Sentence Extraction System
Input Data
Elementary Sentence
Sentence Classification System
Sentences Elements and their classifications
Questions Generation System
Output Questions

Figure 1 Overview diagram of our QG system

1.3 Research Questions

The research questions that we address in this thesis are:

1. Given a complex sentence, how to generate corresponding simple sentences from it?
2. How to classify the sentences in terms of different factoid question types: *Who*, *Whom*, *Which*, *What*, *When*, *Where*, *How many*—so that appropriate questions can be generated?
3. How to generate questions from a simple sentence?
4. How to evaluate generated questions in terms of precision and recall?

1.4 Contributions

This thesis contributes to the sentence-to-question generation task in the following ways:

Sentence Simplification To generate more accurate questions, we extract the elementary sentences from the complex sentences. To attain this we syntactically parse each complex sentence using Charniak parser (Available at ftp://ftp.cs.brown.edu/pub/nlparser/) to construct a syntactic tree representation from the bracketed representation of the parsed sentence. We use the depth first algorithm, to read the tree nodes and leaves, which help us construct the elementary sentences, were we maintain if the phrases to be joined are sequentially correct with the respect of the sentence syntactical structure.

Sentence Classification Based on the associated POS and NE tagged information; from each elementary sentence we get the subject, object, preposition and verb which are used to classify the sentences. We use two simple classifiers in this module. Inspired by the idea proposed for question classification in (Li & Roth, 2002), our first classifier classifies the sentence into fine classes (Fine Classifier) and the second classifies the sentences into coarse classes (Coarse Classifier) that are used to identify what type of questions should be generated from one simple sentence.

Question Generation In this module, the input is the elements of a sentence with its classification coarse classes, the verbs (with their stems and tense information). We use a manually predefined set of interaction rules for the relation between the coarse classes. Based on these interaction rules and the verbs we construct the questions with a set of predefined template questions.

Evaluation and Analysis We use a precision and recall based evaluation technique to extensively investigate the performance of our QG system for the TREC-2007 data experiment. On the other hand, we present the QG-STEC' 2010 evaluation methods and results with deep analysis and compare our system with the participants of the competition.

1.5 Thesis Outline

We give a chapter-by-chapter outline of the remainder of this thesis in this section.

Chapter 2 We give a detailed description about *automatic question generation* and related works done in this area.

Chapter 3 We discuss some English word classes, sentence structure, syntactical parsing and some of the different algorithms used for the purpose of creating a parse trees to represent the sentence structures.

Chapter 4 We present question types and question classification, complex vs. simple questions.

Chapter 5 We discuss the data preparation, and implementation of the system.

Chapter 6 We discuss a syntactical approach, and the steps and an algorithm that is used. The steps show how we use the algorithm to extract elementary sentences from complex sentences, and to generate questions, the question generated are based on the elementary sentences that we extract from the complex sentences.

Chapter 7 We present the experimental results and analyze them thoroughly.

Chapter 8 We conclude the thesis by identifying some future directions of our research.

1.6 Published Work

Some of the material presented in this thesis has been previously published in [9, 10].

- Automation of Question Generation From Sentence, (Ali H., Chali Y. and Hasan S. 2010) In Proceedings of the Third Workshop on Question Generation, Pittsburgh, Pennsylvania.

- Automatic Question Generation from Sentences: a Preliminary Approach (Ali H., Chali Y. and Hasan S. 2010) In Proceedings of the Conference on Traitement Automatique de la Langue Naturelle, Montreal, Canada.

Chapter 2

Automatic Question Generation & Question Classification

2.1 Introduction

The task of generating reasonable questions from a text, Question Generation (QG), is a relatively new research topic that attracted considerable attention from the Psycholinguistics, Discourse and Dialogue, Natural Language Processing (NLP), Natural Language Generation (NLG), Intelligent Tutoring System, and Information Retrieval (IR) communities recently (Rus & Graesser, 2009). Available studies revealed that humans are not very skilled in asking good questions. Therefore, they would benefit from automated QG systems to assist them in meeting their inquiry needs (Rus & Graesser, 2009).

A QG system can be helpful in asking learners questions based on learning materials in order to check their accomplishment or help students focus on the key aspects in their study. It can also help tutors to prepare questions for learners to evaluate their learning capacity (Heilman, 2011). A QG component can be added into a Question Answering (QA) system that could use some pre-defined question-answer pairs to provide QA services (Yao, 2010).

The Text-to-Question generation task has been identified as a promising candidate for shared tasks (Rus & Graesser, 2009). In the Text-to-Question generation task, a QG system is given a text, and its goal would be to generate a set of questions for which the text contains answers. This task can be typically decomposed into three subtasks. First, given the source text, a content selection step is necessary to select a target to ask about, such as the desired answer. Second, given a target answer, an appropriate question type is selected, i.e., the form of question to ask is determined. Third, given the content, and question type, the actual question is constructed.

2.2 Related Work

Recently, tackling Question Generation (QG) in the field of computational linguistics has got immense attention from the researchers. Twenty years ago it would take hours or weeks to receive answers to the same questions as a person hunted through documents in a library. In the future, electronic textbooks and information sources will be main stream and they will be accompanied by sophisticated question asking and answering facilities (Rus & Graesser, 2009).

In (Andernucci & Sneiders, 2005), they introduced a template-based approach to generate questions on four types of entities. The authors in (McGough et al., 2001) used WTML (Web Testing Markup Language), which is an extension of HTML, to solve the problem of presenting students with dynamically generated browser-based exams with significant engineering mathematics content. In (Wang et al, 2008), they generated the questions automatically based on question templates that are created by training on many medical articles. In (Brown et al., 2005), an interesting approach was described to automatically generating questions for vocabulary assessment. Using data from WordNet, they generated 6 types of vocabulary questions of several forms, including wordbank and multiple-choice. In (Wei C. & Mostow, 2009), the authors presented an approach to generate questions from informational text, which was used to generate modeling and scaffolding instruction for the reading comprehension strategy of self-questioning. They proposed three types of questions for informational text: questions about conditional context, questions about temporal information, and questions about possibility and necessity.

The history suggests that the research on QG is still in a preliminary stage. In (Rus & Graesser, 2009), they presented some concrete plans to drive this area of research onward that follows a road map from generating shallow questions to deep questions, from direct questions on explicit information (such as generating “Who likes Apples?” from “Tom likes Apples.”) to indirect questions on inherent information (such as generating “What did Tom eat?” from “Tom went to a restaurant.”), and from using single sentences as sources to using paragraphs or even multiple texts.

The first Question Generation Shared Task and Evaluation Challenge (QGSTEC)-2010 is one of the efforts of the QG community to approach two tasks in the form of challenge and competition. The first task focuses on evaluating the generation of questions from paragraphs and the second on the generation of questions from sentences. QGSTEC follows a long tradition of Shared Task Evaluation Challenge (STEC) in Natural Language Processing such as various tracks at the Text REtrieval Conference (TREC; http://trec.nist.gov): the Question Answering track, the semantic evaluation challenges under the SENSEVAL umbrella (www.senseval.org), or the annual tasks run by the

Conference on Natural Language Learning (CoNLL; http://www.cnts.ua.ac.be/conll/) (Boyer, K. E. and Piwek, P, 2010).

2.3 Question Types

According to the usage principle, questions can be classified into different categories. For instance, in the question answering track of the Text Retrieval Conference-TREC (http://trec.nist.gov/), questions fall under three types: factoid, list and other. Factoid questions (such as "How high is the Mount Everest?") ask for fact-based answers and list questions (such as "Which countries did Barack Obama visit in 2010?") ask for a set of answer terms. In TREC, the answer to the "Other" question was required to be interesting information about the target not covered by the preceding questions in the series of factoid and list questions. In terms of target complexity, the type of QG can be divided into *deep* QG and *shallow* QG (Rus & Graesser 2009). Deep QG generates deep questions that involves more logical thinking (such as why, why not, what-if, what-if-not and how questions) and shallow QG generates shallow questions that focus more on facts (such as who, what, when, where, which, how many/much and yes/no questions).

2.4 Questions Classification

2.4.1 Introduction

Questions are a way of asking for information, but are all questions the same? Let us see these questions:

1. Where is Alberta?
2. What minerals are Alberta rich of?
3. Describe the process of separation of Oil from sand in Alberta?
4. How much does it cost to produce 1 gallon of Oil in Alberta?

By answering these questions we can retrieve lots of information that can lead us to understand the location of Alberta and what is it rich of. Also we can get a basic understanding of the process they use to extract the Oil from the sand in Alberta, also the cost of the process per gallon.

Hence questions are one important way of gaining knowledge. In this chapter we will discuss the questions from the perspective of its complexity, especially when discussing automated Question Answering systems. This also will get more attention when we discuss the question generation systems which can be used for automation of examinations for students, and other purposes where the need of the automation of question generation will be of a great help.

2.4.2 Complex vs. Simple Question

Are all questions the same?

What make questions differ from each other?

For questions number 1, 2 and 4 in section 2.1, we can answer them with one word or a simple sentence. Question 1, the answer is Canada, question 2 the answer is Oil and question 3 the answer can be a figure 50 dollars. But trying to answer question 3 using a word or one sentence will not cover the basic elements of the answer, since the process does include several steps that are different from each other.

2.4.3 Simple Question

Simple questions are questions that can be answered in a word or simple sentence. In general we call the Factoid questions simple questions, since the answer will be a fact Alberta is in Canada, Alberta is rich of Oil sand.

But are all questions have both classifications? How to classify a question, to be a factoid, simple question or a complex question?

Factoid questions are the questions that can be answered with one sentence maximum and in most of the cases they start with this:

- Who ...?
- Whom ... ?

- When ...?
- Where ...?
- Which ...?
- What ...?
- How many ...?
- How much ... ?

But that not all factoid questions let’s consider this:

- Are you a student?
- Is it noon yet?
- Can you cook?

These are also questions that can be answered with one sentence or even one word “yes” or “no”.

2.4.4 Complex Question

Complex questions can be short questions or long, but what matter as we discuss before that the answer type. Let us consider these examples:

- Describe Lethbridge?
- What did you do today?
- How does medicine made?
- Explain quantum theory?

Let us consider an answer for the first question “Describe Lethbridge”:

> “Lethbridge is small size town with population of 80,000 people. that is located south of Calgary, in Alberta; Lethbridge is a windy town, the warm wind from south west help melting the snow in winter; Old Man River cross through the town.”

This answer can be extended longer and can have different types such as:

- Numeric type answer: number of people living in Lethbridge
- Direction type answer: south of Calgary
- Entity type answer: Old Man River.

2.5 Summary

In this chapter, we describe *automatic question generation* in terms of its introduction to different research communities along with related work in this field. We also discussed about different types of questions and QG categories. Next chapter will focus on discourse structure and syntactic parsing in detail.

In this chapter we introduce the Questions Types, discuss the differences between the different types of questions, the complexity, how the complexity of a question can impact the automated systems interacting or generating questions of each type.

Chapter 3

Syntactic Parsing

3.1 Introduction

Parsing is one of the methods used to read and recognise textual entities, or even a programming language code.

Parsing is defined by Daniel Jurafsky and James H. Martin as taking an input and producing some sort of linguistic structure for it (Jurafsky & Martin , 2009, p 45).

In this chapter we will discuss some English word classes, Part of Speech, Tokenization, Stemming, Partial Parsing, Syntactical Parsing, evaluation of parsers, Human Parsing and we will conclude by a summary about the different parsers.

3.2 Some English Word Classes

English Words are belonging to different classes that distinguish them. The distinguishing of words happened in the structure of the sentences that they fall into. Some words fall into more than one class depending on the location in the textual entity.

Since an English word is not just a noun or a verb, the need to understand the differences that distinguish an English word from another in a sentence may help in producing a better system that is dealing with human interaction software's.

We will discuss some of the English Word Classes that are commonly encountered in most of the human interaction applications. The definitions of this word classes are taken from what Daniel Jurafsky and James H. Martin defined then in their book in the chapter that is discussing the Part of Speech Tagging (Jurafsky & Martin , 2009, p 124 to 129).

The part-of-speech consists of two major classes, open class type and closed class type. We will discuss the class types and their sections.

Open class type:

> **Noun:** is the name given to the syntactic class in which the words for most people, places, or things occur.
>
> The Noun type is grouped in four groups:
>
> > **Proper Noun:** they are names of specific persons or entities; generally they are not preceded by articles and also they are usually capitalized; such as Regina, Alberta and IBM.
> >
> > **Common Noun:** they are names of entities that are generally preceded by articles and they are usually not capitalized; such as the book and the pen.
> >
> > Common nouns are divided into count nouns and mass nouns.

Count Nouns: they are nouns that allow grammatical enumeration; which means they can occur in both singular and plural (cat/cats, book/books) and they can be counted (one cat, two cats).

Mass Nouns: they represent something that is conceptualized as a homogeneous group (snow, salt, sugar). Mass nouns can appear without articles (sugar is white). But singular count noun needs an article (the goat is black).

Verb: the verb class includes most of the words referring to actions and processes, including main verbs like draw, draw, provide, differ and go.

Adjective: this class includes many terms that describe properties or qualities.

Adverb: is a class that is divided in four sections:

Directional adverbs or Locative adverbs: it's an adverb that specifies direction or location of some action like home, here, uphill and there etc.

Degree adverbs: this type of adverbs specifies the extent of some actions, process or property like extremely, very, somewhat ... etc.

Manner adverbs: this type of adverb describes the manner of some action or process like slowly, delicately ... etc.

Temporal adverbs: describe the time that some action or event took place like yesterday, Monday, Friday ... etc.

Closed class type:

Prepositions: occur before noun phrases; examples: on, under, over, near, by, at, from, to, with ... etc. a sample phrase will be: *on* time, *beside* himself.

Particles: they are a word that resembles a preposition or an adverb and is used in combination with a verb. Examples: up, down, on, off, in, out, at, by ... etc. a sample usage will be: turn *down*, rule *out*.

Determiners: they occur with nouns. Like a, an, the, this, that ... etc. a sample usage will be: *that* book, *this* cat, *an* apple.

Conjunctions: they are words that join two phrases, clauses or sentences. Like and, but, or, as, if, when. A sample usage: I thought *that* you went home.

Pronouns: they are words that act as a kind of shorthand for referring to some noun phrase of entity or event. Pronouns can be divided in 3 different types:

Personal pronouns: words that refer to a person or entity. Like I, she, he, it, me ... etc. a sample usage will be: she is Sarah, he is Tom.

Possessive pronouns: they indicate actual possession or an abstract relation between a person and an object. Like my, your, his, their ... etc. a sample usage will be: my book, his table.

Wh-pronouns: they are words that are used in a question form or act as a complementizers form. Like what, whom, who, who-ever ... etc. a sample usage will be: Sarah, who drives our school bus.

Auxiliary verbs: they are words (usually verbs) that mark certain semantic features of a main verb, including whether an action takes place in the present, past or future (tense), whether it is completed (aspect), whether it is negated (polarity), and whether an action is necessary, possible, suggested, desired ... etc. which represent (mood). Like: can, may, should, are, is (be), were. A sample usage will be: can eat, should go to school, they are eating, we were playing.

Interjections: oh, ah, hey, um.

Negatives: no, not

Politeness markers: please, thank you.

Greetings: hello, goodbye.

Numerals: they are words that represent numbers or numerical value. Like one, five, first, fourth ... etc.

Different languages may have different classes, but for every language there is a class where every word should belong to. Using these classes in a system, the system will be able to recognize the words class and tag them accordingly, which help improving the quality of systems that deal with human software interactions. This assuming that we have a perfect word class's recognizer; however in the real programming world, there will be always some words that will not be recognized, for so many reasons, such as:

- The corpus that is used to train the class's recognizer does not include a particular word.

- New words are added to languages in regular basis with new technology, products, intellectual interaction between human speaking different languages may produce new words.

3.3 Tokenization and Sentence Segmentation

Textual structures have segmentations of words that are separated by white space or other word separators; these word separators differ based on the language that these word segments belong to. Word separating alone cannot help understand the textual entity; since as an example a sentence end could be the mark of starting a new sentence too.

Tokenization is defined as the task for separating out (tokenizing) words from running text. (Jurafsky & Martin , 2009, P 47)

Finding the start of a sentence and the end of it is a crucial step when processing textual entities. Segmenting a text into sentences is generally based on punctuations (Jurafsky & Martin , 2009, P 69).

Certain punctuations tend to mark sentence boundaries, some examples of the punctuations that have this tendency are (periods, question marks, and exclamation points).

Question marks and exclamation points relatively are unambiguous sentence boundaries, while periods are an ambiguous sentence boundary marker, because it is used for representing abbreviations such as Mr. or Inc.

The ambiguous sentence boundary determination, adds to the difficulty to improve the quality of processing textual entities, improving the quality of processing textual entities is a step for software's that deal with human software interactions.

3.4 Stemming

The words are units that a sentence contains and built by. But are the words as units enough for textual processing? Is cat and cats the same? Are eat, eats, eating, ate and eaten the same?

This is where understanding the word stem is becoming a helpful and a needed step. Being able to get the word stem, helps make use of words in the sentences and the textual structures, when manipulating the sentence structure for a particular need such as generating questions that preserve the verb tens.

To understand stemming and reach to a definition for it we need to understand some other definitions first:

> **Morphemes:** are the small meaning-bearing units that build up words example: the word cats consists of two: the morpheme cat and the morpheme –s (Jurafsky & Martin , 2009, P 47)

Prefix: is the part of the word that precede the stem like: un in unable.

Suffix: is the part of the word that follows the stem like: ing in eating.

Infix: is the part of the word that is inserted inside the stem, in English we find it in some dialects in which the taboo morphemes like “bl**dy” are inserted on the middle of the stem like: abso-bl**dy-lutely (McCawley,1978) (Jurafsky & Martin , 2009, P 47).

Circumfix: is a combination of both Prefix and Suffix. English language does not really have that but in German the past participle of some verbs is formed by adding ge- to the beginning of the stem and –t to the end (Jurafsky & Martin , 2009, P 47)

This leads us to define the stem to be the main morpheme of the word supplying the main meaning (Jurafsky & Martin , 2009, P 47)

3.5 Part of Speech Tagging

In this section we will discuss the Part-Of-Speech tagging; which gave a tag for each word; to identify and differentiate it from other words in the sentence.

For that we have a different tagset that was developed for English based on corpus that was collected from different sources such as the 87-tag tagset which was used for the brown corpus (Francis, 1979; Francis and Kučera, 1982) ; the brown corpus is a million-word collection from 500 different genres (newspapers, novels, non-fiction, academic, etc.) (Kučera and Francis, 1967; Francis, 1979; Francis and Kučera, 1982); the tagset was developed at Brown University.

A sample of the tagset is shown in table 1

Tag	Description	Example
NN	(common) singular or mass noun	Time, world, work, school
NP	Singular proper noun	France, China, Congress
VB	Verb, base form	Make, try, drop
VBD	Verb, past tense	Walked, cooked
JJR	Comparative adjectives	Taller, smaller, shorter

Table 1 sample of the tagset

But in practice other smaller tagsets are used, such as Penn Treebank which consist of 45-tags and C5 tagset which contain 61 tags. A sample of the Penn Treebank tagset is in table 2

Tag	Description	Example
CC	Coordin. Conjunction	And, or, but
CD	Cardinal number	Five, seven, ten
JJ	Adjective	Yellow, green, small
DT	Determiner	A, the
NNP	Proper noun, singular	IBM, Microsoft

Table 2 sample of the Penn Treebank tagset

3.6 Ambiguity

Ambiguity is one of the challenges that face software's that deals with natural language processing, ambiguity is one of the contributors towered how high or low is the quality of the software. Ambiguity can come in different forms and types; in this section we will discuss some of these types. Ambiguity forms from sentence structure, it also can form from lexically or semantically ambiguous words and pragmatics where a phrase should not be understood literally.

We will discuss with examples some of these ambiguity types; which are some of the challenges that Natural Language Processing (NLP) software's may face:

Structural ambiguity: it's a type of ambiguity that mostly related to the structure of the sentence being used, where the parser can construct different valid trees to represent the sentence structure.

"I had Cold orange juice and water." This sentence can be structurally built in two different ways:

- I had [cold orange juice] and water
- I had cold [orange juice and water]

"Show me how perfect actors acts." Also can be built in two ways:

- Show me [how] [perfect actors] [acts]
- Show me [how perfect] [actors acts]

"Look how fast runners ran."

- Look [how] [fast runners] [ran]
- Look [how fast] [runners ran]

"I noticed 5 dogs running along the road."

- I noticed [5 dogs] [running along the road]
- I noticed [5 dogs running along the road]

Lexical level or Semantic ambiguity: where a word can have different meaning depending on the sentence it appears in, one of the well-known examples is the word book, the word book:

- Math book (Noun)
- Book this flight (Verb)

Other example the word bank:

"Sam waited at the bank."

- Same waited at the bank [financial institute]
- Sam waited at the bank [bank of a river]
- Sam waited at the bank [a bank seat]

Bank can be a verb like:

- I Bank my money at a safe.

Ambiguity as a problem was discussed in several researches and different solutions have been implemented for it.

3.7 Syntactical Parsing

Syntactical parsing is one of the ways that is used to represent a textual entity in a tree structure based on a grammar rules and an input sentence.

If we consider the grammar rules below (Jurafsky & Martin , 2009, P 428):

Grammar	Lexicon
S → NP VB	Det → that | this | a
S → Aux NP VB	Noun → book | flight | meal | money
S → VP	Verb → book | include | prefer
NP → Pronoun	Pronoun → I | she | me
NP → Proper-Noun	Proper-Noun → Houston | NWA
NP → Det Nominal	Aux → does
Nominal → Noun	Preposition → from | to | on | near | through
Nominal → Nominal Noun	
Nominal → Nominal PP	
VP → Verb	
VP → Verb NP	
VP → Verb NP PP	
VP → Verb PP	
VP → VP PP	
PP → Prepositional NP	

Table 3 Sample grammar rules

We will use the grammar rules in table 3 to discuss some of the parsing methods.

3.7.1 Top-Down Parsing

Top down parsers searches for a parse tree by trying to build from the root node S down to the leaves (Jurafsky & Martin , 2009, p 429). The algorithm will look for all the trees that have the root as their parent, using the information about the possible trees under the root, the parser will build trees in parallel. ‘S’ represent the start of the sentence. The trees that have ‘S’ as the root are basically the grammar rules that are provided for the parser. Then the parser will expand the new parallel trees using the grammar into a new set of parallel trees.

If we take the sentence below as an example that is used by (Jurafsky & Martin , 2009, p 431):

Book that flight.

The word book can be a noun or a verb, the parser will build a parallel trees represent the all the cases that the grammar have for a sentence which its root is an ‘S’. We will discuss the two possible cases and one of their parse trees tree’s for the case of the parser considering the word book as noun and the other case where it consider the word book as a verb.

For the case where the word book is considered a noun we will use the grammar rule:

S $\rightarrow$ Nominal NP

For the case where the word book is considered a verb we will use the grammar rule:

S → VP

By depicting one of the possible parallel trees at each step using the rule:

S → Nominal NP

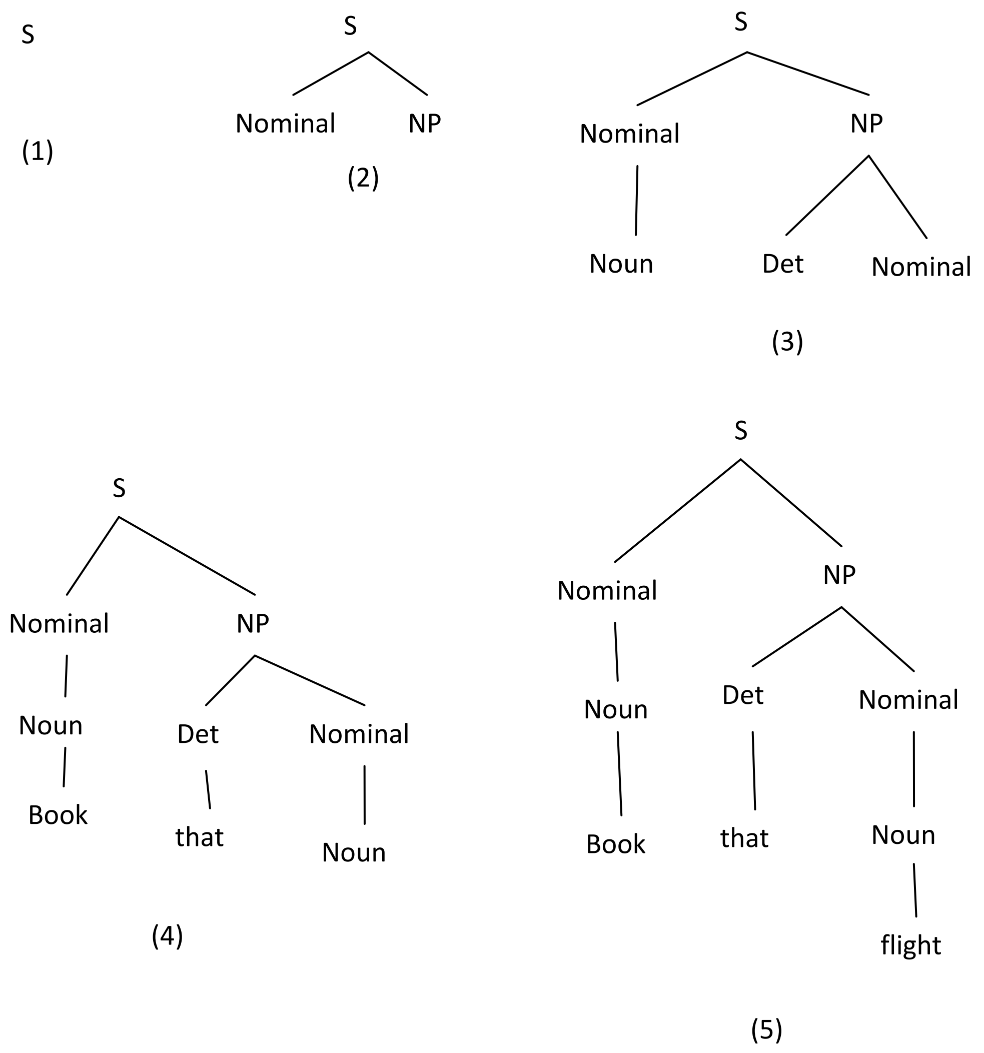


Fig 4.1

By depicting one of the possible parallel trees at each step using the rule:

S → VP

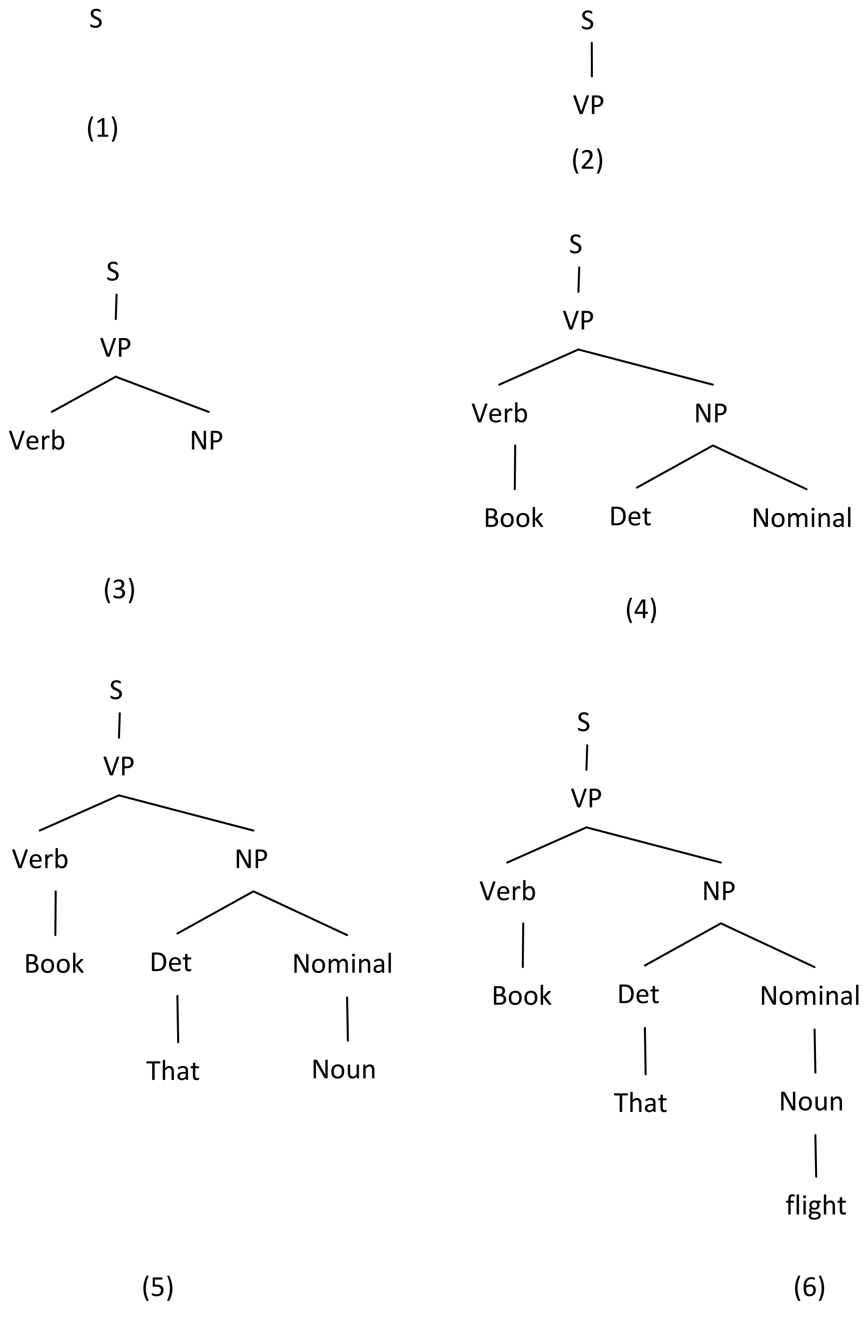


Fig 4.2

3.7.2 Bottom-Up Parsing

In bottom-up parsing the parser starts with the words of the input, and tries to build trees from the word up, again by applying rules from the grammar one at a time. The parser is successful if the parser succeeds in building a tree rooted in the start symbol S that covers all of the input. (Jurafsky & Martin , 2009, p 429).

The algorithm will look for all grammar rules that can be a root for each level of building the tree up, the parser will build trees in parallel one at a time.

If we take the sentence below as an example, while considering the word book as noun:

Book that flight.

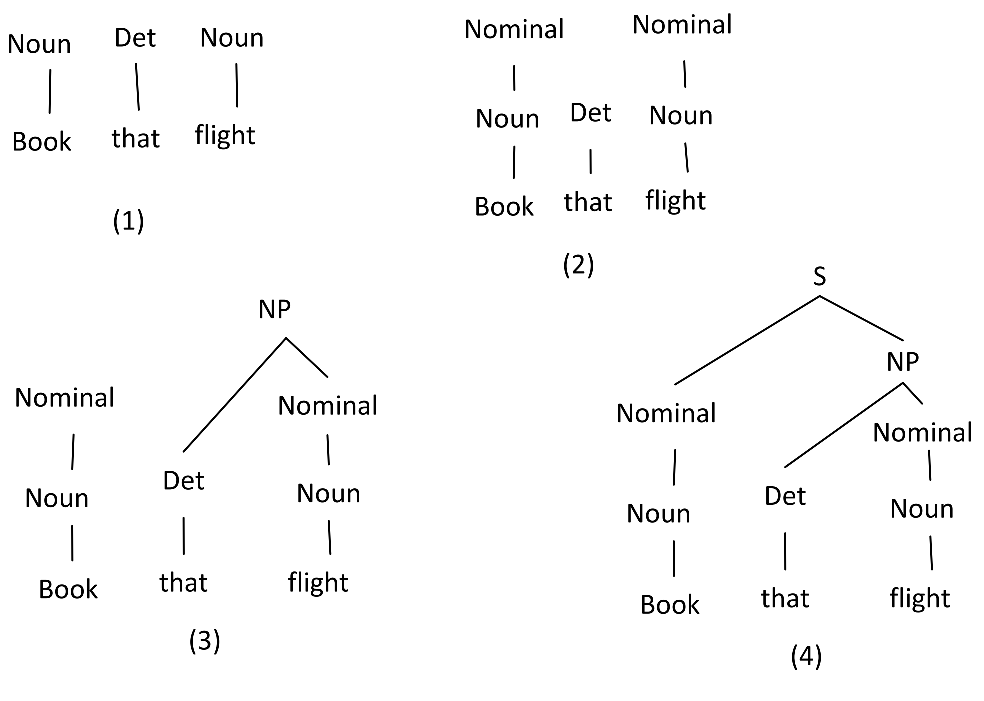
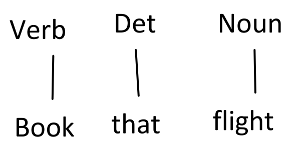
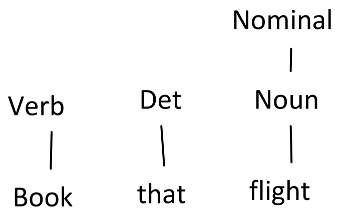


Fig 4.3

By considering the word ‘Book’ as verb:



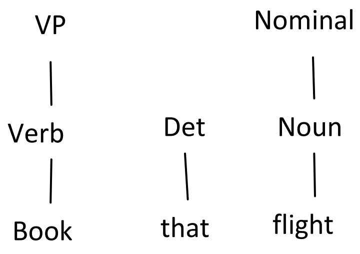
(1)



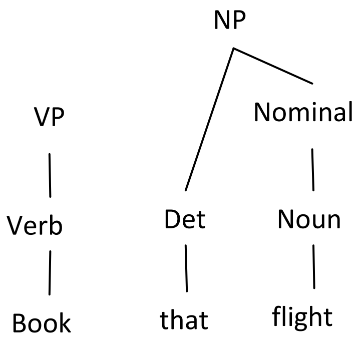
(2)

From step 2 we can have two possibilities:

Case A



(3)



(4)

Fail since there is no grammar rule that have

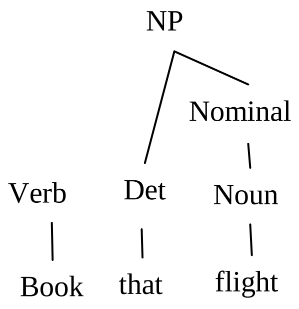
VP NP

as aright side

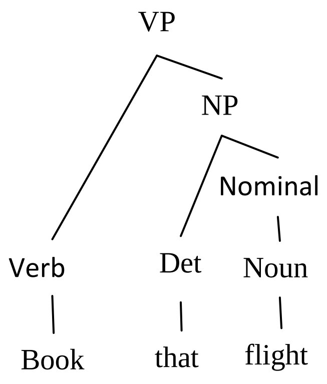
(5)

Fig 4.4

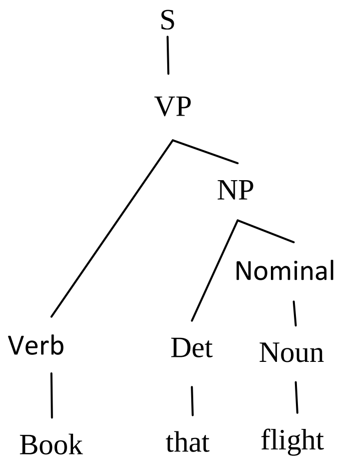
Case B



(3)



(4)



(5)

Fig 4.5

In case B we reach to a parse tree that have S as the root, hence this is a successful parse that can be returned by the algorithm.

3.7.3 The Early Algorithm

This algorithm use the Top Down approach, based on a given input sentence and a given grammar rule set, this algorithm is a more of a recognizer than a parser.

```
Function EARLEY-PARSE(words, grammar) returns chart
    ENQUEUE(($\gamma \rightarrow \bullet S$, [0,0]), CHART[0])
    for i ← from 0 to LENGTH(words) do
      for each state in chart[i] do
         if INCOMPLETE?(state) and NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
        elseif INCOMPLETE?(state) and NEXT-CAT(state) is a part of speech then
        SCANNER(state)
        else
        COMPLETER(state)
        end
    end
return (chart)

procedure PREDICTOR(($A \rightarrow \alpha \bullet B \beta$, [i,j]))
    for each ($B \rightarrow \gamma$) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE($B \rightarrow \bullet \gamma$ ,[j,j] ), chart[j])
    end

procedure SCANNER(($A \rightarrow \alpha \bullet B \beta$, [i,j]))
    if $B \subset$ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE ($B \rightarrow$ word[j],[j,j+1]),chart[j+1])

procedure COMPLETER(($B \rightarrow \gamma \bullet$, [j,k]))
    for each ($A \rightarrow \alpha \bullet B \beta$, [i,j]) in CHART(j) do
        ENQUEUE(($A \rightarrow \alpha\ B \bullet \beta$, [i,k] ), chart[k])
    end

procedure ENQUEUE(state, chart-entry)
    if state is not already in chart-entry then
        PUSH(state, chart-entry)
end
```

Figure 2 The Early Algorithm

We will follow the algorithm in a simple sentence:

Close that window

At first we have a grammar as follow:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition PP

Figure 3 A Sample grammar to be used with the Early Algorithm

The first step will be starting the first Chart and initialize it with a dummy state

$\gamma \rightarrow \bullet S, [0,0]$

Now the algorithm will run and call the PREDICTOR with the grammar since we are not in a complete state nor the state we are at is a part of speech state, the final look of

Chart [0] after all the states have been processed will be:

S0	γ →•S	[0,0]	Dummy start state
S1	S → •NP VP	[0,0]	PREDICTOR
S2	S → •Aux NP VP	[0,0]	PREDICTOR
S3	S → •VP	[0,0]	PREDICTOR
S4	NP → •Pronoun	[0,0]	PREDICTOR
S5	NP → •Proper-Noun	[0,0]	PREDICTOR
S6	NP → •Det Nominal	[0,0]	PREDICTOR
S7	VP → •Verb	[0,0]	PREDICTOR
S8	VP → •Verb NP	[0,0]	PREDICTOR
S9	VP → •Verb NP PP	[0,0]	PREDICTOR
S10	VP → •Verb PP	[0,0]	PREDICTOR
S11	VP → •VP PP	[0,0]	PREDICTOR

Chart [0]

Now the algorithm start with a new Chart [1], using the scanner will read the first word and process the possible cases in the grammar based on the first word part of speech; the "•" will be placed after the part of speech tag to indicate that it is a complete state for this word; after this iteration of the algorithm Chart [1] will looks like this:

S12	Verb →Close•	[0,1]	SCANNER
S13	VP → Verb•	[0,1]	COMPLETER
S14	VP → Verb• NP	[0,1]	COMPLETER
S15	VP → Verb• NP PP	[0,1]	COMPLETER
S16	VP → Verb• PP	[0,1]	COMPLETER
S17	S → VP•	[0,1]	COMPLETER
S18	VP → VP• PP	[0,1]	COMPLETER
S19	NP → •Pronoun	[1,1]	PREDICTOR
S20	NP → •Proper-Noun	[1,1]	PREDICTOR
S21	NP → •Det Nominal	[1,1]	PREDICTOR
S22	PP → •Prep NP	[1,1]	PREDICTOR

Chart [1]

In this step of the iteration of the algorithm the word was known to be a Verb; therefore it completes the states that start with a Verb; then it start with Chart [2] by reading the next word; and process the possible cases in the grammar based on the second word part of speech; the "•" will be placed after the part of speech tag to indicate that it is a complete state for this word. After this iteration of the algorithm Chart [2] will look like this:

S23	D et →that•	[1,2]	SCANNER
S24	NP → Det• Nominal	[1,2]	COMPLETER
S25	Nominal → • Noun	[2,2]	PREDICTOR
S26	Nominal → • Nominal Noun	[2,2]	PREDICTOR
S27	Nominal → • Nominal PP	[2,2]	PREDICTOR

Chart [2]

In this step of the iteration of the algorithm the word was known to be a Det.; therefore it completes the states that start with a Verb followed by a Det. Now it start with Chart[3] by reading the next word ; and process the possible cases in the grammar based on the third word part of speech; the "•" will be placed after the part of speech tag to indicate that it is a complete state for this word.

After this iteration of the algorithm Chart [3] will look like this:

S28	Noun →Window•	[2,3]	SCANNER
S29	Nominal → Noun •	[2,3]	COMPLETER
S30	NP → Det Nominal•	[1,3]	COMPLETER
S31	Nominal → Nominal • Noun	[2,3]	COMPLETER
S32	Nominal → Nominal • PP	[2,3]	COMPLETER
S33	VP → Verb NP•	[0,3]	COMPLETER
S34	VP → Verb NP • PP	[0,3]	COMPLETER
S35	PP → •Prep NP	[3,3]	PREDICTOR
S36	S → VP•	[0,3]	COMPLETER
S37	VP → VP• PP	[0,3]	COMPLETER

Chart [3]

Now the states that participate in final parse will be like the following:

From Chart [1]

S12	Verb →Close•	[0,1]	SCANNER

From Chart [2]

S23	D et →that•	[1,2]	SCANNER

From Chart [3]

S28	Noun →Window•	[2,3]	SCANNER
S29	Nominal → Noun •	[2,3]	COMPLETER
S30	NP → Det Nominal•	[1,3]	COMPLETER
S33	VP → Verb NP•	[0,3]	COMPLETER
S36	S → VP•	[0,3]	COMPLETER

This algorithm can be modified COMPLETER to create a pointer to the older state in a list of constituent states for the new state. Which make the Chart [3] results after the successful found of a complete parse looks like this:

S28	Noun →Window•	[2,3]	SCANNER
S29	Nominal → Noun •	[2,3]	(S28)
S30	NP → Det Nominal•	[1,3]	(S23,S29)
S33	VP → Verb NP•	[0,3]	(S12,S30)
S36	S → VP•	[0,3]	(S33)

This will create us the following tree:

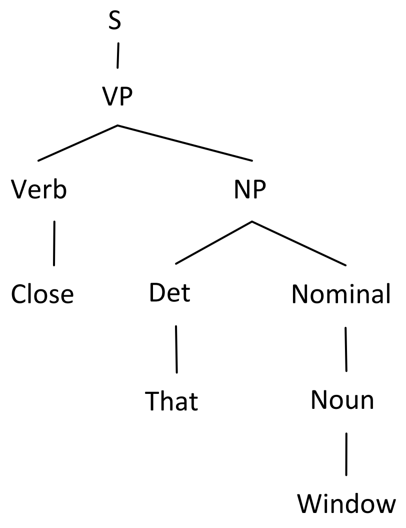


Figure 4.6

3.8 Statistical Parsing

In this type of parsers the grammar rules are associated with probability information, for each non-terminal to expand to a terminal or non-terminals, this grammar rules are then given as input for the parser along with a textual entity to be parsed.

The most commonly used probabilistic grammar is the Probabilistic context-free grammar (PCFG) (Jurafsky & Martin, 2009, P 459)

The statistical parsers get trained in a labelled data, which will help generate the probabilities for each non-terminal to expand to one of the available grammar rules provided.

By considering A as non-terminal, β as list of possibilities that A can expand to, and p as the total probability for each part of the list β, (Jurafsky & Martin, 2009, P 460).

$$\mathrm{A} \rightarrow \beta\ [p]$$

The probability can be represented as

$$P\ (\mathrm{A} \rightarrow \beta)$$

If we take all possibilities in the list β, which A can expand to, the sum of these possibilities must be 1:

$$\sum_{\beta} P(\mathrm{A} \rightarrow \beta) = 1$$

Table 5 will help us understand a sample of probabilistic grammars and their rules, we will discuss the grammar rules with their probabilities, which is discussed by (Jurafsky & Martin , 2009, P 461) .

Grammar		Lexicon
S → NP VB	[.80]	Det → that [.10] | a [.30] | the [.60]
S → Aux NP VB	[.15]	Noun → book [.10] | flight [.30] | meal [.15]
S → VP	[.05]	| money [.05] | flights [.40]
NP → Pronoun	[.35]	| dinner [.10]
NP → Proper-Noun	[.30]	Verb → book [.30] | include [.30]
NP → Det Nominal	[.20]	| prefer; [.40]
NP → Nominal	[.15]	Pronoun → I [.40] | she [.05] | me [.15]
Nominal → Noun	[.75]	| you [.40]
Nominal → Nominal Noun	[.20]	Proper-Noun → Houston [.60] | NWA [.40]
Nominal → Nominal PP	[.05]	Aux → does [.60] | can [.40]
VP → Verb	[.35]	Preposition → from [.30] | to [.30] | on [.20] |
VP → Verb NP	[.20]	near [.15] | through [.05]
VP → Verb NP PP	[.10]	
VP → Verb PP	[.15]	
VP → Verb NP NP	[.05]	
VP → VP PP	[.15]	
PP → Prepositional NP	[1.0]	

Table 4 sample of probabilistic grammars and their rules

3.8.1 The Collins Parser

Collins Parser is the statistical parser described in Michael Collins in his PhD Dissertation at University of Pennsylvania (Collins. 1999). The Collins parser is a head driven statistical model. In this module, Collins consider every right side of the grammar rules consist of three parts, a Head part (H), a Non-Terminals to the left of the Head and a Non-Terminal to the right of the Head.

3.9 Evaluating Parsers

There are different measuring methods; we will discuss the most commonly used measures in the area of information retrieval.

- Precision
- Recall
- F-measure

From domain to a domain these measures may vary in their importance, and there consideration, for a domain that targeting an exact results Recall might be the way to go for measuring the accuracy in getting that results, for a domain that target results without having a set of exact expected results then Precision may be the way to go.

In the following we will discuss the definitions of these measures:

3.9.1 Precision

Precision measures the percentages of system-provided chunks that were correct. (Jurafsky & Martin, 2009) which is defined as:

$$Precision = \frac{Number\ of\ correct\ chenks\ given\ by\ system}{Total\ number\ of\ chunks\ given\ by\ system}$$

The definition of correct chunks depend on the way we want to measure the system, do we consider partially matched chunks as correct, or do we consider exact match are correct, that contributes to the level of accuracy we are looking for from the system.

3.9.2 Recall

Recall measures the percentage of chunks actually present in the input that were correctly identified by the system. (Jurafsky & Martin, 2009) which is defined as:

$$Recall = \frac{Number\ of\ correct\ chunks\ given\ by\ system}{Total\ number\ of\ actual\ chunks\ in\ the\ text}$$

Again the definition of correct chunks depend on the way we want to measure the system, do we consider partially matched chunks as correct, or do we consider exact match are correct, that contributes to the level of accuracy we are looking for from the system.

3.9.3 F-Measure

F-measure provides a way to combine the previous two measures into a single metric. (Jurafsky & Martin, 2009) which is defined as:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

The parameter β can be used based on the application needs, so if $\beta > 1$ this will favour the recall while if $\beta < 1$ the favour is for the precision.

But when $\beta = 1$, that means Recall and Precision are equally balanced (Jurafsky & Martin , 2009). And therefore this can be called F_1 which is defined as:

$$F_1 = \frac{2PR}{P + R}$$

For our system we use both Precision and Recall as a way of measuring its performance.

3.10 Human Parsing

Recent studies in the field called human sentence processing suggests that there are at least two ways in which humans apply probabilistic parsing algorithms, although there is still disagreement on the details (Jurafsky & Martin , 2009, P 483) .

One way suggest that the ability to predict words is based on the bigram of two words, such as verb-noun, a simple example to illustrate that below:

High Probability: One way to avoid confusion is to make changes during vacation (Jurafsky & Martin , 2009, P 483)

Low Probability: One way to avoid discovery is to make changes during vacation (Jurafsky & Martin , 2009, P 483)

The bigram "avoid confusion" have a higher probability than the bigram "avoid discovery"

More recent experiments have suggested that the probability of an upcoming word given the syntactic parse of preceding sentence prefix also predicts word reading time (Hale, 2001; Levy, 2008)(Jurafsky & Martin , 2009, P 483)

The second family of studies has examined how humans disambiguate sentences that have multiple possible parses, suggesting that humans prefer whichever parse is more probable (Jurafsky & Martin , 2009, P 483)

3.11 Summary

In this chapter we discuss some English word classes and their definition. We look at the sentence segmentation and tokenization, stemming of verbs and part of speech tagging for sentences components and some of the tagging tools available for this purpose.

We discuss the ambiguity as one of the challenges facing natural language processing software. We give an introduction for two different common used parsing methods (syntactical parsing, statistical parsing).

In syntactical parsing we discuss some of the methods used such as Top-Down Parsing, Bottom-Up Parsing and the Early Algorithm. We illustrate using examples the way some of this parsing methods work, and we discuss a commonly used parser which is SKY which is a parser that use the bottom up approach. We give an introduction for the statistical parsers and introduce the Collins Parser as one of the statistical parsers commonly used.

We discuss the three different evaluation measures (Precision, Recall and F-Measure), then we look at the Human Parsing and the way that some scientists suggested to explain human brain parsing techniques.

Chapter 4

Implementation and Data Processing

4.1 Introduction

Data can come from different sources, with different format, different structure, having apostrophes, double quotation marks, slashes, symbols etc. which may cause parsing to brake when using regular expressions and other tools to manipulate data via different stages of the life cycle of the tools that are used to extract the information from the given corpus.

Cleaning and processing raw data is an important initial part of any Natural Language Processing (NLP) task. We can get a corpus that is having unneeded character or symbols. Hence we need to remove from the provided corpus the unneeded chars or symbols and redundant tags like html tags or redundant text. The tools we use for data manipulation can crash or gave unexpected results if these tags or special characters are there. The main goal for cleaning the provided data is to eliminate what can cause the tools to crash or behave in a different way than we expect during the life cycle of the natural language processing application. The tools used may not always be written by us, which restrict us to work around the tools input formats.

Provided corpus does not necessary come in one sentence per line, we may get lines that have paragraphs that contain several sentences. We use the Oak system to tokenize the sentences in the provided corpus files, the Oak system take parameters to determine the input format and the output format for the tokenization process.

The corpus data have different format in which they are laid out, some will have a file with information about the data and the structure of the folders and files that are in the folders.

After the data is cleaned and tokenized, we pass the relevant sentences to the Named Entity (NE) tagger and the Parts of Speech (POS) tagger. We use the Oak system to generate the POS tagged sentence. The POS tagged sentences will provide us with the information about the verbs and their tenses.

A sentence may include a certain Named Entity types (among the 150 NEs possible types provided by Oak system) such as: PERSON, LOCATION, ORGANIZATION, GPE (Geo-Political Entity), FACILITY, DATE, MONEY, PERCENT, TIME, etc.

In the next sections we will discuss the task definition, the corpus, data processing and experiments with corpus using a preliminary approach, where next chapter will discuss the syntactical approach step by step.

4.2 Task Definition

Given a text, a paragraph or a sentence, we need to generate all questions that the given text will have answers for.

Ex. Tom came to my office, gave me a presentation and left happy.

We need to generate all possible questions, sample questions will be:

Who came to my office?

Who gave me a presentation?

Who left happy?

How did Tom leave?

Where did Tom come?

The given text can be in form of a group of simple sentences, complex sentences, paragraph or a bigger textual entity.

Based on the fact that the provided text can be different, we need to simplify the process and the task by making sure that the provided corpus is in the simplest format possible, such as breaking it into smaller entities that does not exceed a one sentence each.

We define a sentence as a "Textual entity that starts with a capital letter or a number and ends with a period".

According to that definition these sentences are valid sentences:

a. Tom swims in Park Lake.
b. 3 women rent an apartment.

While these sentences are not valid according to our definition:

a. tom swims in Park Lake

b. 3 women rent an apartment

c. my cat is sleeping.

d. Tom eats an apple

Sentence a. does not start with a capital letter nor end with a period.

Sentence b. does not end with a period.

Sentence c. does end with a period but does not start with a capital letter.

Sentence d. does start with a capital letter but does not end with a period.

So we can summarize the task definition as following:

- We get a corpus of different format and different structure
- The corpus data need to be cleaned from symbols and other characters that cause regular expression to break
- The corpus data need to be break down into a small textual entities that contain only one sentence each
- The small textual entity which is a sentence can be a simple sentence or a complex sentence
- Complex sentences need to be run via a process to generate simple sentences out of them
- Simple sentence is the input of the system that will generate questions based on the components of the simple sentence and interaction rules that govern the type of questions that can be generated for each of the combinations of elements in the simple sentence

4.3 Corpus

Data used as an input for the system are coming from different sources; we use development data provided by the Question Generation Shared Task Evaluation Challenge 2010 (QGSTEC); we also use TREC-2007 (Question Answering Track) dataset for our experiments and evaluation.

4.3.1 TREC-2007 (Question Answering Track)

It is short for Text REtrieval Conference, TEREC-2007 is governed by National Institute of Standards and Technology (NIST), NIST is an agency of the U.S. Commerce Department's Technology Administration (http://trec.nist.gov/tracks.html).

The conference has developed different data sets for different tracks some of the tracks and their goals are explained below:

- Million Query Track: This track goal is to test the hypothesis that a test collection built from very many very incompletely judged topics is a better tool than a collection built using traditional TREC pooling. The Million Query Track was last run in TREC 2009 (http://trec.nist.gov/tracks.html).

- Novelty Track: This track was designed to investigate systems abilities to locate new non–redundant information. The Novelty Track last run in TREC 2004 (http://trec.nist.gov/tracks.html).
- Question Answering Track: This track was designed to take a step closer to information retrieval rather than document retrieval. The Question Answering Track last run in 2007 (http://trec.nist.gov/tracks.html).
- Chemical IR Track: The goals is to develop and evaluate technology for large scale search in chemical documents including academic papers and patents to better meet the needs of professional searchers; specifically patent searchers and chemists. The Chemical IR Track was last run in TREC 2009 (http://trec.nist.gov/tracks.html).
- Cross-Language Track: which its goal is to investigate the ability of retrieval systems to find documents that pertain to a topic regardless of the language in which the document in written. While last run in TREC 2002, Cross-Language retrieval tasks are studied in different workshops (http://trec.nist.gov/tracks.html).

For our thesis we consider the Question Answering Track dataset, this dataset provide files that contain sentences and paragraphs, the files that contain the sentences and paragraphs are sorted under folders, the folders names represent the topics IDs for the provided dataset. We have 70 different folders that represent the topics in the TREC dataset.

Provided with the corpus data an xml file, this xml file contain 3 different types of questions, Factoid, List and Other. These questions are grouped under a topic id that is represented in the folders names of the provided dataset mentioned in the previous paragraph.

Below is a definition for the three provided question types:

- Factoid Questions: they are questions that there answer is a fact, or a one simple sentence like how old is Tom?
- List Questions: they are questions that there answer is a list like how many days in the week? Or how many newspapers are in USA?
- Other: is any question that may not belong to either of the previous types

A sample of a topic in the provided xml file is as follow:

```
<target id = "216" text = "Paul Krugman">
    <qa>
    <q id = "216.1" type="FACTOID">
        For which newspaper does Krugman write?
    </q>
    </qa>

    <qa>
    <q id = "216.2" type="FACTOID">
        At which university does Krugman teach?
    </q>
    </qa>
```

<qa>
<q id = "216.3" type="FACTOID">
From which university did he receive his doctorate?
</q>
</qa>

<qa>
<q id = "216.4" type="FACTOID">
What is Krugman's academic specialty?
</q>
</qa>

<qa>
<q id = "216.5" type="FACTOID">
What prize originating in Spain has Krugman won?
</q>
</qa>

<qa>
<q id = "216.6" type="LIST">
What are titles of books written by Krugman?
</q>
</qa>

```
    <qa>
    <q id = "216.7" type="LIST">
      What persons has Krugman criticized in his op-ed columns?
    </q>
    </qa>

    <qa>
    <q id = "216.8" type="OTHER">
      Other
    </q>
    </qa>
</target>
```

4.3.2 QGSTEC 2010

It is short for Question Generation Shared Task Evaluation Challenge 2010; this workshop was held at Carnegie Mellon University (Pittsburgh, Pennsylvania, USA). There were 3 tasks in this workshop (http://www.questiongeneration.org/QGSTEC2010):

- Task A: Question Generation from paragraph. In this task there was a group of paragraphs provided, and the participants need to provide six question in three scope levels, level one is broad that cover the entire paragraph, level two medium which cover multiple sentences and level three specific to a sentence or less (http://www.questiongeneration.org/QGSTEC2010).

- Task B: Question Generation from a sentence. In this task there was a group of sentences provided, and the participants need to provide 2 questions of each type for each sentence, based on a predefined question types to be produced (http://www.questiongeneration.org/QGSTEC2010).
- Task C: Open Task. In which Question Generation Evaluation effort does not lie in Task A or Task B (http://www.questiongeneration.org/QGSTEC2010).

Sample for Task A as presented in (http://www.questiongeneration.org/QGSTEC2010):

Abraham Lincoln (February 12, 1809 – April 15, 1865), the 16th President of the United States, successfully led his country through its greatest internal crisis, the American Civil War, preserving the Union and ending slavery. As an outspoken opponent of the expansion of slavery in the United States, Lincoln won the Republican Party nomination in 1860 and was elected president later that year. His tenure in office was occupied primarily with the defeat of the secessionist Confederate States of America in the American Civil War. He introduced measures that resulted in the abolition of slavery, issuing his Emancipation Proclamation in 1863 and promoting the passage of the Thirteenth Amendment to the Constitution. As the civil war was drawing to a close, Lincoln became the first American president to be assassinated.

The following set of questions will be scored based on scope as 1, 2, 2, 3, 3, 3 because they are ranked from the broadest to the most specific.

1) Who is Abraham Lincoln?

2) What major measures did President Lincoln introduce?

3) How did President Lincoln die?

4) When was Abraham Lincoln elected president?

5) When was President Lincoln assassinated?

6) What party did Abraham Lincoln belong to?

Sample for Task B as presented in (http://www.questiongeneration.org/QGSTEC2010):

INPUT SENTENCE:

Abraham Lincoln (February 12, 1809 – April 15, 1865), the 16th President of the United States, successfully led his country through its greatest internal crisis, the American Civil War (1861 – 1865).

TARGET QUESTION TYPE: WHEN?

OUTPUT:

(1) In what year was Abraham Lincoln born?

(2) In what year did the American Civil War commence?

The samples are taken from the workshop samples.

The workshop provides 2 datasets, one for development and one for test, both dataset was gathered from 3 different sources

- Wikipedia: The online encyclopaedia
- Yahoo! Answers: A social question answering service
- Open Learn: An educational resource

We participate in Task B; since we do not include the semantic features extraction which is needed to participate in task A, we used both the development data and the test data provided to evaluate our system and to participate in the workshop activity.

4.4 Data Processing

Corpus data are row data; they do not always match the input requirement of systems that use them. Therefore the row data need to go through different steps of preparation in order to be ready for use in any system. For our system we use the following steps to prepare the row data we get from different sources.

- Data Cleaning

  Corpus usually need to be cleaned from redundant data, from double quotes, single quotes, apostrophes and other special characters that can cause the different tools and scripts to broke. List of characters removed during this step is shown in appendix B.

- Tokenization

  Tokenization is defined as separating valid sentences that are in the same line into a separate line. After cleaning the corpus from the redundant data, corpus that is provided not always structured as valid sentences. We define a sentence as a textual structure that starts with a capital letter or a number and ends with a period. Therefore a paragraph that contains 10 valid sentences will become a 10 line of 10 valid sentences.

  To manipulate the data according to that definition, we used the OAK system, to tokenize the corpus that is provided.

The returned corpus after running the OAK system will be aligned with what we define as a sentence. An example illustrating the input and the output after running the input into OAK systems is described as follow:

Tom is eating an apple and playing chess. Kim is happy.

After Tokenization we get:

i. Tom is eating an apple and playing chess.
ii. Kim is happy.

If the line holds these sentences:

Tom is eating an apple and playing chess. kim is happy.

After Tokenization we have:

i Tom is eating an apple and playing chess.

The second part "kim is happy." will be omitted since it does not satisfy the definition for a sentence.

Below is an example from the TREC dataset:

The United States appears to have failed the first major test of new security arrangements since the September 11 attacks as the superpower struggles to cope with the destruction caused by Hurricane Katrina.

- POS Tagging

  From Tokenization step, we get every sentence in a separate line; in POS Tagging step, we get every word of every sentence tagged with the matching Part of Speech tag.

  Ex.

  - The/DT United/NNP States/NNPS appears/VBZ to/TO have/VB failed/VBN the/DT first/JJ major/JJ test/NN of/IN new/JJ security/NN arrangements/NNS since/IN the/DT September/NNP 11/CD attacks/NNS as/IN the/DT superpower/NN struggles/VBZ to/TO cope/VB with/IN the/DT destruction/NN caused/VBN by/IN Hurricane/NNP Katrina/NNP ./.

  We use the Penn Tree bank of Part of Speech tags; the list is included in Appendix C.

- Parsing

  We use Charniak syntactical parser to parse the Tokenized sentences, Charniak Parser provides us with a bracketed representation of a syntactical structure of the Tokenized sentence; considering this sentence:

  The cat that killed the rat that stole the milk that spilled on the floor that was slippery escaped.

  (http://www.cse.iitb.ac.in/~jagadish/parser/evaluation.html)

Below we show an example of the bracketed parsed representation:

```
(S1 (S (NP (NP (DT The) (NN cat))
 (SBAR (WHNP (WDT that))
  (S (VP (VBD killed)
     (NP (NP (DT the) (NN rat))
      (SBAR (WHNP (WDT that))
       (S (VP (VBD stole)
        (NP (NP (DT the) (NN milk))
         (SBAR (WHNP (WDT that))
          (S (VP (VBD spilled)
           (PP (IN on) (NP (DT the) (NN floor)))
           (SBAR (IN that)
            (S (VP (AUX was) (ADJP (JJ slippery))))))))))))))))
    (VP (VBD escaped))
    (. .)))
```

(http://www.cse.iitb.ac.in/~jagadish/parser/charniak/embed.txt)

- Data structure modeling

  From a bracketed representation, we construct a tree representation for the data.

- Word classification

  OAK system has 150 named entity types that can be tagged. They are included in a hierarchy.

  A sample of the input sentence after running Named Entity tagging process will be as follow:

  a. <ENAMEX TYPE=GPE>The United States</ENAMEX> appears to have failed the first major test of new security arrangements since the <TIMEX TYPE=DATE>September 11</TIMEX> attacks as the superpower struggles to cope with the destruction caused by Hurricane Katrina.

  This information is used to make candidate fine and coarse classes. This word classification will be used in the future steps to help determine the type of questions to be generated.

  Below we will define the Coarse Classes and Fine classes that are used in our system:

  a. Coarse Classes

     We define the five major coarse classifications as:

   - i. Human: This will have any subject that is the name of a person.
   - ii. Entity: This includes animals, plant, mountains and any object.
   - iii. Location: This will be the words that represent locations, such as country, city, school, etc.
   - iv. Time: This will be any time, date or period such as year, Monday, 9 am, last week, etc.
   - v. Count: This class will hold all the counted elements, such as 9 men, 7 workers, measurements like weight and size, etc.

b. Fine Classes

Fine classes are a hierarchy of classes that the root is a Coarse Class. Example of that will be the word “Toyota”, its classification will be as follow:

Car → Vehicle → Product → Entity

Car, Vehicle and Product are Fine classes, while Entity is the Coarse class.

A list of the fine classes is shown in Appendix D.

4.5 Experiments with corpus

We tried 2 different ways to experiment with system implementations:

4.5.1 A preliminary approach

In this approach we used the information we get from the Named Entity tagged sentences and the POS tagged sentences.

- Using NE information

  We use the NE information to generate the questions for each sentence. For example, from the entity PERSON we can generate the questions like "When was PERSON born?", "Where was PERSON born?", "Where did PERSON die?", "When did PERSON die?", "How did PERSON die", "What is PERSON provision?", "What degree PERSON hold?", "Where did PERSON get his degree from?", "When did PERSON graduated?" etc.

- Using POS information

  The system extracts all the verbs from the POS tagged sentence, and creates the proper questions that are relevant to these verbs with respect to the Noun. For example, the verb "live" will generate a question like "Where Noun lives?"

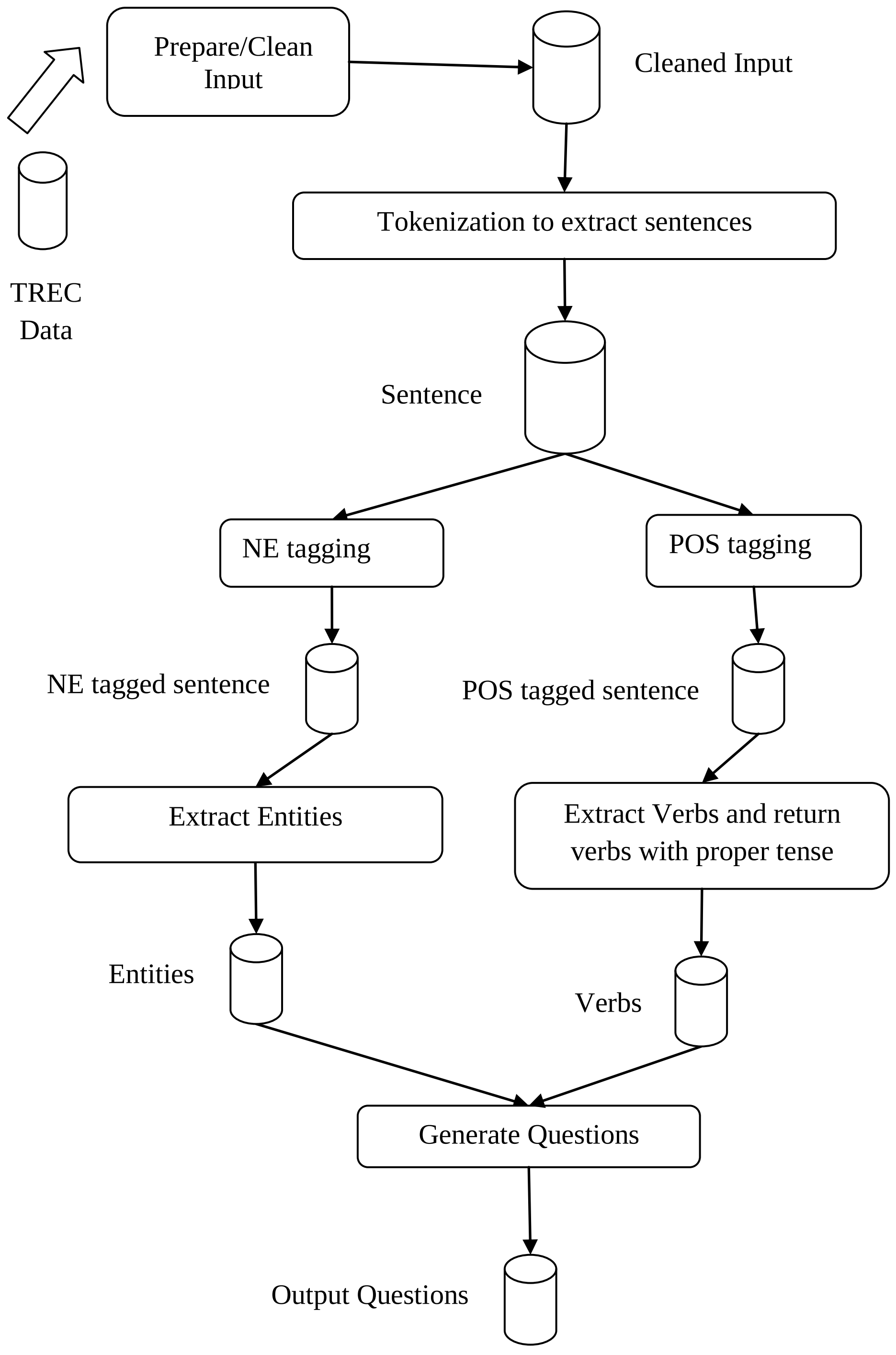


Figure 4 General overview of the Question Generation system component

4.5.2 A Syntactical approach

In this approach we use the syntactical structure of a given sentence to generate questions. We use the syntactic parsed sentence to extract elementary sentences from complex sentences. We reparse the elementary sentences and regenerate questions based on interaction rules between the Nouns (NPs), Verbs (VPs) and Prepositions (PPs).

We will discuss the steps followed in details in this approach in the next chapter.

4.6 Summary

We discussed in this chapter the main task definition for this thesis, which include several intermediate tasks.

We discuss the corpus that was used for development and for testing purposes. We describe the data processing steps, the processing steps include cleaning the provided corpus data from symbols and special characters that can break the tools used for parsing and tagging the different parts of sentence in the dataset.

We also discuss Tokenization of row data, Part of Speech Tagging, Parsing using Charniak parser, Data structure modeling as a tree representation from the syntactically bracketed representation of the sentences that is generated using Charniak parser.

We explain the classification method used, the method we use classify the sentence entities into Coarse and Fine classes.

In the next chapter we will discuss the Syntactical approach toward the question generation task, which show more promising evaluation results and having the door open for improvement in future work by adding more features to the system such as including semantic feature extraction.

Chapter 5

Syntactical approach

5.1 Introduction

Syntactical parsing provides information about the structure of the sentence, not just the type of the words it contains. From syntactical parsing we can represent the sentence and model it in a data structure. We use the Tree data structure to represent the sentence structure. We use Charniak parser to produce the syntactically parsed bracketed representation of the sentences.

The tree representation of the structure helps us to identify phrases, to generate elementary sentences from complex sentences, to avoid over generation of elementary sentences and questions.

We will discuss in this section how the system works? What are the components of the system? What are the interaction rules, which direct the type of questions to be generated?

5.2 System Components and Overview

The system is divided in different modules, and works in steps to generate the questions from an input data.

The input data can be a paragraph, a complex sentence or a simple sentence.

The system components are:

- Preparation and Clean Input System
- Elementary Sentence Extraction System (ESES)
- Sentence Classification System
- Question Generation System

Figure 8.1 shows a general overview of the system components.

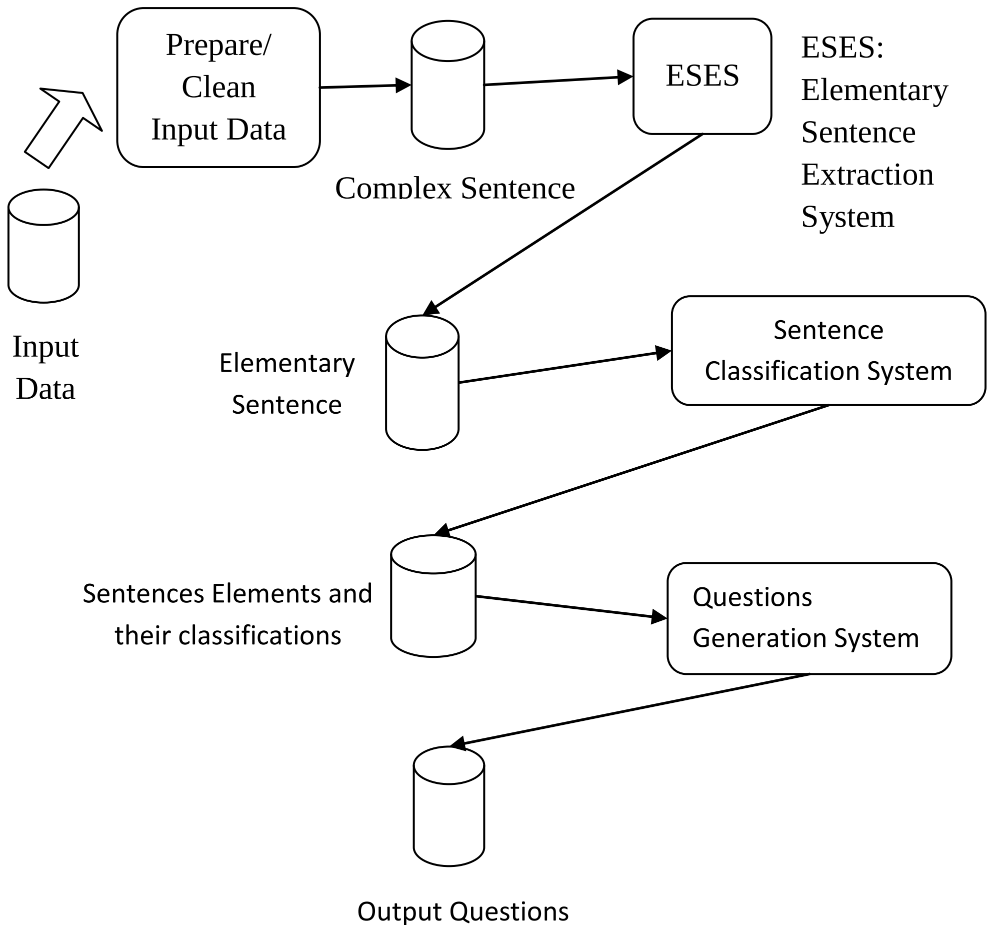

Prepare/ Clean Input Data
Complex Sentence
ESES
ESES: Elementary Sentence Extraction System
Input Data
Elementary Sentence
Sentence Classification System
Sentences Elements and their classifications
Questions Generation System
Output Questions

Figure 5 General overview of the Question Generation system component

Below is an algorithm that describes the way the system work:

```
Get Complex Sentence
    Process Sentence
    Get bracketed representation of parsed sentence
    Build bracketed representation of the sentence as a tree
        While building the tree
            Generate arrays for NPs and VPs
            Generate an array to preserve the syntactical order of NPs and VPs
    File of elementary sentences = call ESES (NPs,VPs)
    While (read elementary sentence from File of elementary sentences) {
        Classified elementary sentence =
        call Classify elementary sentence (elementary sentence)
      Call Generate Questions (classified elementary sentence)
    }
```

The ESES system is where the elementary sentences get extracted; the algorithm for the ESES is described below:

```
ESES (NPs,VPs, syntactical ordered NPsVPs ){
   For every NP
    For every VP
       If(VP in the scope of NP)
              Elementary sentence file =
                     Read from the tree from the NP to the VP and its preposition
 //file name where the elementary sentences stored
Return Elementary sentence file
}
```

The processes of the classification and the question generation are described below:

```
Classify elementary sentence (file  name)
    Subject_class = classify(get_sebject())
    Object_class = classify(get_object())
    Preposition_class = classify(get_preposition())
    Verb = get_verb()
    Return classification
Generate questions (classified elementary sentence)
    Question_types()=check_rules(sentence element classes)
    Call question_type(classified elementary sentence elements)

Question_type(classified elementary sentence elements)
    Generate the questions based on patterns
```

5.2.1 Preparation and Clean Input Module

In this step we have the entire corpus data cleaned, and tokenized. Since we run the system against large number of files, this step also regenerates the data and maintain track of the new data with relation to the original source for verification purposes.

Text parts get removed where they do not fit in the definition of a sentence as to be starting with a capital letter or a number and ends with a period. Special characters that cause the system to fail due to using regular expressions are also removed in this step.

5.2.2 Elementary Sentence Extraction System (ESES)

The provided sentences by corpus data may include complex grammatical structure with embedded clauses. Hence to generate more accurate questions, we developed an Elementary Sentence Extraction System (ESES). The ESES take a complex sentence as input and generate the possible elementary sentences from the complex sentences.

Considering the sentence below:

> The student, who is supervised by Tom, came to my office, gave me a presentation and left happy.

This sentence can be divided in a several simple sentences where every sentence will have a simple piece of information. Let us look at the possible simple sentences that can be extracted from the complex sentence above.

- The student is supervised by Tom.
- The student came to my office.
- The student gave me a presentation.
- The student left happy.

Figure 6 Illustrates how the Elementary Sentence Extraction System works.

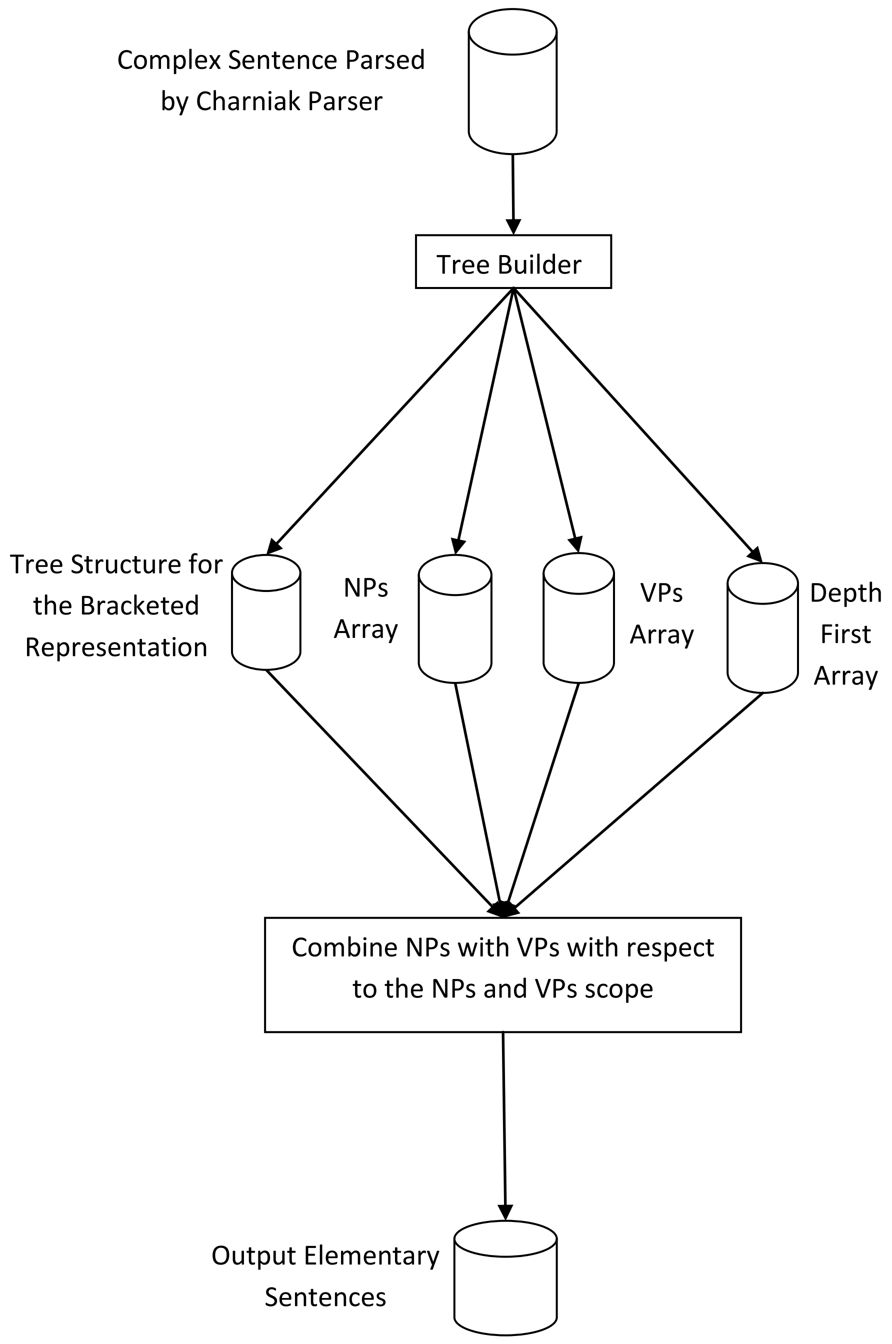

Complex Sentence Parsed by Charniak Parser
Tree Builder
Tree Structure for the Bracketed Representation
NPs Array
VPs Array
Depth First Array
Combine NPs with VPs with respect to the NPs and VPs scope
Output Elementary Sentences

Figure 6 Elementary Sentences Extraction System (ESES)

To extract the elementary sentences from complex sentences we syntactically parse each complex sentence. Syntactic parsing is analyzing a sentence using the grammar rules. We use Charniak parser. This module constructs a syntactic tree representation, from the bracketed representation of the parsed sentence. While building the tree process, we construct 3 arrays, one for the Noun Phrases (NPs), one for the Verb Phrases (VPs) and one for the Prepositions (PPs) with their location in the tree, a fourth array is generated, from the tree to represent the depth first sequence of the tree nodes and leaves structure.

We combine the NPs with the VPs and PPs by reading the NPs till the scope of the VPs and the PPs that are in the VPs scope and thus, we get the elementary sentences. The depth of the first sequence helps us to determine if the phrases to be joined are sequentially correct with the respect of the sentence structure. As an example, "Tom eats an apple and plants a tree". The depth of the first sequence check will prevent the system from generating the elementary sentence, "Tom plant an apple" since the verb plant came after the noun apple.

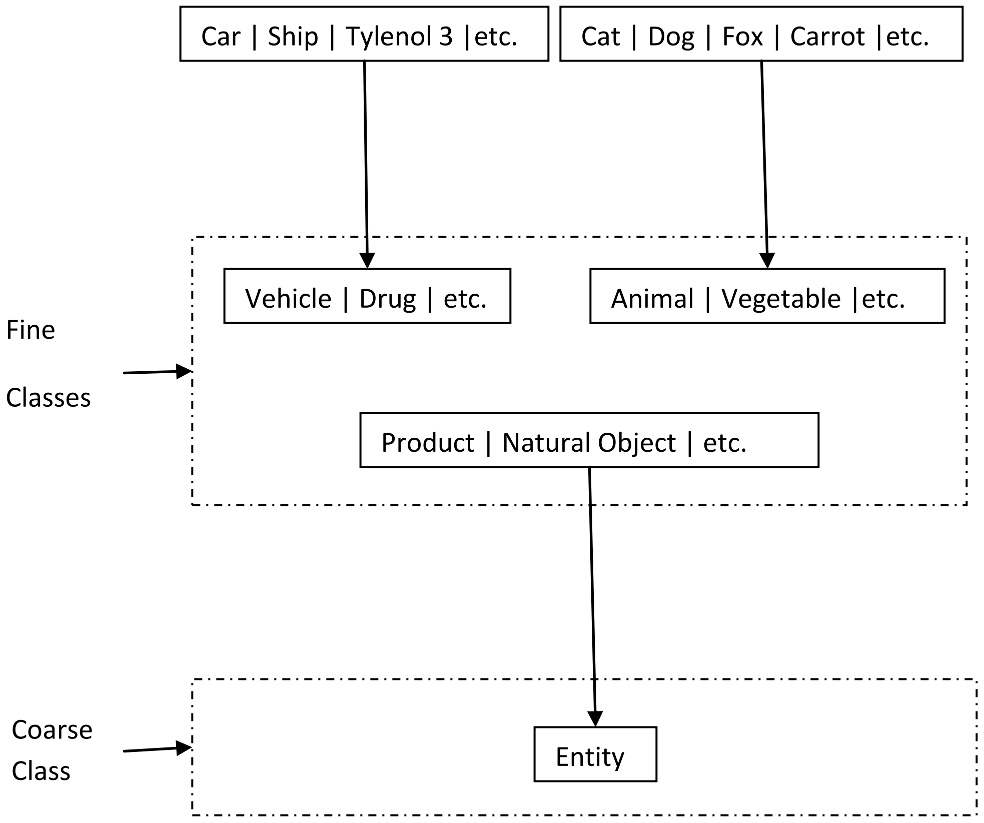
Organizations which include companies, institutes, government, market, etc are all a type of category Entity in our classification. Once the sentence words have been classified to coarse classes, we consider the relationship between the words in the sentence. As an example, if the sentence has the structure "Human Verb Human", it will be classified as "whom and who" question types. If it is followed by a preposition that represents date, then we add the "When" question type to its classification.

5.2.3 Question Generation System

Elementary sentences extracted by the ESES system are the inputs of this module. Based on the associated POS and NE tagged information, we get the subject, object, preposition and verb for each elementary sentence. We use this information to classify the sentences. We follow a two-layered taxonomy to represent a natural semantic classification for the sentences. Our sentence classifier module makes use of a sequence of two simple classifiers. The first classifies the sentences into fine classes (Fine Classifier) and the second into coarse classes (Coarse Classifier). This is a similar but opposite approach to the one described in (Li & Roth, 2002). The second classifier influences the first in that its candidate labels are generated by reducing the set of retained fine classes from the first into a set of coarse classes. This set is treated as the confusion set for the first classifier, the confusion set keep shrinking till we find the Coarse classes that the word belongs to. OAK System has 150 types that can be tagged. They are included in a hierarchy. This information is used to make candidate fine and coarse classes. We define the five coarse classes as (Ali H., Chali Y. and Hasan S. 2010):

1. Human: Any subject or object that is a name of a person.
2. Entity: Includes animals, plant, mountains and any object.
3. Location: Words that represent locations, such as country, city, school, etc.
4. Time: Words that represent time, date or period such as year, Month, 2011, Monday, 9 am, last week, etc.
5. Count: Hold all the counted elements, such as 9 men, 10 workers, measurements like weight etc.

A sample word classification is described in Fig 7:



Car | Ship | Tylenol 3 |etc.
Cat | Dog | Fox | Carrot |etc.
Fine Classes
Vehicle | Drug | etc.
Animal | Vegetable |etc.
Product | Natural Object | etc.
Coarse Class
Entity

Figure 7 Word to Fine Class to Coarse Class Classification process description

The words Car, Ship etc. will be in a fine class, so we will find the fine class first, then this fine class will be classified to a more general fine class Vehicle, since Vehicle is not a Coarse class therefore the system will try to find what class the Vehicle class is belonging to, and the loop keep going till finding the Coarse class Entity in this case. Every time we get a more general classification that will reduce the number of fine classes that the word can be belonging to.

Organizations which include companies, institutes, government, market, etc are all a type of category Entity in our classification (Ali H., Chali Y. and Hasan S. 2010).

We process each sentence in a top-down manner to get it classified.

Let, the confusion set of any sentence be $C_0 = \{c_1, c_2, \ldots\ldots\ldots; c_n\}$, the set of all the coarse classes. Initially, the fine classifier determines the fine classes. Then, the set of fine classes is reduced to a coarse class determined by the class hierarchy (Ali H., Chali Y. and Hasan S. 2010).

That is, the set $\{ f_{i1}, f_{i2}, \ldots\ldots, f_{im} \}$ of fine classes is mapped into the coarse class c_i Based on the coarse classification, we consider the relationship between the words in the sentence (Ali H., Chali Y. and Hasan S. 2010).

The sentence below for example:

"Husam cook the rice at 4 pm"

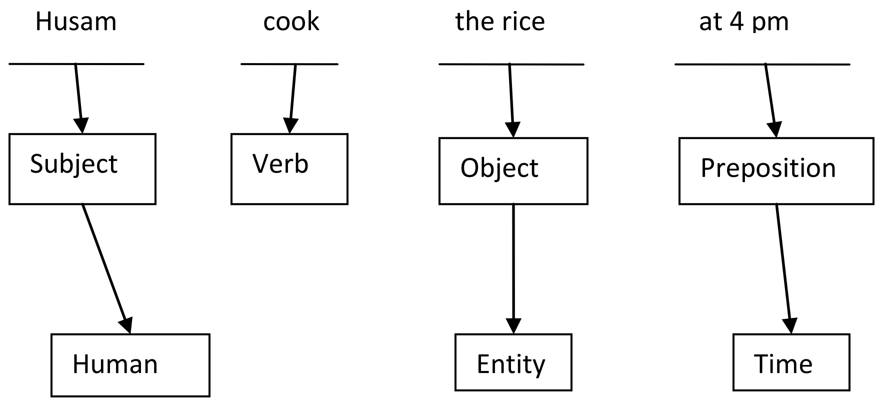


Figure 8 Word to Fine Class to Coarse Class Classification example

We check the coarse classes of the input elementary sentence according to the word-to-word interaction rules predefined by us (Ali H., Chali Y. and Hasan S. 2010). The rule check will produce the type of questions that can be generated while considering the verb tense and stem (Ali H., Chali Y. and Hasan S. 2010). In fact, the system's output will be the questions of the types suggested here. In addition; the system generate questions of other types other than "WH" questions types such as questions start with "Is, Did, … etc.".

We define a set of interaction rules like "Human Verb Human", "Human Verb Entity", "Human Verb Human Time" ... etc (Ali H., Chali Y. and Hasan S. 2010).

We check the coarse classes according to the word to word interaction.

For example:

"Tom ate an orange at 7 pm"

Tom is a subject of coarse class Human (Ali H., Chali Y. and Hasan S. 2010).

An orange is an object of type Entity (Ali H., Chali Y. and Hasan S. 2010).

At 7 pm is a preposition of type Time (Ali H., Chali Y. and Hasan S. 2010).

Sample generated questions based on the rule "Human Entity Time" will be: (Ali H., Chali Y. and Hasan S. 2010)

- Who ate an orange?
- Who ate an orange at 7 pm?
- Who ate at 7 pm?

- An orange ate by whom?
- An orange ate by whom at 7pm?
- What did Tom eat?
- What did Tom eat at 7 pm?
- What did Tom do at 7 pm?
- When did Tom eat an orange?
- When did Tom ate?
- Did Tom eat an orange?
- Did Tom eat an orange at 7 pm?
- … Etc.

If the input elementary sentence has the coarse class's structure:

"Human Verb Location"

The classification for the possible question types will be as "Who, Where" question types.

Let us consider the sentence below:

"Husam loves Canada"

The questions can be generated will be:

- Who loves Canada?
- Husam loves where?
- Does Husam love Canada?
- … Etc.

If the sentence is followed by a preposition that represents time, then we add the "When" question type to its classification (Ali H., Chali Y. and Hasan S. 2010).

"Husam left Calgary at 9 am"

The questions can be generated will be:

- Who left Calgary?
- Who left Calgary at 9 am?
- Where did Husam left at 9 am?
- Where did Husam left?
- When did Husam left Calgary?
- When did Husam left?
- Did Husam leave Calgary at 9 am?
- … Etc.

This module has two simple classifiers. The first classifies the sentence into fine classes (Fine Classifier) and the second classifies the sentences into coarse classes (Coarse Classifier) (Ali H., Chali Y. and Hasan S. 2010).

The diagram in Figure 9 shows a diagram of the classification part of the question generation system.

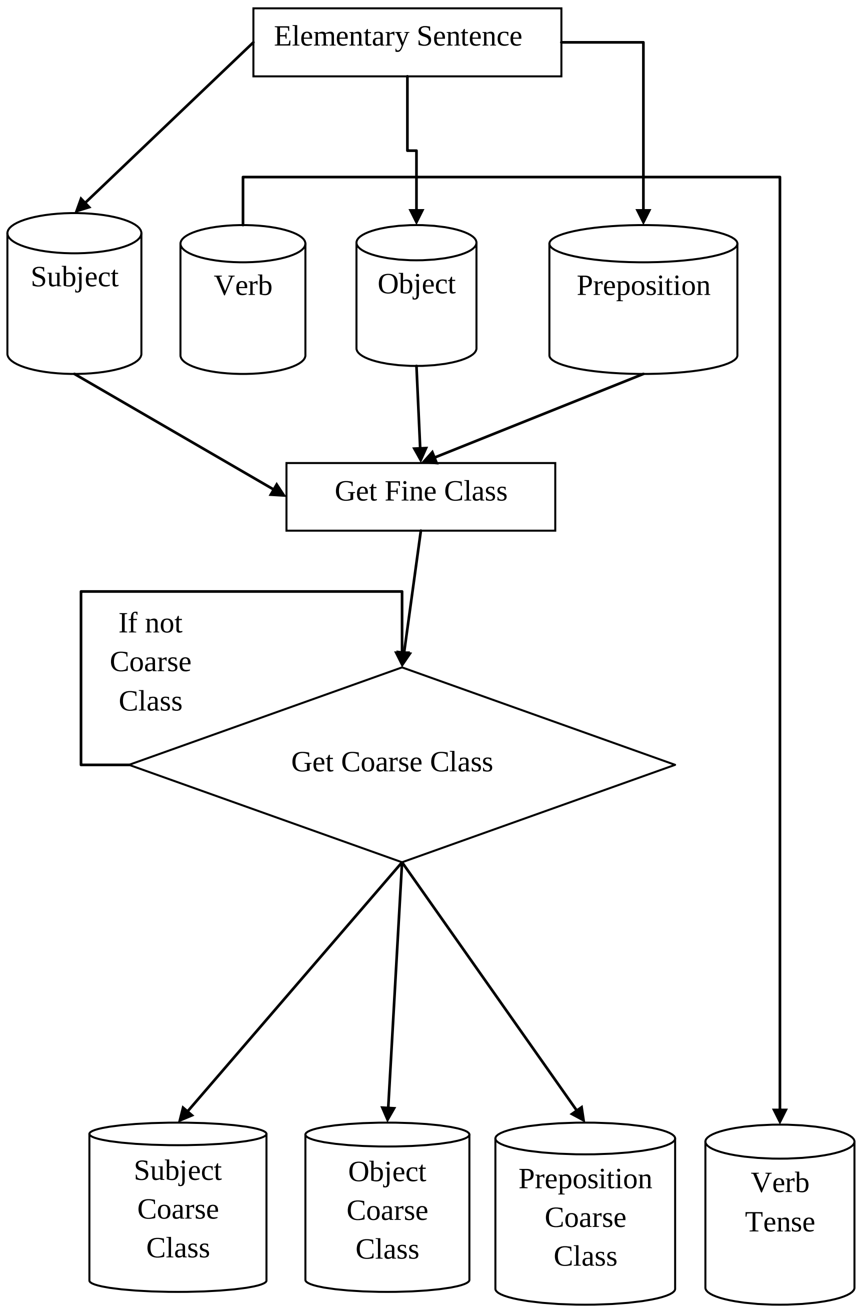


Figure 9 Question Generation System Classification Process

Below is a sample of the interaction rules set that we predefine, like “Human Verb Human”, “Human Verb Entity”, “Human Verb Human Time” ... etc.:

Core classes: **H** = Human **E**= Entity **L**= Location **T**=Time **C**=Count

The sentence: “Tom teach Sam”

Relations			**Question type**	**Example Questions**
Subject	**Object**	**Preposition**		
H	H	-	Who Whom What	Who teach Sam? Whom Tom is teaching? What are Tom and Sam doing?

Table 5 Sample 1 Questions based on relationship between sentence parts

The sentence: “Tom introduce Sam to Kim”

Relations			**Question type**	**Example Questions**
Subject	**Object**	**Preposition**		
H	H	H	Who Whom What	Who introduce Sam to Kim? Tom introduces Whom to Kim? What did Tom do?

Table 6 Sample 2 Questions based on relationship between sentence parts

The sentence: “Tom pushes Sam toward the car”

Relations			**Question type**	**Example Questions**
Subject	**Object**	**Preposition**		
H	H	E	Who	Who pushes Sam towered the car?
			Whom	Tom pushes whom toward the car?
			What	Tom pushes Sam toward what?

Table 7 Sample 3 Questions based on relationship between sentence parts

The sentence: “Tom bring Sam to Canada”

Relations			**Question type**	**Example Questions**
Subject	**Object**	**Preposition**		
H	H	L	Who	Who brings Sam to Canada?
			Whom	Tom brings whom to Canada?
			What	What did Tom do?
			Where	Where Tom did bring Sam?

Table 8 Sample 4 Questions based on relationship between sentence parts

The sentence: “Tom met Sam at 9am”

Relations			Question type	Example Questions
Subject	**Object**	**Preposition**		
H	H	T	Who	Who met Sam at 9am?
			Whom	Tom met whom at 9am?
			What	What did Tom do at 9am?
			When	Tom met Sam when?
				When Tom did meet Sam?

Table 9 Sample 5 Questions based on relationship between sentence parts

The full interactions rules are shown in Appendix A.

The system assign different values for coarse classes for ranking purposes, and take a Boolean input to determine if we want a “WH” question types only or all types that can be generated.

- Who ate an orange?
- Who ate an orange at 7 pm?

Which one of these two questions should be ranked higher?

By giving a value for each component of the interaction rules as below:

Human	10
Entity	8
Location	6

Time	4
Count	2

Therefore the question: “Who ate an orange?” has two parts of the sentence involved the “Human Class” and the “Entity Class” which gives it a value of 10+8 = 18.

Therefore the question: “Who ate an orange at 7 pm?” has three parts of the sentence involved the “Human Class”, the “Entity Class” and the “Time Class” which gives it a value of 10+8 +4= 22.

The question: Who ate an orange at 7 pm? Will be ranked higher than the question: Who ate an orange?

Next chapter we will discuss the results of running the system against 2 different corpuses and evaluate the questions generated.

Chapter 6

Experiments and Evaluation

6.1 Introduction

We experiment with our system; using 2 different corpus data. We used development and test data provided by the Question Generation Shared Task Evaluation Challenge 2010. We also used the TREC-2007 (Question Answering Track) dataset.

In the next sections we will discuss the process step by step.

6.2 Experiments

We run the system to experiment with its performance against 2 corpus data as follow:

6.2.1 TREC- 2007 (Question Answering Track)

The purpose of the track was described as "The goal of the TREC question answering (QA) track is to foster research on systems that directly return answers, rather than documents containing answers, in response to a natural language question." (Dang, Kelly and Lin, 2007).

The way the dataset prepared, was based on the assumption, that an adult native English speaker, who is seeking information about a particular target. This target can be a person, organization, event or thing.

The main task was to have systems that provide answers for a set of questions; these questions are organized based on a target, the target can be a person, organization or an event.

The answers were to be found, by searching documents, these documents are in files, these files are stored in folders, and the folders names are representing topics that are given a number such as 216 or 217.

The way our system uses the provided datasets, and the set of questions provided by TREC track, was the opposite of the way the dataset was intended to be used.

We use the questions that are provided as a way to measure the quality of the questions that our system generates. Sample of a person target based topics, and the questions provided by the dataset are in the table 10:

Topic: 216, Target: Paul Krugman		
Question		
Type	ID	Text
FACTOID	216.1	For which newspaper does Krugman write?
	216.2	At which university does Krugman teach?
	216.3	From which university did he receive his doctorate?
	216.4	What is Krugman's academic specialty?
	216.5	What prize originating in Spain has Krugman won?
LIST	216.6	What are titles of books written by Krugman?
	216.7	What persons has Krugman criticized in his op-ed columns?
OTHER	216.8	Other

Table 10 Sample of topic 216, the target for this topic is Paul Krugman

Sample of an organization target based topics, and the questions provided by the dataset are in table 11:

Topic: 224, Target: WWE		
Question		
Type	ID	Text
FACTOID	224.1	Who is the chairman of WWE?
	224.2	Who is the chief executive of WWE?
	224.3	Where is WWE headquartered?
	224.4	What is "WWE" short for?
	224.5	WWE evolved from what earlier organization?
	224.6	What cable network airs WWE?
LIST	224.7	What wrestlers have appeared in WWE events?
OTHER	224.8	Other

Table 11 Sample of topic 224, the target for this topic WWE

Sample of an event target based topics, and the questions provided by the dataset are in table 12:

Topic: 225, Target: Sago Mine disaster		
Question		
Type	ID	Text
FACTOID	225.1	On what date did the disaster occur?
	225.2	Who was the sole survivor?
	225.3	What company owned the Sago Mine?
	225.4	How many miners died in the disaster?
	225.5	In what state was the Sago mine?
	225.6	What organization investigated the disaster?
LIST	225.7	Who were victims of the disaster?
OTHER	225.8	Other

Table 12 Sample of topic 225, the target for this topic Sago Mine disaster

It is more practical to reduce the sentences that are going to be processed by the system; due to having 50 files in each topic, and having 70 different topics and varied number of sentences per file; we reduce the number of sentences to be processed by discarding the sentences that does not have the target or the answers. Thus we use the sentences that have the target or the answer for the questions provided, this helps reduce the load on the system, speed the process and reduce the noise that the system will have to produce.

We process these sentences that have the target or the answer; then we inject the cleaned processed sentences into the system.

The system will create the elementary sentences then generate the questions that are related to the elementary sentence syntactical and grammatical structure.

We process each sentence in a top-down manner to get it classified. Let, the confusion set of any sentence is $C_\delta = \{C_1, C_2,\cdots, C_n\}$, the set of all the coarse classes.

Initially, the fine classifier determines the fine classes. Then, the set of fine classes is reduced to a coarse class determined by the class hierarchy.

That is, the set $\{f_{i1}, f_{i2},\cdots, f_{im}\}$ of fine classes is mapped into the coarse class c_i. Based on the coarse classification, we consider the relationship between the words in the sentence. For example, if a sentence has the structure: “Human Verb Human”, it will be classified as “whom and who” question types. If it is followed by a preposition that represents time, then we add the “When” question type to its classification.

We check the coarse classes according to the word-to-word interaction rules. The rule check will produce the type of questions that can be generated while considering the verb tense and stem. Indeed, the output of the system will be the questions of the type that is suggested based on an interaction rules and questions patterns. In this research, we define a set of interaction rules like “Human Verb Human”, “Human Verb Entity”, “Human Verb Human Time” ... etc. below is a sample of these rules:

Relations			**Question type**	**Example Questions**
Subject	**Object**	**Preposition**		
H	H	-	 Who Whom What	Tom teach Sam Who teach Sam? Whom Tom is teaching? What Tom and Sam doing?
H	H	H	 Who Whom What	Tom introduce Sam to Kim Who introduce Sam to Kim? Tom introduces Whom to Kim? What did Tom do?
H	H	E	 Who Whom What	Tom push Sam toward the car Who push Sam towered the car? Tom pushes whom toward the car? Tom pushes Sam toward what?
H	H	L	 Who Whom What Where	Tom bring Sam to Canada Who bring Sam to Canada? Tom brings whom to Canada? What did Tom do? Tom brings Sam to where?
H	H	T	 Who Whom What When	Tom wake Sam at 9am Who wake Sam at 9am? Tom wakes whom at 9am? What did Tom do at 9am? Tom wakes Sam when?

Table 13 Sample of Interaction rules

The full interactions rules are shown in Appendix A.

Since the questions provided by the dataset are human generated questions, the ability to produce the exact questions is going to be limited by the human interference with the original syntactic structure of the sentences that hold the answers. Also human generated questions can use synonyms that our system does not use as a feature to help generate the same syntactically and grammatically structured questions with different words.

Table 14 show the recall of the experiment where Qg is the number of questions generated by our QG system that are in the actual questions provided by TREC dataset, Qa is the number of actual questions given for each topic in the TREC dataset excluding questions with grammatical error.

We experiment with 70 topics from the TREC 2007 dataset. For 20 topics, our system could generate the questions given the target. However, for the other topics our system was unable to generate any questions. We get a recall of 0.000 for the other topics. From Table 1 we see that for the “When”, “Where” and “Who” type questions, the recall is similar. For the type “What”, we get the lowest recall of 0.135. We also show the overall Recall considering all the question types.

Type	Qg	Qa	Recall
What	7	52	0.135
Which	3	10	0.300
Who/Whom	11	20	0.550
How Many/Much	3	13	0.231
Where	4	8	0.500
When	1	2	0.500
Over all	29	105	0.276

Table 14 Question generated that are in the Actual set of questions provided by TREC dataset and the Recall measure for the results

Figure 10 shows a diagram that depicts the different factoid questions and our system ability to generate the questions provided by the TREC dataset.

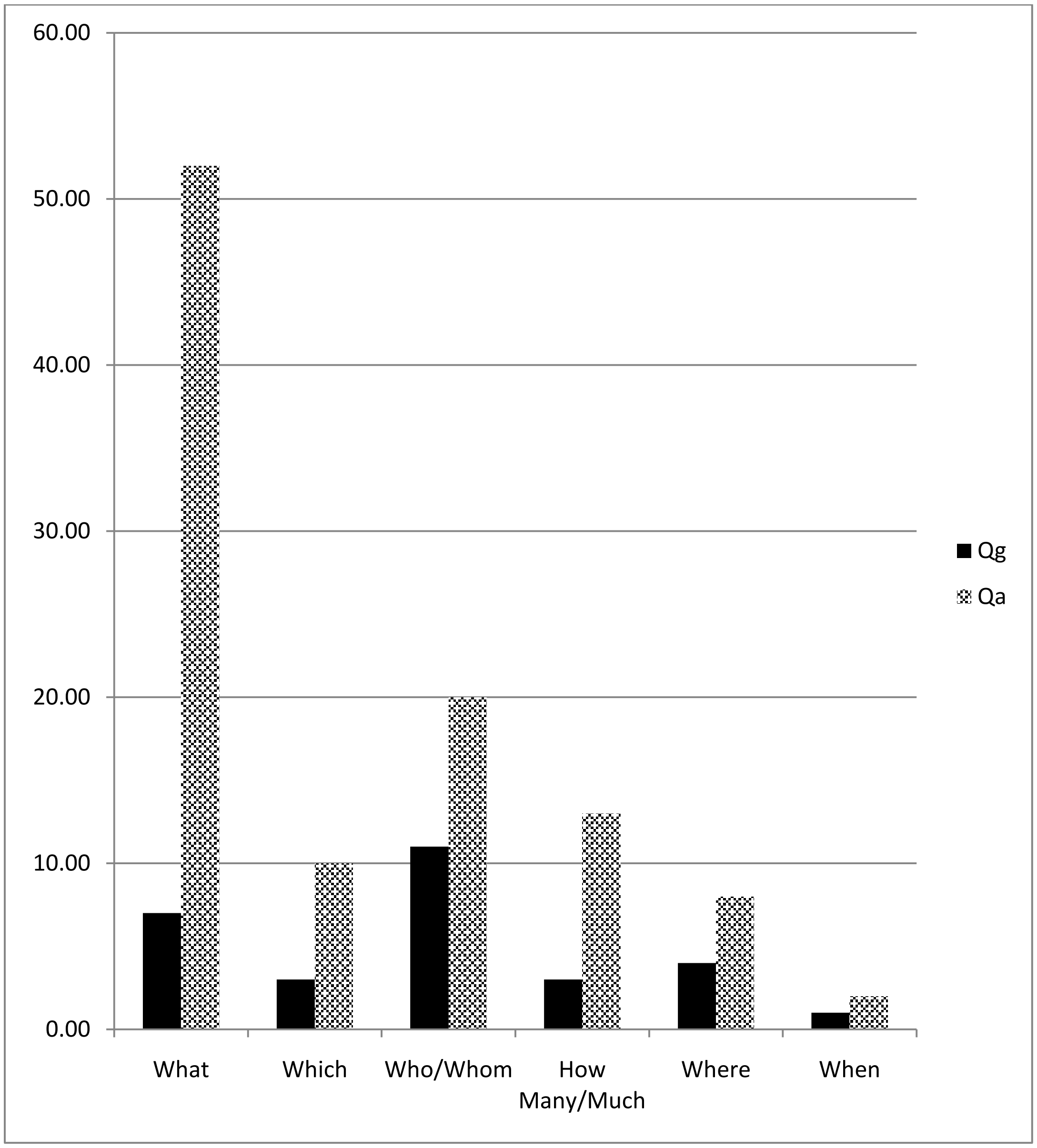


Figure 10 Questions generated that are in the Actual set of questions provided by TREC dataset

For Precision we experiment with 5 topics from the 20 topics that we could generate question from, from TREC 2007 dataset.

In this analysis we rejected the grammatically incorrect generated question from the relevant set. Table 2 shows that the precision was high for the types “Who and Where“, the type “When“ was still above 0.5, the other types was hovering around 0.4.

The reason for the lower precision is that we do not considering the grammatically wrong constructed questions to be a valid relevant questions, we believe that if the system included a semantic analysis for the sentence and its components the results will be higher, and the grammatically wrong sentences to decrease in number.

Table 15 show the results of the experiment with the Precision measure, Where, Qg is the number of questions generated by our QG system, Qr is the number of related questions generated by the system excluding questions with grammatical error.

Type	Qg	Qg ∩ Qr	Precision
What	105	43	0.410
Which	57	23	0.404
Who/Whom	144	106	0.736
How Many/Much	43	17	0.395
Where	117	89	0.761
When	71	37	0.521
Over all	537	315	0.587

Table 15 Question generated that are related to the target provided by TREC dataset and the Precision measure for the results

We find that the precision for the factoid questions “Who/Whom and Where” was higher than the other type of factoid questions, that is because this are based on human or location which are easier to generate and predict comparing to the other types of factoid questions.

Figure 11 shows a diagram that depicts the different factoid questions and our system ability to generate questions related to the topics or the answers of the questions provided by the TREC dataset.

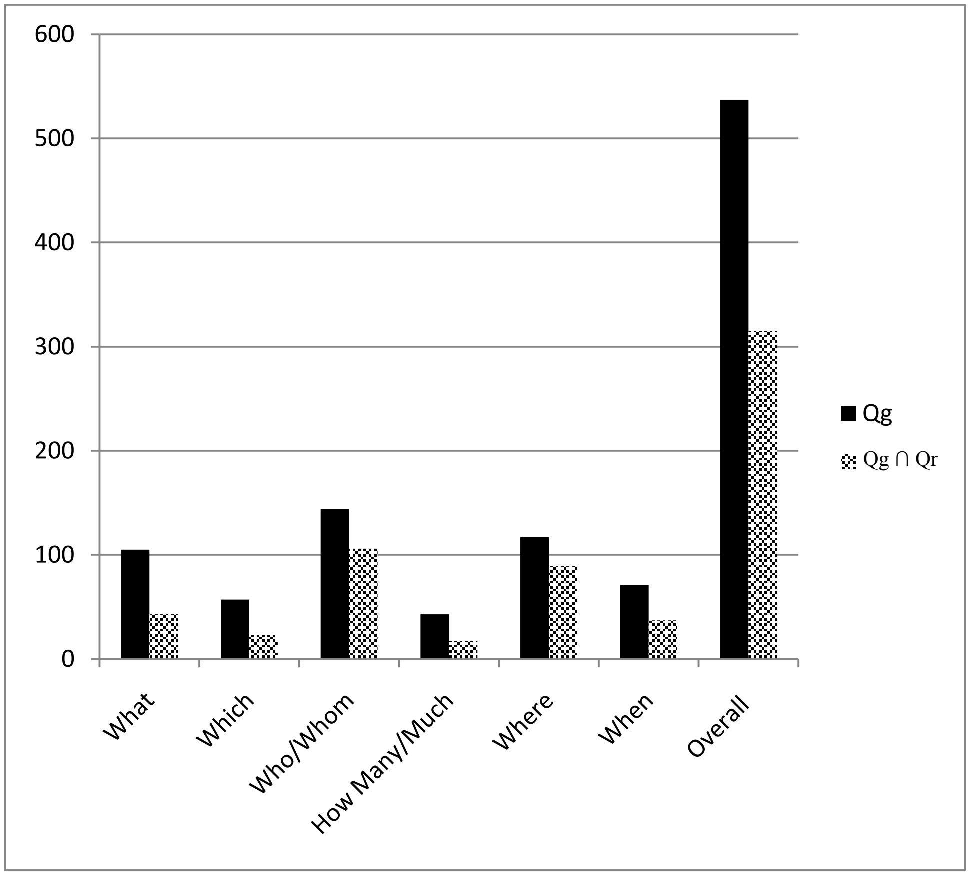


Figure 11 Generated factoid questions that are related to the topics or the answers in the Actual set of questions provided by TREC dataset

Overall we find the precision was 0.587, "Who/Whom and Where" question types were having the highest precision and "How Many/Much" question types was having the lowest precision.

6.2.2 QGSTEC (Question Answering Track)

For the evaluation of the system, we ran it against the development data provided by (QGSTEC).

The development data were provided in the format of a single sentence and a specific target question type (E.g. WHO? WHY? HOW? WHEN? etc). Sentences were selected from the primary data sources for the QGSTEC (Wikipedia, Yahoo! Answers and OpenLearn).

In warrants mentioning that extremely long or short sentences were avoided. We used the questions provided by (QGSTEC) for each sentence from different types of possible questions for the sentence.

We then employed the widely used evaluation measure: recall. We define recall as follows:

$$Recall = \frac{Qg \cap Qa}{Qa}$$

Where, Qg is the number of question generated by our QG system and Qa is the number of actual questions provided by (QGSTEC). With consideration for the fact that humans sometimes generate questions that are worded differently than the sentence structure.

Also human can use synonyms which are a feature that can be added to the system to help improve its overall quality.

Results can be seen in Table 16 which represents the recall for the different types of questions that the system was able to generate. We also included the overall recall for the system results:

Type	Qg	Qa	Recall
What	14	36	0.389
Which	3	30	0.100
Who	15	25	0.600
Where	7	23	0.304
When	11	29	0.379
How	3	21	0.143
Why	2	8	0.250
Yes/No	2	9	0.222
Over all	57	181	0.315

Table 16 Question generated that are in the Actual set of questions provided by QGSTEC dataset and the Recall measure for the results

We found that type "Who" had the highest recall 0.600, while types "What, Who, Where and When" were in a closed range between above 0.300. Question type "How" was 0.143; the lowest recall was for question type "Which" 0.100.

The reason for the variation of quality for different type of questions is that the ability to generate questions about Human, Location and Time are more likely to have a better rate of getting an accurate syntactical and grammatical structure, while other types of questions face the challenge of the formation of the question that can fit for the type while preserving the grammatical accuracy.

Over all recall was low since we just consider the questions provided by the QGSTEC dataset.

Figure 12 shows a diagram that depicts the different questions and our system ability to generate the questions provided by the QGSTEC dataset.

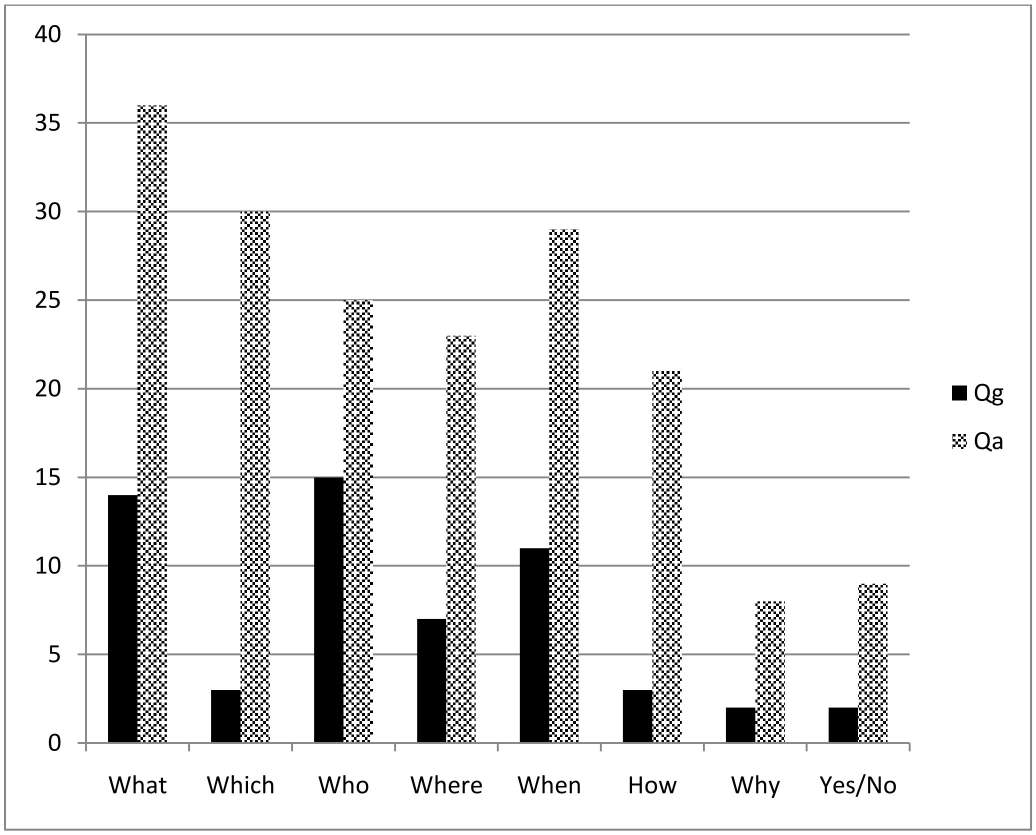


Figure 12 Questions generated that are in the Actual set of questions provided by QGSTEC dataset

However the other types were not as good, since they had a recall below 0.300%.

The overall recall for the results was 0.315%, which can be improved if we included the semantically parsed sentences, in the process of generating the elementary sentences, and the generation of the questions to adopt the possibilities of wording the questions differently while preserving the meaning.

The noise that was generated was high considering the provided sample questions. The noise was due to both grammatical incorrectness and questions that were generated, which are not included in the dataset provided, but the grammatically correct generated questions were high.

We also used the other widely used measure of evaluation which is precision. We calculated the precision for the factoid questions types, and we found that it is better to use the other widely used evaluation measure: precision. We define Precision as follow:

$$Precision = \frac{Qg \cap Qr}{Qr}$$

We conducted experiment with 20 sentences for the precision evaluation due to the human effort needed for this kind of evaluation method.

Type	Qr	Qg ∩ Qr	Precision
What	49	19	0.388
Which	31	7	0.226
Who	70	45	0.643
Where	55	31	0.564
When	53	27	0.509
How Many/Much	43	17	0.395
Over all Factoid	301	146	0.485

Table 17 Question generated that are related to the target provided by QGSTEC dataset and the Precision measure for the results

Figure 13 shows a diagram that depicts the different factoid questions and our system ability to generate questions related to the sentences provided by the QGSTEC dataset.

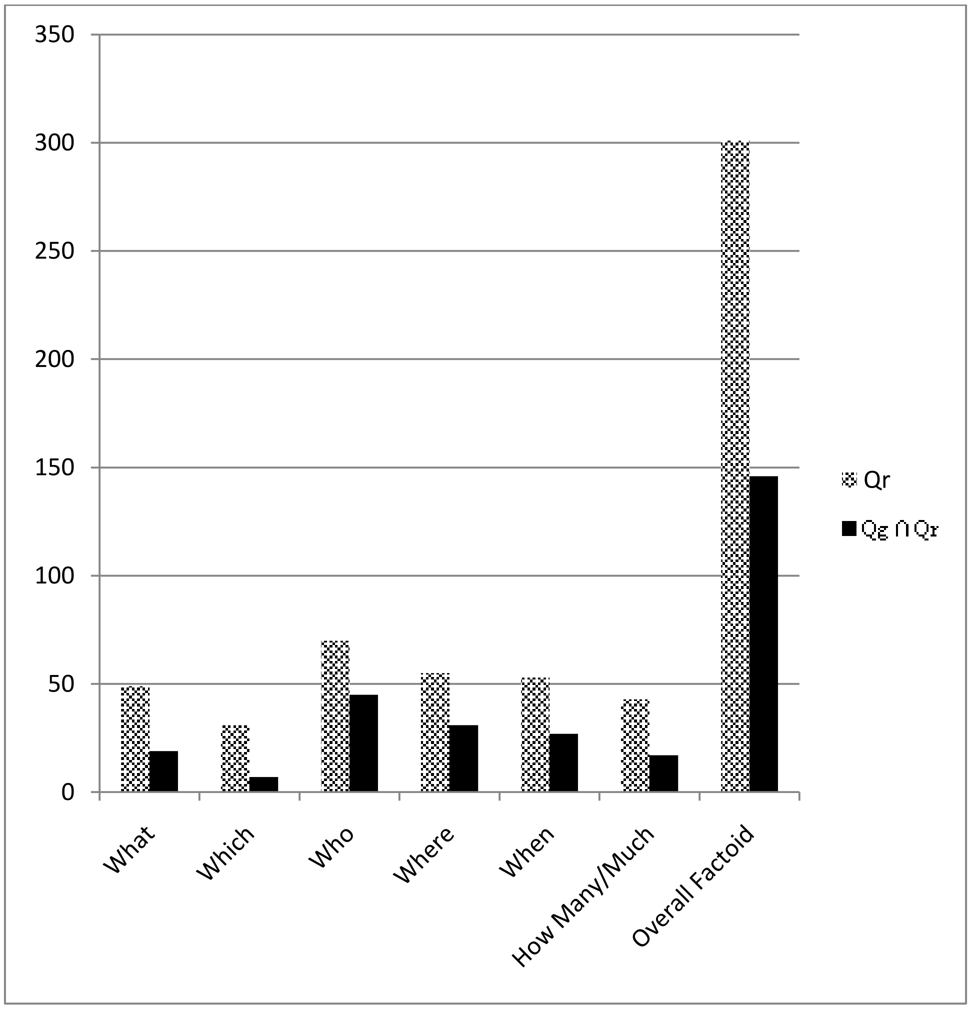


Figure 13 Questions generated that are in the Actual set of questions provided by QGSTEC dataset

In the next chapter we will discuss the findings and the conclusion of this thesis.

Chapter 7

Conclusion

7.1 Our Approach

In this thesis, we proposed an approach to automatically generate questions given sentences.

We used the dataset provided by the TREC 2007 Question Answering Track. We filtered out important sentences from the dataset by following a target-driven method.

We used the development data given by (QGSTEC) and test data given by (QGSTEC). We evaluated the performance of our system using Recall and Precision.

We simplified the process by extracting elementary sentences from the complex sentences using syntactic information and Part Of Speech Tagging (POS).

After classifying the elementary sentences based on their subject, verb, object and preposition, we generated the questions automatically from them using a predefined set of interaction rules. We plan to extend the number of interaction rules in the future. We will also focus on the sentence classification module to make it more robust. Since human generated questions always tend to have words with different meanings and senses, the system can be improved with the inclusion of semantic information and word sense disambiguation. We hope to carry on these ideas and develop additional mechanisms to question generation based on the dependency features of the answers and answer finding (Li & Roth, 2006; Pinchak & Lin, 2006).

7.2 Findings

The recall of the results from running the system against the TREC 2007 Question Answering Track data set we find from Table 1 that for the "When", "Where" and "Who" type questions, the recall is similar. For the type "What", we get the lowest recall of 0.135. We also show the overall Recall considering all the question types.

Type	Qg	Qa	Recall
What	7	52	0.135
Which	3	10	0.300
Who/Whom	11	20	0.550
How Many/Much	3	13	0.231
Where	4	8	0.500
When	1	2	0.500
Over all	29	105	0.276

Table 18 Question generated that are in the Actual set of questions provided by TREC dataset and the Recall measure for the results

Table 19 show the results of the experiment with the Precision evaluation measure, Where, Qg is the number of questions generated by our QG system, Qr is the number of related questions generated by the system excluding questions with grammatical error.

Type	Qg	Qg ∩ Qr	Precision
What	105	43	0.410
Which	57	23	0.404
Who/Whom	144	106	0.736
How Many/Much	43	17	0.395
Where	117	89	0.761
When	71	37	0.521
Over all	537	315	0.587

Table 19 Question generated that are related to the target provided by TREC dataset and the Precision measure for the results

The reason for the differences in the precision and recall for the different types of questions is that questions that are of type Who/Whom are having less possible choices for human to generate them in different wording, or using synonyms. Same apply for When/Where question types. But when considering the What/Which and How Much/Many questions types the chances for using synonyms are higher, and that gave the human generated questions the variety that a machine generated questions cannot match without using the semantic features.

Results of running the system against development data provided by (QGSTEC) can be seen in Table 20 which represents the recall for the different types of questions that the system was able to generate. We also included the overall recall for the system results:

Type	Qg	Qa	Recall
What	14	36	0.389
Which	3	30	0.100
Who	15	25	0.600
Where	7	23	0.304
When	11	29	0.379
How	3	21	0.143
Why	2	8	0.250
Yes/No	2	9	0.222
Over all	57	181	0.315

Table 20 Question generated that are in the Actual set of questions provided by QGSTEC dataset and the Recall measure for the results

The precision can be seen in table 21.

Type	Qr	Qg ∩ Qr	Precision
What	49	19	0.388
Which	31	7	0.226
Who	70	45	0.643
Where	55	31	0.564
When	53	27	0.509
How Many/Much	43	17	0.395
Over all Factoid	301	146	0.485

Table 21 Question generated that are related to the target provided by QGSTEC dataset and the Precision measure for the results

7.3 Future research direction

To increase the quality of the generated questions, and increase the overall recall and precision for the system, we have to make it able to generate more questions, and in order to achieve this, we need to increase the number of patterns that are used, we need to increase the number of the interaction rules, we also need to include the semantic features.

The addition of semantic features will increase the volume of questions that are generated by the system, which will increase drastically the precision of the overall system performance.

Semantic features will help produce the same questions with different wording such as:

What did Sam study at his class?
What did Sam study at his lesson?
What did Sam study at his course?
What did Sam study at his subject?

All these are valid questions that can be generated by using the synonyms for the word " study".

Also a second sample will be:

Did Sam teach Tom?
Did Sam educate Tom?
Did Sam show Tom?
Did Sam train Tom?
Did Sam tutor Tom?
Did Sam lecture Tom?
Did Sam guide Tom?

All this are valid questions that can be generated by using the synonyms for the word "Teach".

In table 22, we can see the synonyms of the word teach in the verb form as shown in http://thesaurus.com/browse/teach.

Main Entry:	**teach**
Part of Speech:	*verb*
Definition:	educate; instill knowledge
Synonyms:	advise, brainwash*, break in, brief, catechize, coach, communicate, cram, demonstrate, develop, direct, discipline, drill, edify, enlighten, exercise, explain, expound, fit, form, give instruction, give lessons, give the facts, ground, guide, illustrate, imbue, impart, implant, improve mind, inculcate, indoctrinate, inform, initiate, instruct, interpret, lecture, nurture, open eyes, polish up, pound into, prepare, profess, rear, school, sharpen, show, show the ropes, train, tutor
Notes:	**learn** means to acquire or gain skill, knowledge or comprehension; **teach** means to impart skill, knowledge or comprehension to
Antonyms:	learn

* = informal/non-formal usage

Table 22 Synonyms for the verb "Teach"

Therefore the use of semantic feature will help generate more questions which will increase the precision for the system in general.

References

- Ali H., Chali Y. and Hasan S. (2010) Automatic Question Generation from Sentences: a Preliminary Approach In Proceedings of the Conference on Traitement Automatique de la Langue Naturelle, Montreal, Canada
- Ali H., Chali Y. and Hasan S. (2010) Automation of Question Generation From Sentence, In Proceedings of the Third Workshop on Question Generation, Pittsburgh, Pennsylvania, United States of America.
- Andrenucci A. & Sniders E (2005). Automated Question Answering: Review of the Main Approaches. In Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05), Sydney, Australia.
- Boyer, K. E. and Piwek, P. eds. (2010). Proceedings of QG2010: The Third Workshop on Question Generation. Pittsburgh: questiongeneration.org.
- Brown J. C., Frishkoff G. A. & Eskenazi M. (2005). Automatic Question Generation for Vocabulary Assessment. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada.
- Brown J. C., Frishkoff G. A. & Eskenazi M. (2005). Automatic Question Generation for Vocabulary Assessment. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada.

- Chali Y. (2008). Some Aspects of a Target-Driven Question Generation, Workshop on the Question Generation Shared Task and Evaluation Challenge, NSF, Arlington, VA, September 25-26, 2008.
- Charniak E. 1993. Statistical Language Learning, MIT Press.
- Dang H. T., Kelly D. & Lin J. (2007). Overview of the TREC 2007 Question Answering Track. In Proceedings of the 16th Text REtreival Conference, Gaithersburg, Maryland.
- Even-Zohar and D. Roth. 2001. A sequential model for multi class classification. In EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing.
- Graesser A. C., Vanlehn K., Rose C. P., Jordan P. W. & Harter D. (2001). Intelligent Tutoring Systems with Conversational Dialogue. AI Magazine, 22(4), 39–52.
- Hasan S. Answering Complex Questions: Supervised Approaches. Thesis 2009.
- Heilman M., 2011. Automatic Factual Question Generation from Text. Ph.D. Dissertation, Carnegie Mellon University. CMU-LTI-11-004.
- Leung H., Frederick Li, Rynson Lau. Advances in Web Based Learning - ICWL 2007: 6th International Conference.
- Kristy E., Chali Y., Corbett A. & etl. The question generation shared task and evaluation challenge Workshop Report, Sponsored by the National Science Foundation Edited By: Vasile Rus AND Arthur C. Graesser The University Sponsored by the National Science Foundation Edited By: Vasile RUS and Arthur C. Graesser The University ISBN: 978-0-615-27428-7

- Lauer T. W., Peacock E. & Graesser A. C. (1992). Questions and Information Systems.
- Li X. & Roth D. (2002). Learning Question Classifiers. In Proceedings of the 19th International Conference on Computational Linguistics, p. 1–7, Morristown, NJ, USA : Association for Computational Linguistics.
- Li X. & Roth D. (2006). Learning Question Classifiers: The Role of Semantic Information. Journal of Natural Language Engineering, 12(3), 229-249
- McCawley, J. D. (1978). Where you can shove infixes. In Bell, A. and Hooper, J. B. (Eds.), Syllables and Segments, pp. 213-221. North-Holland
- McGough J., Mortensen J., Johnson J. & Fadali S. 2001. A Web-based Testing System with Dynamic Question Generation. In ASEE/IEEE Frontiers in Education Conference.
- Pinchak C. & Lin D. 2006. A Probabilistic Answer Type Model. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, p. 393–400.
- Roth D. (1998). Learning to resolve natural language ambiguities: A unified approach. In Proc. of the American Association of Artificial Intelligence, pages 806–813.
- Roth D. (1999). Learning in Natural Language: Theory and Algorithmic Approaches.
- Rus, V. and Graesser, A.C. (2009). Workshop Report: The Question Generation Task and Evaluation Challenge, Institute for Intelligent Systems, Memphis, TN, ISBN: 978-0-615-27428-7.

- Rus V. & Grasser A. C. (2009). The Question Generation Shared Task and Evaluation Challenge. In Workshop on the Question Generation Shared Task and Evaluation Challenge, Final Repot, the University of Memphis: National Science Foundation.
- Sagae K. (2009). Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In Proceeding IWPT '09 Proceedings of the 11th International Conference on Parsing Technologies.
- Tamura, A., Takamura, H., Okumura, M. Classification of Multiple-Sentence Questions. IJCNLP 2005, LNAI 3651, p. 426-437, 2005.
- Voorhees H. (2000). Overview of the TREC-9 question answering track. In The Ninth Text Retrieval Conference (TREC-9), pages 71–80. NIST SP 500-249.
- Wang W., Tianyong H. & Wenyin L. (2008). Automatic Question Generation for Learning Evaluation in Medicine. In LNCS Volume 4823.
- Wei Chen G. A. & Mostow J. 2009. Generating Questions Automatically from Informational Text. In Proceedings of the 2nd Workshop on Question Generation (AIED 2009), p. 17–24.
- Wolfe J. H. Automatic question generation from text - an aid to independent study. In Proceeding SIGCSE '76 Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education ACM New York, NY, USA ©1976
- Yao X., Question Generation with Minimal Recursion Semantics. Master Thesis. Saarland University & University of Groningen. 2010

- Zhang, W., Lee, S. (2003) Question classification using support vector machines, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, Toronto, Canada.

Appendix A

Relations			Question type	
Subject	**Object**	**Preposition**	WH	Other
H	H	-	Who Whom What	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
H	H	H		
H	H	E		
H	E	H		
E	H	H		
H	H	L	Who Whom What Where	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	E	H		
L	L	H		
L	H	E		
L	H	-		
L	H	H		
L	H	L		
E	L	H		
E	H	L		
H	H	T	Who Whom What When	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	H	T		
E	T	H		

C	E	T	What When How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	C	T		
E	C	T		
E	T	C		
C	L	H	Who Whom What Where How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	E	C		
E	C	L		
E	L	C		
L	C	C		
C	T	L	When Where How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	T	C		
C	L	L	Where How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	C	-		
L	L	C		

C	T	H	Who When How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	H	L	Who Whom What Where How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	H	C		
L	C	H		
C	H	T	Who Whom What When How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	H	C		
H	E	-	Who What	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	E	H		
E	H	-		

H	E	T	Who What When	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
H	T	-		
H	T	T		
H	C	T		
H	E	C	Who What How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
H	C	C		
E	C	H		
C	H	-		
C	C	H		
H	L	-	Who Where What	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
H	L	L		
H	C	L		
H	E	L		

H	L	T	Who	Is/was
H	T	L	What	Do/does/did
L	T	H	Where	Have/has/had
			When	Can/could
L	H	T		Will/would
				Shall/Should
H	L	C	Who	Is/was
			What	Do/does/did
L	C	L	Where	Have/has/had
			How many/much	Can/could
				Will/would
L	C	E		Shall/Should
C	L	T	Who	Is/was
			What	Do/does/did
			Where	Have/has/had
L	C	T	When	Can/could
			How many/much	Will/would
				Shall/Should
			Who	Is/was
			What	Do/does/did
H	T	C	When	Have/has/had
			How many/much	Can/could
				Will/would
				Shall/Should

H	C	-	Who Whom How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
H	C	H	Who Whom What How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	E	-	what	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	E	L	What Where	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	L	-		
E	L	E		
E	L	L		

E	L	T	What Where When	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	T	L		
L	E	T		
E	T	-	What When	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	T	E		
E	T	T		
E	E	T		
E	C	-	What How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
E	C	E		
E	C	C		
L	E	E	What Where	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	E	L		
L	L	E		
L	E	-		

L	L	-	Where	Is/was Do/does/did
L	L	L		
L	T	E	Where When	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
L	T	L		
L	T	T		
L	L	T		
L	T	-		
C	H	H	Who Whom What How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	H	E		
C	H	C		
C	E	H		
C	E	-	What How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	E	E		
C	C	-		
C	C	E		
C	C	C		
E	E	C		

C	L	-	What Where How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	L	E		
C	L	C		
C	C	L		
C	E	L		
C	T	-	When How many/much	Is/was Do/does/did Have/has/had Can/could Will/would Shall/Should
C	T	T		
C	T	C		
C	T	E		

Appendix B

```
Top
Name
    PERSON                        # Bill Clinton
        LASTNAME                  # Clinton
        MALE_FIRSTNAME                # Bill
        FEMALE_FIRSTNAME              # Mary
    ORGANIZATION                      # United Nation, NATO
        COMPANY                   # IBM, Microsoft
        COMPANY_GROUP             # Start Alliance, Tokyo-Mitsubishi Group
        MILITARY                  # The U.S Navy
        INSTITUTE                 # the National Football Leage, ACL
        MARKET                    # New Yourk Stock Exchange, TSX, NASDAQ
        POLITICAL_ORGANIZATION    #
            GOVERNMENT            # Department of Education, Ministry of Finance
            POLITICAL_PARTY           # Green Party, NDP, Liberal Party
            PUBLIC_INSTITUTION        # Canada Post Office
        GROUP                 # The Beatles
            SPORTS_TEAM           # NHL, CHL
        ETHNIC_GROUP              # Han race, Hispanic
        NATIONALITY               # Canadian
    LOCATION                      # Time Square
        GPE                   # Asia, Middle East, Palestine
            CITY              # Toronto, Calgary, London
            COUNTY                # Westchester
            PROVINCE              # Alberta, BC
            COUNTRY               # Canada
        REGION                    # Scandinavia, North America, Asia, East coast
        GEOLOGICAL_REGION             # Altamira
            LANDFORM              # Rocky Mountains
            WATER_FORM            # Hudson River, Slave Lake
            SEA               # Pacific Ocean, Gulf of Mexico
        ASTRAL_BODY                   # Halley's comet, the Moon
            STAR                  # Sun, Centaurus
            PLANET                # the Earth, Mars, Venus
        ADDRESS                   #
            POSTAL_ADDRESS            # 19 Colombia Blvrd. Lethbridge, AB, Canada
            PHONE_NUMBER          # 403-222-1234
            EMAIL             # husam.ali@uleth.ca
            URL               # http://www.uleth.ca/artsci/math-computer-science
```

```
FACILITY                    # Empire State Building
  GEO                     # Calgary Hospital
    SCHOOL                  # University of Lethbridge
    MUSEUM                # MOMA
    AMUSEMENT_PARK          # Walt Disney World, Oakland Zoo
    WORSHIP_PLACE           # Canterbury Cathedral
    STATION_TOP             #
      AIRPORT               # Calgary Airport, London Heathrow Airport
      STATION               # London Victoria Station
      PORT                  # Port of Vancouver
    CAR_STOP                # Sydney Bus Depot
  LINE                    # Westchester Bicycle Road
    RAILROAD                # New Jersey Transit
    ROAD                    # 23RD Street
    WATERWAY                # Suez Canal
    TUNNEL                  # Euro Tunnel
    BRIDGE                  # Golden Gate Bridge
  PARK                      # Central Park, Hyde Park
  MONUMENT                  # Statue of Liberty

PRODUCT                     # Windows 2007, Rosetta Stone
  VEHICLE                   # Vespa ET2, Honda Elite 50s
    CAR                   # Toyota, Audi 90
    TRAIN                   # Bullet Train
    AIRCRAFT                # B-747
    SPACESHIP               # Apollo 11
    SHIP                  # Titanic
  DRUG                      # Tylenol
  WEAPON                    # Patriot Missile
  STOCK                   # NABISCO stock
  CURRENCY                  # Euro, yen
  AWARD                     # Nobel Peace Prize
  THEORY                    # Newton’s law, GB theory
  RULE                      # The U.S. Constitution
  SERVICE                   # Air Canada Flight AC227
  CHARACTER                   # Mickey Mouse
  METHOD_SYSTEM               # Federal Tax
  ACTION_MOVEMENT             # The U.N. Peace-keeping Operation
  PLAN                      # Star Wars Plan
  ACADEMIC                  # Sociology, Physics
  CATEGORY                  # 48kg class
  SPORTS                    # Men’s 100 meter, tennis
  OFFENCE                   # first-degree murder
  ART                     # Venus of Melos
    PICTURE                 # Night Watch, Guernica
```

```
        BROADCAST_PROGRAM  # Friends, Larry King Live
        MOVIE               # Jurassic Park, Star Wars
        SHOW                # Les Miserable, Madam Butterfly
        MUSIC               # My Life, Your Song
    PRINTING                # 2001 Consumer Survey
        BOOK                    # 1001 Ways to Reward Employees
        NEWSPAPER               # The New York Times, Wall Street Journal
        MAGAZINE                # Newsweek, Time
DISEASE                         # Cancer
EVENT                       # Hanover Expo
    GAMES                   # Wald Cup
    CONFERENCE                  # APEC, Naples Summit
    PHENOMENA                   # El Nino
    WAR                     # World War II
    NATURAL_DISASTER                # Kobe Earthquake
    CRIME                   # Murder of Black Dahlia

TITLE                           # Mr., Ms., Miss.
    POSITION_TITLE              # CEO, King
LANGUAGE                        # English
RELIGION                    # Christianity, Islam, Buddhism

NATURAL_OBJECT                      # mitochondria, shiitake mushroom
    ANIMAL                          # elephant
    VEGETABLE                       # rice, spinach
    MINERAL                         # Hydrogen, carbon monoxide

COLOR                           # black, white, red, blue
TIME_TOP
    TIMEX
        TIME                        # 10 p.m., afternoon
        DATE                        # August 10, 2011
        ERA                     # Glacial period
    PERIODX                         # 2 semesters
        TIME_PERIOD                 # 10 minutes, 15 hours
        DATE_PERIOD                 # 10 days, 50 days
        WEEK_PERIOD                 # 10 weeks
        MONTH_PERIOD                # 10 months
        YEAR_PERIOD                 # 10 years
NUMEX                               # 10 bits
    MONEY                           # $10, 100 yen
    STOCK_INDEX                     # 26 5/8
    POINT                           # 10 points
    PERCENT                         # 10%, 10 1/2%
    MULTIBLICATION                      # 10 times
    FREQUENCY                       # 10 times a day
```

RANK	# 1ST prize
AGE	# 36, 77 YEARS OLD
MEASURMENT	# 10 bytes, 10 millibar
PHYSICAL_EXTENT	# 10 meters, 10 inches
SPACE	# 10 acres
VOLUME	# 10 cubic feet
WEIGHT	# 10 tons
SPEED	# 10 miles per hour
INTENSITY	# 10 lumina, 10 decibel
TEMPERATURE	# 60 degrees
CALORIE	# 10 calories
SEIMIC_INTENSITY	# 6.8 (on Richter scale)
COUNTX	
N_PERSON	# 10 biologists, 10 workers
N_ORGANIZATION	# 10 industry groups, 10 credit unions
N_LOCATION	# 10 cities, 10 areas, 10 regions, 10 states
N_COUNTRY	# 10 countries
N_FACILITY	# 10 buildings, 10 schools, 10 airports
N_PRODUCT	# 10 systems, 20 paintings, 20 super computers
N_EVENT	# 5 accidents, 5 interviews, 5 bankruptcies
N_ANIMAL	# 10 animals, 10 horses
N_VEGETABLE	# 10 flowers, 10 carrots
N_MINERAL	# 10 diamonds