T.C.
Mersin Üniversitesi
Sosyal Bilimler Enstitüsü
İngiliz Dili ve Edebiyatı Ana Bilim Dalı

CORPUS LINGUISTICS THEORY AND DESIGN AND APPLICATION OF A TURKISH CORPUS

Taner SEZER

YÜKSEK LİSANS TEZİ

Mersin, 2010

T.C.
Mersin Üniversitesi
Sosyal Bilimler Enstitüsü
İngiliz Dili ve Edebiyatı Ana Bilim Dalı

CORPUS LINGUISTICS THEORY AND DESIGN AND APPLICATION OF A TURKISH CORPUS

Taner SEZER

Danışman
Prof. Dr. Mustafa Ş. AKSAN

YÜKSEK LİSANS TEZİ

Mersin, 2010

T.C.
MERSİN ÜNİVERSİTESİ
SOSYAL BİLİMLER **ENSTİTÜSÜ**
TEZ ONAY SAYFASI FORMU

K-Q TSE-ISO-EN 9000

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE,

Taner SEZER tarafından hazırlanan Derlem Dilbilim Kuramı ve Türkçe Dil Derlemi Tasarım ve Uygulaması başlıklı bu çalışma, jürimiz tarafından İngiliz Dili ve Edebiyatı ☒ Ana Bilim / ☐ Ana Sanat Dalında ☒ YÜKSEK LİSANS / ☐ SANATTA YETERLİK / ☐ DOKTORA TEZİ olarak kabul edilmiştir.

		Başarılı	**Başarısız**		
Yüksek Lisans / Doktora		☒	☐	Başkan	İmza Prof. Dr. Mustafa Ş. AKSAN
		☒	☐	Üye	İmza Doç. Dr. Şaziye YAMAN
		☒	☐	Üye	İmza Yrd. Doç. Dr. Aygül UÇAR
		☐	☐	Üye	İmza Unvan, Ad Soyadı, İmza
		☐	☐	Üye	İmza Unvan, Ad Soyadı, İmza

Başkan dahil jüri üye sayısı 3

ONAY

Yukarıdaki imzaların, adı geçen öğretim elemanlarına ait olduklarını onaylarım.

29 / 09 / 2010

Prof. Dr. Mustafa Ş. AKSAN
Sosyal Bilimler Enstitüsü Müdürü

ÖZET

Derlemdibilim, dilbilimin son yıllarda öne çıkan bir dalıdır. Bu çalışmada derlemdilbilim kuramı ve derlemdilbilimin temel kavramlarına kısaca değinilmiştir.

Bu çalışmanın temel amacı, internet üzerinden erişilebilen, yetkili kullanıcılarca veritabanına yeni metinler eklenebilen ve çıkartılabilen, bağlam içinde anahtar kelime gösterebilen, platform bağımsız ve Türkçe karakterleri sorunsuz gösterebilen bir derlem oluşturmaktır.

Geliştirilen örnek derlemin kodları çalışma içinde verilerek, belirtilen bu temel özelliklere sahip bir derlemi oluşturmak isteyenlere yardımcı olunmaya çalışılmak istenmiştir.

Anahtar Sözcükler: derlem, derlem dilbilim, bilgisayar tabanlı dilbilim, kodlama

ABSTRACT

Corpus linguistics is a rapidly developing field in linguistics. In this paper, we aimed to explain the basic concepts of corpus linguistics.

The study aims to build a corpus which can be reached via internet which is independed of OS, that authenticated users can add or delete data to the corpus database, that can display hit sets in keyword in context (KWIC) view and can serve Turkish characters, which are not exist in English alphabet, without any problems.

The code of the sample corpus developed for this study is given in this paper so that researchers can use it freely.

Keywords: corpus, corpus linguistics, computational linguistics, coding

CONTENTS

LIST OF ABBREVATIONS

ANC	American National Corpus
AFL	Academic Free License
BNC	British National Corpus
CQP	Corpus Query Processor
CWB	Corpus Workbench
METU	Middle East Technical University
TNC	Turkish National Corpus
UTF-8	Unicode Transformation Format
W3C	World Wide Web Consortium
IETF	Internet Engineering Task Force
IMC	Internet Mail Consortium
TEI	Text Encoding Initiative
GPL	General Public License
EULA	End User License Agreement
OCR	Optical Character Recognition
XML	Extensible Mark-up Language
SGML	Standardized Generalized Mark-up Language
SQL	Structured Query Language
PHP	Personal Home Page

LIST OF FIGURES

INTRODUCTION

This study aims at peeking the corpus linguistics theory and the design and the development processes of a Turkish Corpus.

Corpus is a Latin word that means "*body*". In linguistics, corpus (plural corpora) refers to collected and combined data from written or spoken language, which is served to obtain linguistic data or used to verify hypothesis about a language. There are many other definitions for corpus.

The rise of the corpus linguistics has a parallel line with the development of computer technology. As computer technology made it possible to store and process larger amount of data, corpora also get larger. The features of corpora are now more detailed and can supply more information to linguists. The numbers of corpora are increasing rapidly in the last decades.

Though seems only related with linguistics, corpus linguistics also serves for computational linguistics, cryptologist, educational linguists, etc.

When we talk about corpus today, we mention digitalized (computerized) collection of texts and the tools that supply the usage of the corpus. However, even not called as "corpus linguistics", the text collection and studying of language via this collection has deeper roots. In 1897 Kading collected an enormous number of words for his age. In 1912 J.B. Estoup formed a French corpus for his studies on frequency graphemes. In 1913 Markov and in 1936 Zipf compiled English corpora as well.

Another linguist Boas studying American-Indian language collected texts in 1940.

In this study, we will concern with the existing corpora, especially BNC and how it came to life. More than all, the main concept is to build an online sample corpus of Turkish which has never built before.

Today there are many well-known corpora accepted as representative and reliable. Some of them are; British National Corpus (BNC), American National Corpus (ANC), Bank of English. Brown Corpus, etc. In this paper the well-known British National Corpus (BNC) will be the base both in design and criteria. The corpus that we build is TsCorpus (will called as TsC) and will try to sample a simple corpus building with basic features.

CHAPTER I: An Introduction to Corpus Linguistics

I.1 What is Corpus and Corpus Linguistics

The definition for corpus in Longman Dictionary is "*a large collection of written or spoken language, that is used for studying the language*". Actually, this definition is clear enough for users of the dictionary. But from the viewpoint of linguists, there is much more to add on.

Below there are descriptions of "corpus" by important scholars. One of the early definitions of corpus comes from Francis:

> "*corpus as a collection of texts assumed to be representative of a given language, or other subset of language, to be used for linguistic analysis*".

Similarly Sinclair (1991) define corpus as:

> "*corpus is a collection of some pieces of language that are selected and ordered according to explicit linguistic criteria in order to be used as a sample the language*".

McEnery, Hardie and Baker puts a slightly different definition that underlines the digital side of corpora.

> Corpora are usually large bodies of machine-readable text containing thousands or millions of words. A corpus is different from an archive in that often (but not always) the texts have been selected so that they can be said to be representative of a particular language variety or genre, therefore acting as a standard reference.
>
> (McEnery, Hardie and Baker 2006)

Like many other definitions given by the scholars above, all these definitions are related not only with "a simple" collection of texts, but also the criteria and some specific properties. The main concepts of corpus linguistics will be argued in the next chapter.

In our era, in linguistics, corpus stands for a large collection of texts, that are combined according to rules of corpus linguistics, digitalized according to standards of corpus linguistics and served to users with useful interface and tools.

Corpus linguistics aims to pull of examples of language from a large scale of texts, including spoken and/or written language. Chomsky underlines that, the daily language may have many mistakes, so the examples that are going to be used in linguistic studies should be driven from carefully created samples.

Despite to Chomsky, corpus linguistics defend that the authentic texts represent the language in use better and can carry out better samples which can drive investigators to the solution that they investigate.

I.2. Corpus Related Works in Turkish

The Brown Corpus had been build in 1960's, and there are number of former corpus linguistics studies in English as well; the very first corpus related works in Turkish traces back to late 1980's. One of the first computational linguistics study belongs to Aydın[1]. His thesis was related to forming a morphological parser for Turkish. In the following years, interest on the topic increased.

The Turkish Treebank (Oflazer et al. 2006), created by the Middle East Technical University and Sabancı University. This Turkish Treebank is based on a small subset of the METU Turkish Corpus.

METU Turkish Corpus is a balanced collection of 2 million words from 10 genres. The collected texts. METU includes

1 "Türkçenin Özdevimli Biçimbilgisi Çözümlemesine İlk Yaklaşım" Aydın, K., Hacettepe University. 1975.

There are two on going corpus at the time this study is being held. The first is METU spoken Turkish Project.[2] The project aims to build a spoken corpus of Turkish consisting of one million word that all linguistically analyzed.

The other project, Turkish National Corpus (TNC) is founded by TUBİTAK[3] and studies are going on at Mersin University.

TNC aims to build a Turkish corpus that will include both spoken and written language samples, consist of 50 million-POSTagged words.

Compared to METU Corpus, both these projects could be available via internet.

I.3. POSTag and Parsing Related Works in Turkish

In Turkish, the pioneer POSTagging and morphological parser studies came from Oflazer (1994). Oflazer worked with PC-KIMMO environment to test a tagger. In 1994 he implement Tomita's parser which had developed in Carnegie Mellon University during machine translation studies.

The very recent natural language processing project of Turkish is Zemberek[4]. Zemberek started as a built-in spell checker plug-in for the open sourced operating system Pardus, developed by TUBITAK, then became an independent project. Zemberek is now a platform independed, general purpose Natural Language Processing library and tool set that can handle Turkish and Turkic languages.

2 http://std.metu.edu.tr/
3 Scientific and Technological Research Council of Turkey
4 http://code.google.com/p/zemberek

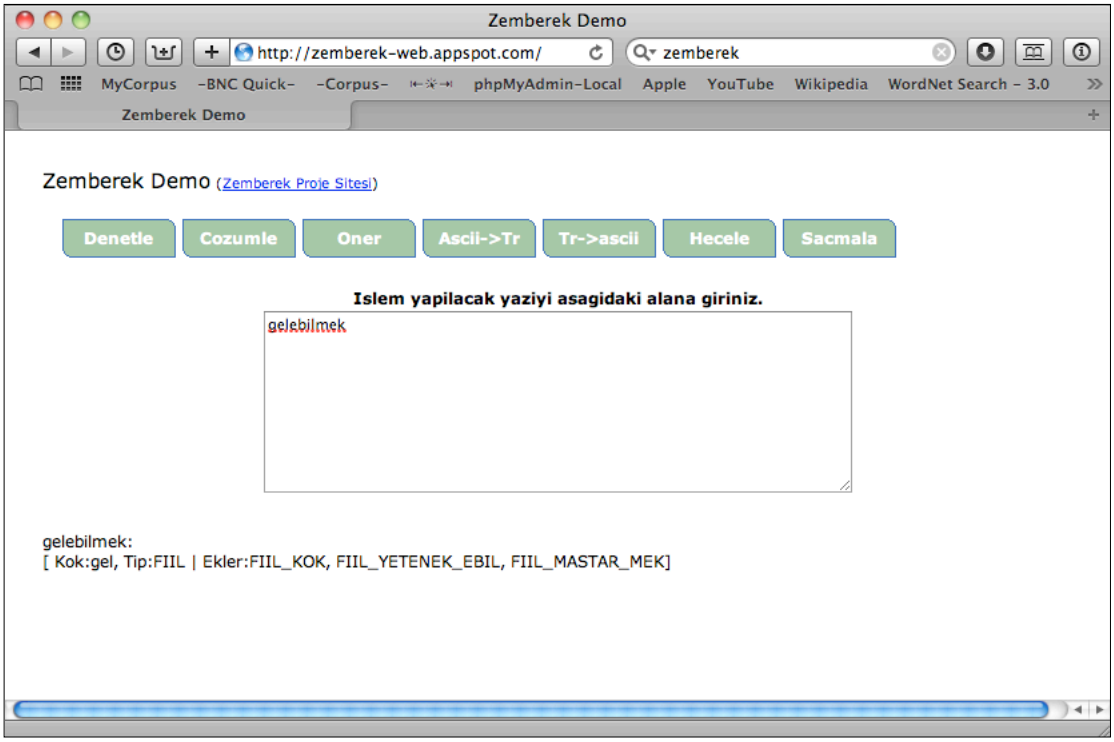


Figure 1. Zemberek online demo

This figure above represents the result for parsing the word *gelebilmek*. In this case Zemberek solves only one parsing for the input word. On some cases, Zemberek parses more than one solution for the input word. Figure 2 shows how more than one solution is represent by Zemberek.

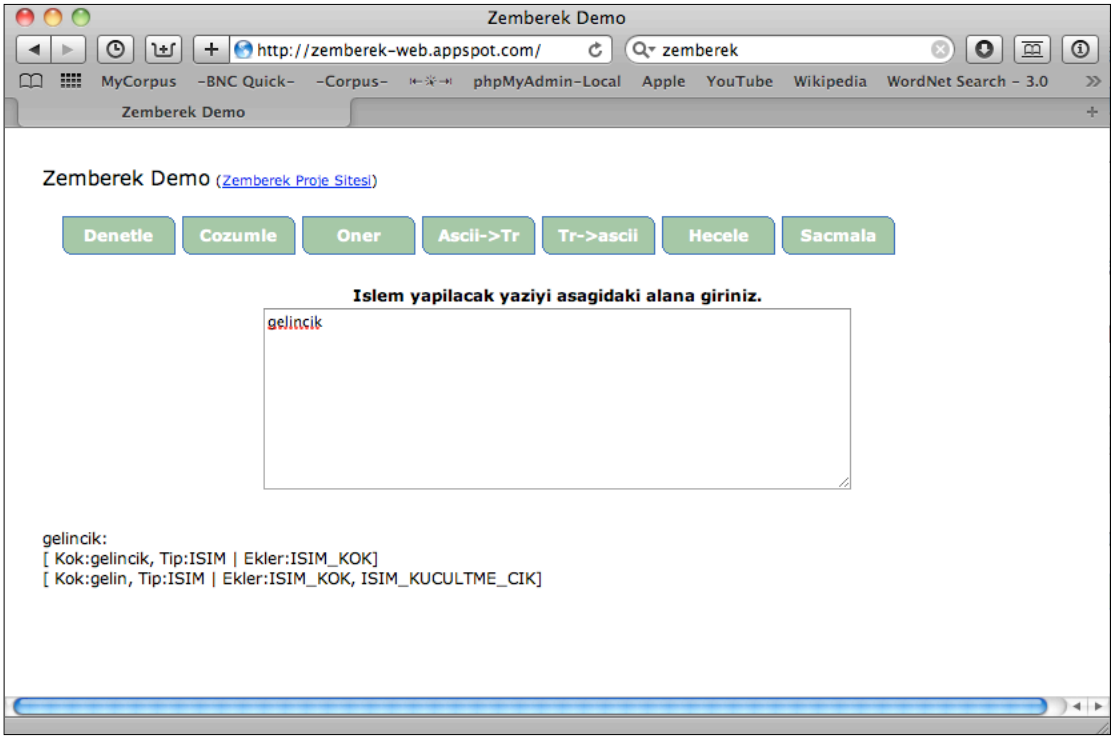


Figure 2. Zemberek parsing example

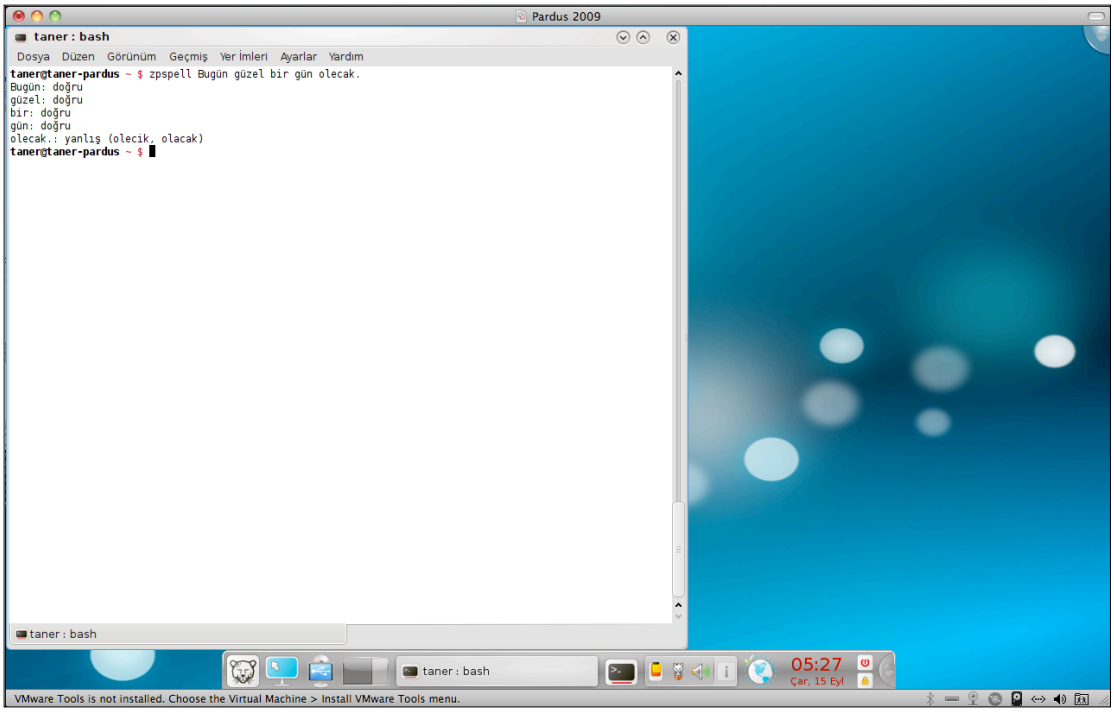


Figure 3. Zemberek Bash Interface

Figure 3 is the screenshot from Pardus 2009.2 bash. Zemberek bash command *zpspell* is used for spell checking for the sentence *Bugün güzel bir gün olecak*. The verb *olacak* had misspelled on purpose. Zemberek offered *olecik* and *olacak* as the correct form.

Using source code of Zemberek, the input text can be processed to form a text file with the suggestions of POSTagged forms. If the all suggestions are the same word classes for a word, as seen on figure 2, then the correct POStag could be attached to the word. In some cases Zemberek offers more than one solution that are also different word-classes.

Figure 4 shows the suggestions for the word *gelecek* parsed by Zemberek. The solution set contains both the NOUN and VERB classes for the input.

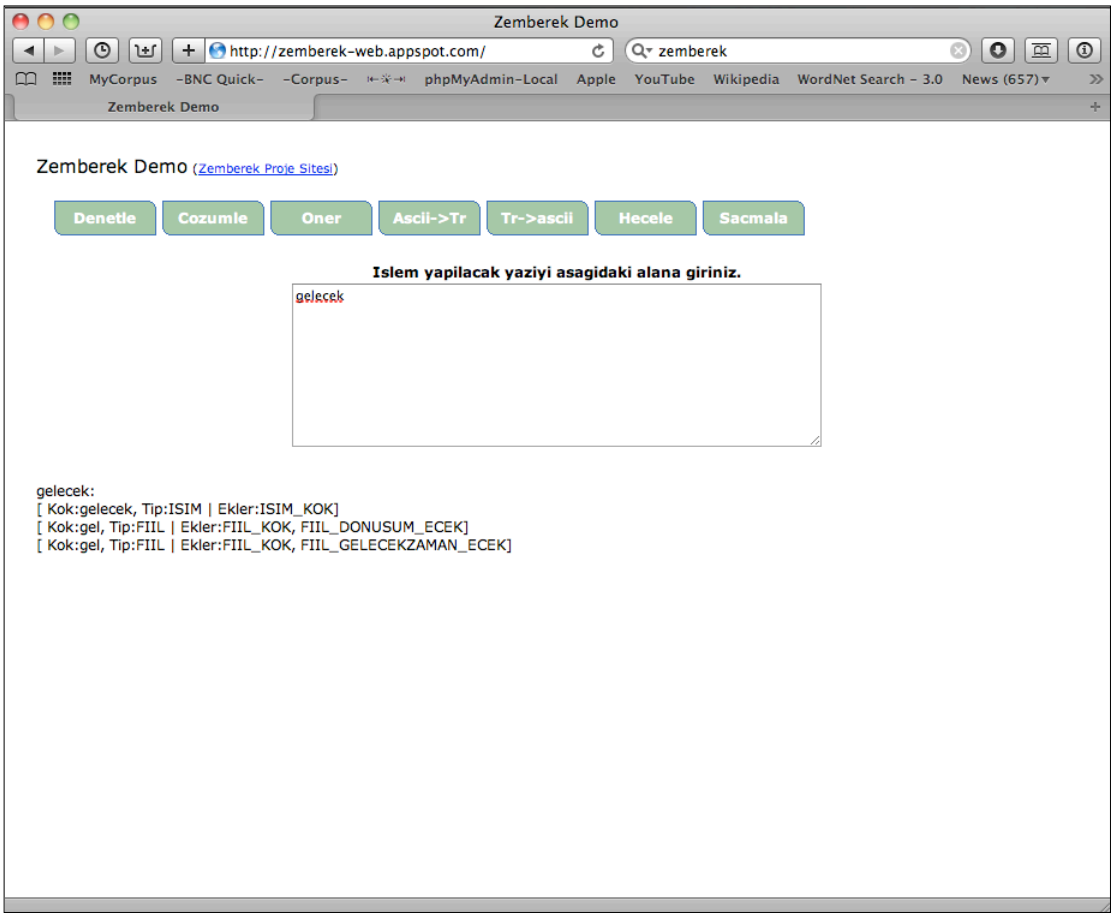


Figure 4. Zemberek parsing

In this case, where Zemberek outputs more than one solution, the solution set should be checked statistically and/or manually to found out the correct POSTag.

As Zemberek is not designed as a morphological parser, it has a very limited functionality on morphological processes. Therefore the rule sets for Zemberek should be developed in order to use it a morpholocial tools. Below is an example of Zemberek code:

```
ekle(uretici(SESSIZ_YUMUSAMASI, new Yumusama()).
                sesliEkIleOlusur(true).
                yapiBozucu(true));
ekle(uretici(SESSIZ_YUMUSAMASI_NK, new YumusamaNk(alfabe)).
                sesliEkIleOlusur(true).
                yapiBozucu(true));
ekle(uretici(ISIM_SESLI_DUSMESI, new AraSesliDusmesi()).yapiBozucu(true));
ekle(uretici(CIFTLEME, new Ciftleme()).
                sesliEkIleOlusur(true).
                yapiBozucu(true));
ekle(uretici(FIIL_ARA_SESLI_DUSMESI, new AraSesliDusmesi()).yapiBozucu(true));
        ekle(uretici(KUCULTME, new SonHarfDusmesi()).yapiBozucu(true));
        Map<String, String> benSenDonusum = new HashMap<String, String>();
        benSenDonusum.put("ben", "ban");
        benSenDonusum.put("sen", "san");
ekle(uretici(TEKIL_KISI_BOZULMASI, new YeniIcerikAta(alfabe, benSenDonusum)).yapiBozucu(true));
        Map<String, String> deYeDonusum = new HashMap<String, String>();
        deYeDonusum.put("de", "di");
        deYeDonusum.put("ye", "yi");
        ekle(uretici(FIIL_KOK_BOZULMASI, new YeniIcerikAta(alfabe, deYeDonusum)).yapiBozucu(true));
```

KIMMO uses rule sets. Below is an example from Kimmo rule set:

```
; devoicing
RULE "  {b, d}:{p, t} <=> _# | _ +:0 (X:0) [CONS | c:C]" 7 10

    b d  b  d  #  +  X  CONS  c   @
    p t  @  @  #  0  0  CONS  C   @
 1:  2 2  3  3  1  1  1    1  1   1
 2.  0 0  0  0  1  4  0    0  0   0
 3:  2 2  3  3  0  6  1    1  1   1
 4.  0 0  3  3  0  0  5    1  1   0
 5.  0 0  3  3  0  0  0    1  1   0
 6:  2 2  0  0  1  1  7    0  0   1
 7:  2 2  0  0  1  1  1    0  0   1
```

I.4. A Posttagger Model

The ongoing TNC project aims a POSTagged corpus. Therefore, a PosTagger should be coded. Below is the flowchart of TNC POStagger model.

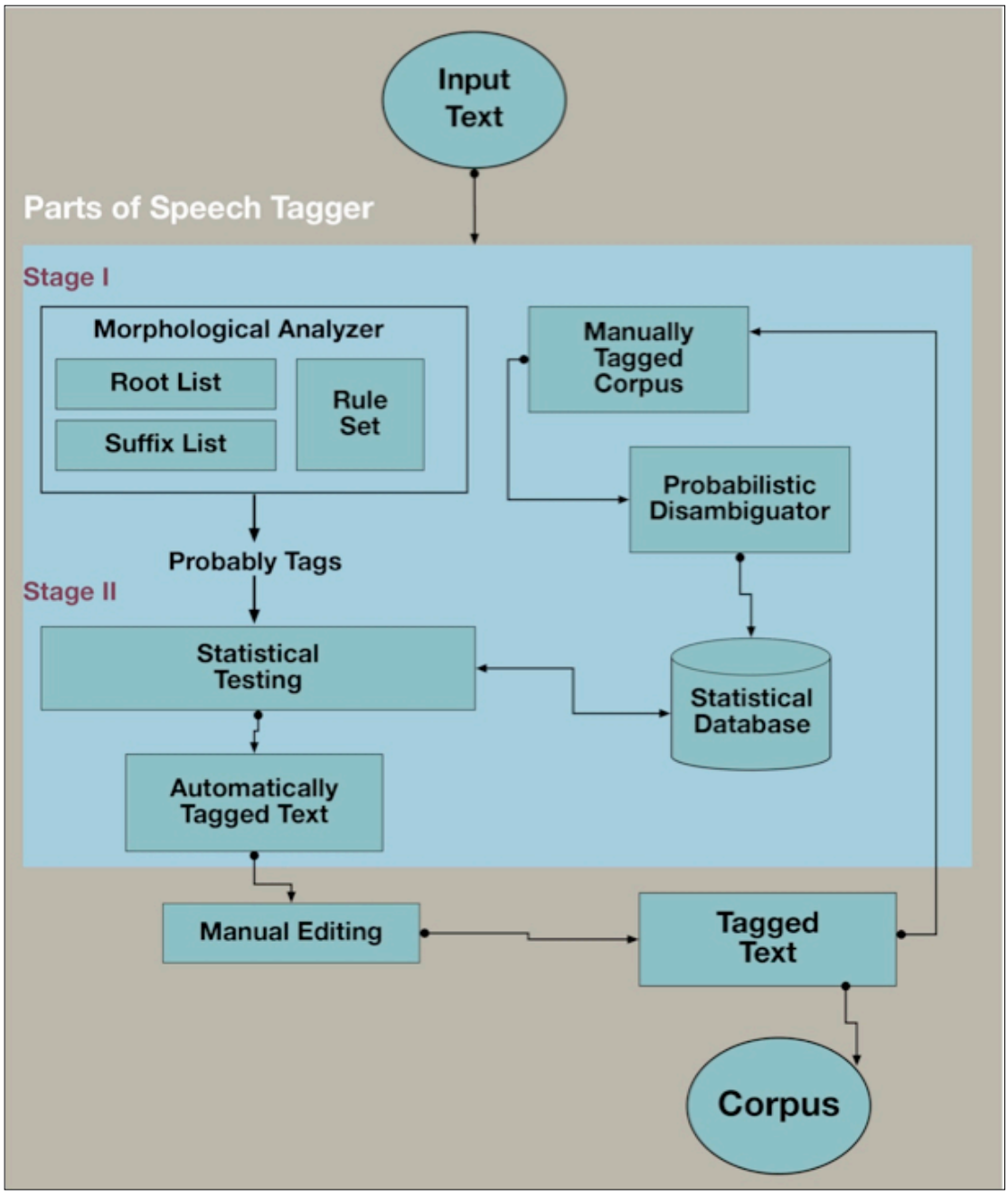


Figure 5. TNC POSTagger Flowchart

It's no need to say that the input text is already processed to form processable data. These pre-processes (tokenization, lemmatization, etc.) argued on the next chapter. One other pre-process is processing the input text by matching the lexicon. If the input word is a match of lexicon then the probable tag is added directly.

As the first step, input text is sent to a morphological analyzer. This analyzer has a rule set, and a list of roots and fixes. The probable tag is added to the word and the output is sent to statistical testing which uses a statistical database.

Statistical dependency parsers first compute the probabilities of the unit-to-unit dependencies, and then find the most probable dependency tree T among the set of possible dependency trees (Oflazer:2006). If the word is correctly tagged than it is sent to corpus database.

TNC POSTagger will also use a manually tagged corpus in order to increase the success ratio.

CHAPTER II: Concepts of Corpus Linguistics

It's not surprising that corpus linguistics has been built up it's own lexicon. In this chapter we will focus on these concepts which are the keywords for corpus linguistics. Some of the titles that are going to be mentioned below would be familiar for the people with the knowledge of computer literacy. The titles below are not deeply discussed but handled with their most basic properties. Further information will be given in following chapters.

II.1. Character Encoding and Encoding

The character is the smallest unit in language processing. For computers, every text is formed by characters. Every stored text has more complicated signatures included in the file header. This header contains different information about the file that depends on the file type. For text files, one of the vital information stored in the header is the encoding of the file.

On the computational side, this encoding is called as "*character encoding*" or "*character set*". Basically, the character encoding is a map of characters that are used and allowed to use in a file. There are many different character encodings. Some encodings are operation system depended. "*Mac Turkish"* encoding or "*Windows 1254"* are Turkish encodings for these operating systems. Some others like "*ISO 8859-1* (Western Europe)", "*ISO 8859-9"* (Western Europe with amended Turkish character set) are most commonly used for web.

For international perfection of displaying characters, "*UTF*-8" (Unicode Transformation Format) character encoding is advised. *UTF* represents a general set with subsets UTF-7, UTF-8, UTF-16, UTF-32. What makes UTF different than other character

sets is the variable byte length of each character. In UTF-8, each character is represented by 1 to 4 bytes. By this byte order and length, the first 128 character is fitted into one byte and the following 1920 is fitted into two and more bytes.

World Wide Web Consortium (W3C), The Internet Engineering Task Force (IETF) and Internet Mail Consortium (IMC) requires UTF-8 encoding for related tasks.

In corpus linguistics, as text files are the fundamental of all, corpora that are aimed to be use internationally have to build using UTF-8 character encoding.

The *encoding* is totally different from character encoding. Encoding is one of the major steps in corpus design. McEnery (2006) says that encoding "*rules the way of representing elements in texts such as paragraph breaks, utterance boundaries etc.*" We can say that an encoded text has a manual for user.

There is a consortium named Text Encoding Initiative (TEI) works on developing and maintain standards for the representation of texts in digital form. Some corpora including BNC used TEI standards.

Basically, encoding a text is compiling a list of markups in order to keep every data in an order. The specified labels are attached to the specified parts of the text. In most cases the whole text, paragraphs, sentences and words are marked.

II.2. Genre and Domain

The word genre comes from French word *class* which is originated from Latin. In corpus linguistics genre refers to *distinctive type of text*. Each text that is added to corpus database is marked for it's genre. Religion, fiction, academic writing etc. are examples of genres of BNC. The complete list of codes and genre distribution of BNC is shown in appendix A and B.

II.3. Synchronic and Diachronic

As we mentioned before, a corpus is a collection of texts and these texts are collected in order to make linguistic researches. Therefore the texts must be collected in an order.

One of the important criteria about the texts is the time period of the them. A *synchronic corpus* consists of texts that have been collected from the same time period of corpus building. Synchronic corpus allows users to make researches from the particular period of time of the target language in which it is constructed.

Opposite to synchronic corpus, *diachronic corpus* contains texts from a larger period of time. Diachronic corpus give an option to researchers to track back in time about the target language, and figure out linguistic changes.

II.4. Balance and Sampling

In corpus design, one of the major problem is making the corpus balanced and representative. Balance, sampling and representativeness are closely related. A balanced corpus is known as containing texts from a large scale of genres. McEnery (2006) explains balanced corpus as a corpus that contains texts from a wide range of different language genres and text domains. For example, it may include both spoken&written, and public&private texts. Scholars agreed that there is no scientific measure for a balanced corpus.

The most widely followed approach on balancing corpus is that corpus-builders adopt an existing corpus model when building their own corpus (Aksan&Aksan 2009).

Collecting maximum types of different genres can represent language more effectively. The genres are shown in the corpus interface as a search criteria. Most corpora allow users to make sub-corpus depending on genres.

Sampling is an other important issue about corpus design. Sampling means deciding which part of the chosen text will be used in the corpus database. BNC for example, composed of text samples generally no longer than 45,000 words. The sampling of texts may be from the beginning, end or middle of the text. Besides, by the sampling procedure, repeating words or word phrases are not allowed to be included in corpus database.

II.5. Representativeness in Corpus Design

A corpus could be called *representative* if it has enough number of words (generally more than 10 or 20 million) that are collected from a wide range of genres. In other words, the more data a corpus has the more representative it is. A representative corpus can be a source to make hypothesis about the target language. McEnery et. al explains representativeness as follows.

> For a general corpus, it is understandable that it should cover, proportionally, as many text types as possible so that the corpus is maximally representative of the language or language variety it is supposed to represent
>
> (McEnery, Xino and Tono, 2006).

Biber (1993) defines representativeness from another point. According to him "*representativeness refers to the extent to which a sample includes the full range of variability in a population."* (Here, Biber refers to language *variety* as population.)

Biber (1993) emphasizes that *"the limits of the population that is being studied must be defined as clearly as possible before sampling procedures are designed"*.

McEnery also says that there is no reliable criteria for measuring the balance and representation of a corpus.

The general idea about the balance and representativeness of a corpus is that, as more genres are added to corpus database, the corpus gets more close to be representative. BNC is commonly accepted as a representative corpus. The genres in BNC is shown in Appendix C. BNC has 24 spoken and 47 written genres in total 71 genres, ranging from emails to personal letters, classroom spoken samples to public debates.

II.6. Parsing and Part of Speech Tagging

Parsing, basically, is assigning labels to sentences and words and their constituents. McEnery(2006) says that when a text is parsed, tags are added to it in order to indicate its syntactic structure. He also exemplifies a parsed text as the start and end points of units such as noun phrases, verb phrases, and clauses would be indicated by parsing tags.

Atkins (1993) define parsing as "*to assign a fully labeled syntactic tree or bracketing of constituents to sentences of the corpus.*"

In corpus linguistics, parsing is associated with *morphological analysis*. A morphological parser (analyzer) is a computer software that determines the word structure in terms of the root, affixes and other morphological features.

Parsing involves the procedure of bringing basic morphosyntactic categories into high-level syntactic relationships with one another. This is probably the most commonly encountered form of corpus annotation after part-of-speech tagging. Parsed corpora are sometimes known as tree-banks. McEnery (2006) defines part of speech

tagging (POSTag) as "*POSTAG is a type of annotation or tagging whereby grammatical categories are assigned to words.*".

After a text had been processed by a POS tagger, each word is marked up with the appropriate tag. There are many different tag sets. CLAWS is one of the major tagset. The CLAWS1 tagset has 132 basic word tags. CLAWS2, which is developed between 1983-1986 has a revised 166 word tags.

During the tagging process, some words can not be defined to have only one word class. These words are called *ambiguous*. Ambiguity is a great problem in POS tagging. The words which are not clear belong to one specific word class are tagged as ambiguous.

Below is a sample POS tagging representation of the sentence "*I will certainly keep you informed about the corpus project.*" from BNC.

I <PNP> will <VM0> certainly <AV0> keep <VVI> you <PNP> informed <VVD> about <PRP> the <AT0> corpus <NN1> project <NN1> . <PUN>

The whole tagset of BNC which is also known as BNC basic tags set is shown in appendix D. The ambiguity tag list has shown in appendix E, and CLAWS7[5] which is also known as BNC enriched tag set is shown in appendix F. This enriched set used tagging of the core corpus of BNC which is including 2 million words both from written and spoken data.

5 See Appendix F

II.7. Tokenization and Lemmatization

In order to POS tag a text, the text must be tokenized. Tokenization is the segmentation of a text into individual word-tokens. A *token* is the word level unit in the text. In linguistics a *lemma* is the different sightings of the same lexeme in the language. For example, occurrences of words such as “dogs”, “dogged” and “dogging” are lemmas of “dog”; “does”, “did” and “done” are lemmas for “do”.

McEnery and Wilson (2001) and Leech (1997) states that “*lemmatization is important in vocabulary studies and lexicography, e.g. in studying distribution pattern of lexemes and improving dictionaries and computer lexicons*”.

During tokenization process, each unit of the text is separated into one-word-per-line format. \n stands for the new line character in computational world. With a simple piece of code, a text file can be tokenized. Below is the sample python[6] code for tokenization. This code reads the input file and replaces each space with the new line character.

```
print [l.split(“ ”) for l in file(“<filename>”).read().split(“\n”)]
```

Christ (1996) defines four main steps for efficient encoding where the transformation of the text file in one-word-per-line format is the first step.

II.8. Freqency

Frequency, in corpus linguistics, represents the ratio of a word (token) compared to the total number of words in corpus. This is called raw frequency and

6 Python is very powerful programming language.

calculated by dividing the total number of words in corpus to the number of the found result. However this number is useless since there is no other criteria to compare with.

For example the word “eternity” (searched both singular and plural form by using a regular expression) is found 426 times in BNC. The raw frequency of this token is 98.313.429/426 and this gives us the raw frequency as 230.7. As we searched the same token limited to written part of BNC, the total hit is 379 and in spoken part of BNC, the number of the hits is 47.

We should notice that, we still have not enough clue to make a hypothesis about our token as the total number of written and spoken part is BNC is 90% to 10%. Thus, the difference of the ratios should be equalized to a common base. This common base is mostly used as per million words.

In our example, the token eternity seems as used 8 times in written language but when we carry the numbers to a common base, the ratio of total hits become 421 in written and 470 in spoken which leads us to find out that the token eternity has more frequency in spoken language.

CHAPTER III: Corpus Design

Corpus design is a heavy job; planning of a corpus is a detailed work. Atkins et.al. (1991) defines five major steps for corpus design. These steps are *planning*, *permissions*, *data capture*, *text handling* and *user feedback and corpus development*.

Most of the corpora are designed by following the steps of a previously build corpus. The BNC model has been followed in the construction of the American National Corpus, the Korean National Corpus, the Polish National Corpus, and the Russian Reference Corpus (McEnery et.al: 2006).

The ongoing Turkish National Corpus (TNC) project is also modeling BNC as Aksan&Aksan (2009) say. They underline that only the necessary replacements will be done which are driven by the differences of the target language.

III.1 Planning

The planning stage of corpus building is the first and the most important part of the process. During planning, corpus developers should strictly decide what they aim. Following procedures of corpus design depends strictly on planning.

The linguistic design of the corpus, the type of corpus, how the data will be collected, how the data will be handled and how it is going to be served to users are all the inner steps of planning. Another point is planning the costs of the corpus. According to Atkins et. al. (1991) the primary stages of a well planned corpus are:

specifications and design
selection of sources
obtaining copyright permissions
data capture and encoding/markup
corpus processing.

III.1.1. Specifications and Design

As we mentioned above there are different types of corpora. Designers should choose one of these corpus types. Are they going to built a reference corpus or a monitor corpus, a synchronic corpus or a diachronic corpus? Will it be a tagged corpus or not? If the corpus is going to be a tagged corpus, how the corpus will be tagged? Which tagset is going to use? Will corpus consist of written or spoken language or both of them? How many words will the corpus consist?

If we focus on the well-known corpus BNC, the designers decided to built a reference corpus. BNC consists of 100 million words of written and spoken language which are collected in a period of thirty years (1960-1993), and marked up POS tagging using variations of CLAWS tag set.

Also the interface of the corpus is a problem that should be considered carefully. The interface should be simple enough to serve each user and should be reach by most of the common operating systems.

III.1.2. Selection of Sources

The selection of sources of the corpus database is very important as it will effect the genres and domains in the corpus. As mentioned above, as a chain reaction, the range of the genres affects the representativeness of the corpus.

BNC has a various sources including catalogs of books published yearly, bestseller lists, prize winners, library lending statistics, list of current magazines and periodicals, etc. Texts were classified into three selection features: domain (subject field), time and medium.

During the design of BNC each selection feature was separated into classes. Medium for example, separated into books, periodicals, unpublished texts etc., domain into imaginative, informative, etc.

For a Turkish corpus, unfortunately it is not that much easy to reach this kind of lists. Most of the publishing houses do not keep statistics, library lending statistics. This is an important problem to deal with. Even the most of the magazines and periodicals which are being published currently are not digitalized or hard copies of the previous issues are not archived by the publishers.

III.1.3. Obtaining Copyright Permissions

A major problem that the corpus designers have to deal with is obtaining copyright permissions. As a corpus will be open for a large group of users, and may became a commercial item, the data sources should be obtained with copyright permissions.

Atkins, et. al. (1991) underlines the importance of copyrights:

> The effect for the corpus builder is that it is quite likely that any text, (or sample of text) which is to be computerized and included in a corpus, will be under copyright protection and that permission will have to be obtained for its use.

They also state that if the corpus is to be used for commercial purposes all these copyright licenses should be clearly stated with the copyright holder.

Moreover, it is not only the texts but also the software used during corpus building and the usage of corpus should be licensed. If these software has different copyrights, then according to the license type, the necessary notifications should be added to the distributed or served corpus.

There are many different types of licenses; GPL (*GNU-General Public License*), AFL (*Academic Free License*), EULA (*End User License Agreement*), etc. Each license type has specific issues related to usage and distribution of the software.

III.1.4. Data Capture

As corpus is a collection of texts in a digitalized database, each source should be carried into digital platform. This process is called *data capturing*. Data can be carried into digital platform in three main ways.

The first source of data is the texts that are already stored on computers. These texts can be found from publishing houses and/or from web sites. Most of the major national newspapers and magazines have their issues on their web sites (For Turkish newspapers most of them are dating back to no more than mid 90's). These data could be collected easily in order to use in corpus database.

The older sources that are not already digitalized could be computerized by scanning. The scanning process consists of three steps. First of all each page should be scanned and renamed. During the renaming, a key point is adding leading numbers to the scanned pages. This protects computers to mix up the order of the pages. After renaming each page could be sent to OCR software (Optical Character Recognition). These software convert images to texts. Then the achieved text file should be saved in the format desired by corpus designers.

Finally, if the source is not suitable for scanning, it should be typed. Poor quality books or magazines, newspapers that are large in size of pages or hand writings are not suitable for OCR. These materials should be typed.

Each captured data needs one more checking. This checking process involves checking the file type and character encoding of the file, discovering any mistake carried from OCR software and spelling mistakes that could be made during typing.

Also during this checking process, operators control the text file to fit perfectly to the pre-defined standards. For the encoding and markup step, each text file should have the same properties. For example, if the encoding software has set up for handling the text according to number of the new line or end of line marks, each text should have the same property.

III.1.5. Annotation

Each text file added to corpus database should have been annotated. This annotation information enables users to create sub-corpus and make sub-searches. The genre, domain, author related, and similar information form the annotation of the file.

Leech (1997) underlines the importance of annotation: "*it (annotation) enriches the corpus as a source of linguistic information for the future research and development*".

This figure below samples the annotation information for a written text in BNC. Each piece of information seen on the left side column is also a key for search criteria.

BNC header information for file A00	
Title:	[ACET factsheets & newsletters]. Sample containing about 6688 words of miscellanea (domain: social science)
Spoken or Written:	Written
Number of Words (tagged items):	6,708
Average sentence length (<w>-tags per <s>-unit):	15.8582
Derived text type:	Non-academic prose and biography
David Lee's Genre Classification:	W:non_ac:medicine
Text type:	Written miscellaneous
Publication date:	1985-1993
Age of Author:	unknown
Domicile of Author:	unknown
Sex of Author:	unknown
Type of Author:	Multiple
Age of Audience:	Adult
Text Domain:	Informative: Social science
Perceived level of difficulty:	Medium
Medium of Text:	Miscellaneous published
Place of publication:	UK: South (south of Bristol Channel-Wash line)
Text Sample:	Composite
Estimated circulation size:	Medium
Target audience sex:	Mixed

Figure 6. BNC Annotation

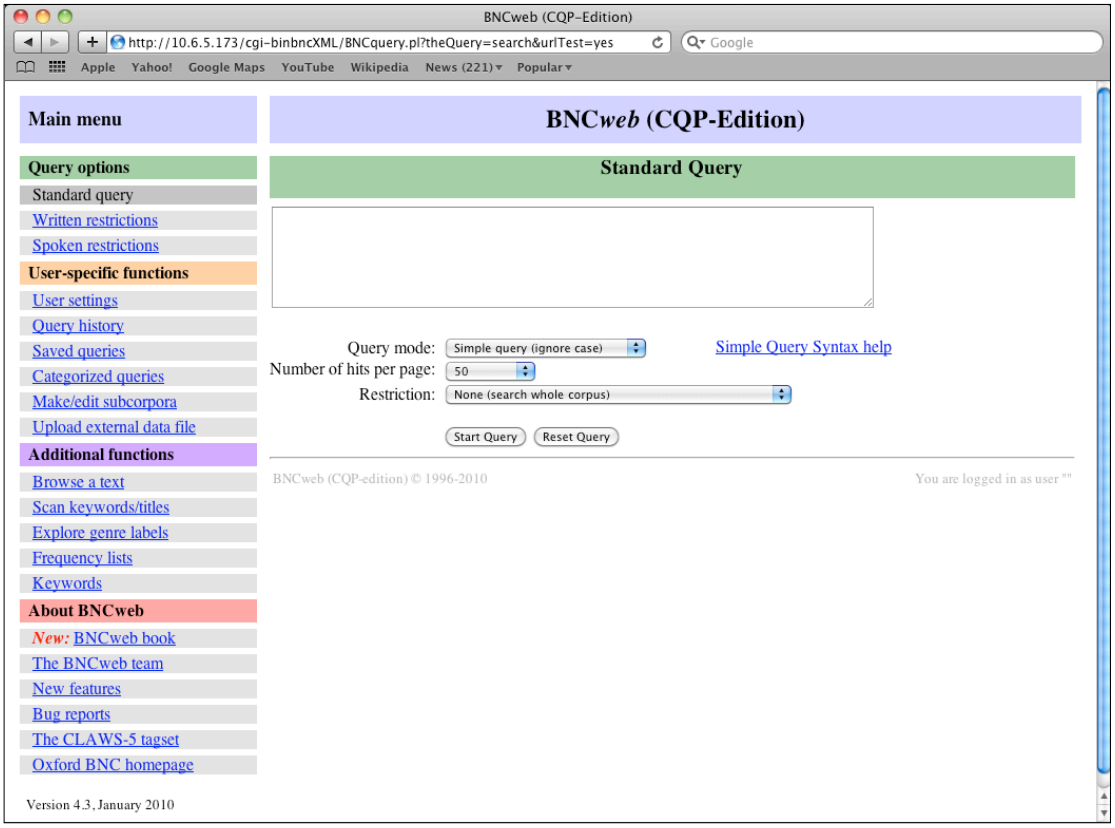


Figure 7. BNC Search Page

III.1.6. Encoding

As mentioned in the first chapter, encoding identifies how the texts will be represented by means of paragraph breaks or utterance boundaries, etc.

Most of the modern corpora use XML (Extensible Markup Language) for encoding texts. An XML encoded text can represent any division that is needed by corpus builders.

Formerly SGML (Standardized Generalized Markup Language) was used for encoding of the documents in electronic format. As internet evolves and the needs of dynamic content rises, SGML and similarly HTML needed to be updated for serving dynamic content. Therefore, the need of a new and more flexible markup language arise. The Extensible Markup Language (XML), in this mean, fits the needs of marking up the electronic documents.

XML is a general-purpose markup language. It is classified as an extensible language because it allows users to define their own tags.

The main differences between SGML and XML are:

1. XML is simpler compared to SGML.

2. XML documents should be readable with SGML parsers while some SGML might produce errors in XML parsers.

XML's primary purpose is to create an easier way for the sharing of *structured data* across different systems, especially over internet. It is used to encode documents.

A simple XML marked up data looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="sample.xsl"?>
<LIST>
<COMP>
<MODEL>PPC G4</MODEL>
<CPU>DP 1.0 GHZ</CPU>
<RAM>1 GB</RAM>
</COMP>
<COMP>
<MODEL>iMac</MODEL>
<CPU>400 MHZ</CPU>
<RAM>512 Mb</RAM>
</COMP>
<COMP>
<MODEL>iBook</MODEL>
<CPU>1.2 GHZ</CPU>
<RAM>512 MB</RAM>
</COMP>
</LIST>
```

Piece of XML marked up data

This XML sample contains data including computer models and their features. The tag names (MODEL, CPU, RAM) are defined by coder.

As mentioned above, XML is for sharing structured data. The corpus texts created using XML can be carried to any other software that can parse XML. This means that the data of the corpus is not needed to be processed each time that corpus designers want to use it with another software or in different ways.

The example below represents the XML usage from BNCWeb.

```
<wtext type="FICTION">
<s n="1">
<w c5="NN1" hw="chapter" pos="SUBST">CHAPTER </w>
<w c5="CRD" hw="1" pos="ADJ">1</w>
</s>
<p>
<s n="2">
<c c5="PUQ">‘</c>
<w c5="CJC" hw="but" pos="CONJ">But</w>
<c c5="PUN">,</c>
<w c5="VVD" hw="say" pos="VERB">said </w>
<w c5="NP0" hw="owen" pos="SUBST">Owen</w>
<c c5="PUN">,</c>
<c c5="PUQ">'</c>
<w c5="AVQ" hw="where" pos="ADV">where </w>
<w c5="VBZ" hw="be" pos="VERB">is </w>
<w c5="AT0" hw="the" pos="ART">the </w>
<w c5="NN1" hw="body" pos="SUBST">body</w>
<c c5="PUN">?</c>
<c c5="PUQ">'</c>
</s>
</p>
</div>
</wtext>
```

As we observe the tags, we can simply identify the tags and their correspondents. **<wtext type="FICTION">** is for written texts and also this tag includes the genre that the text is belong to. **<s n="1">** represents the beginning of the sentence and also includes the sentence number in the text. The tag ** ** refers to the beginning of a new paragraph. Each *c5* mark represents the tag set the text belongs to, in this case which is CLAWS5.

Notice that some tags began by **<w c5>** and some began **<c c5>**. Here “w” represents that the content of this tag is a word and “c” represents that the content of this tag is a punctuation mark.

As noticed, each tag should be closed, which means if the tag is opened as ** ** then it should be closed with the losing mark** ** to inform the computer what is included in the tag and where it begins and ends.

III.1.7. POS-Tagging

POS-Tagging of a corpus is a very detailed compilation of some processes. We will now take a peek how BNC had been POStagged. BNC had been POSTagged by using CLAWS4 automatic tagger and Template tagger.

The software used to tag BNC was *CLAWS4 automatic tagger*, which had been developed by Roger Garside at Lancaster, and a second program, known as *Template Tagger*, developed by Mike Pacey and Steve Fligelstone.

The tagging process consists of six steps, which are tokenization, initial tag assignment, tag selection, idiom tagging, template tagger and post-processing. Now we will focus on each of these steps.

III.1.8. Tokenization

For automatic tagging of the corpus data, texts should be divided into sentences and individual words. Word boundaries are mostly spaces, and are not hard to split each word from the other automatically. In III.1.7, we have sampled how to tokenize texts using a python code.

The difficult part of this process is handling sentence boundaries and separated texts into sentences. Full stops, at first, are thought to be sentence breakers but they are ambiguous as they can also be used as abbreviation marks (English B.C. or Turkish M.Ö.), between dates (24.06.1983) and to represent ordinal numbers (the representing of 1st (*first*) is “1.” in Turkish).

III.1.9. Initial Assignment of Tags

This process is assigning one or more tags to each word. For unambiguous words one tag is added to word. During this process, for ambiguous words, every possible tag is added to word. The number of the maximum possible tag is seven.

Let's observe the possible tags for the token *book* as an example. It could be both NN1(*singular common noun*) and VVB (*the finite base form of lexical verbs*).

Leech and Smith (2000) explain this procedure of CLAWS POS-tagging within two steps. In the first step software tries to match the input token with the pre-prepared lexicon. If the software succeeds POSTag the word, it attaches the tag to the word. If there are more than one possible tag for the word then software initialize a new process. Leech and Smith (2000) list this process as:

> • Look for the ending of a word: e.g. words in -ness will normally be nouns.
>
> • Look for an initial capital letter (especially when the word is not sentence-initial). Rare names which are not in the lexicon and do not match other procedures will normally be recognized as proper nouns on the basis of the initial capital.
>
> • Look for a final -(e)s. This is stripped off, to see if the word otherwise matches a noun or verb; if it does, the word in -s is tagged as a plural noun or a singular present-tense verb.
>
> • Numbers and formulae (e.g. 271, *K9, ß+) are tagged by special rules.
>
> • If all else fails, a word is tagged ambiguously as either a noun, an adjective or a lexical verb.

III.1.10. Tag Selection (Disambiguation)

On the previous stage, every possible tag had added for the ambiguous words. Now, in this stage, the software elects the most possible tag among the ones carried before. It's important to notify that the tags are not deleted but ordered by means of being the most probable tag for the given word.

In this process the software uses an algorithm known as Viterbi algorithm. This algorithm depends on Hidden Markov Model (HMM) and aims to find the most probable ordering of the input possibilities.

Here we should explain HMM. HMM is a set of finite states of the possible distribution of probable situations. The figure below represents how a finite state machine handles the word taner.

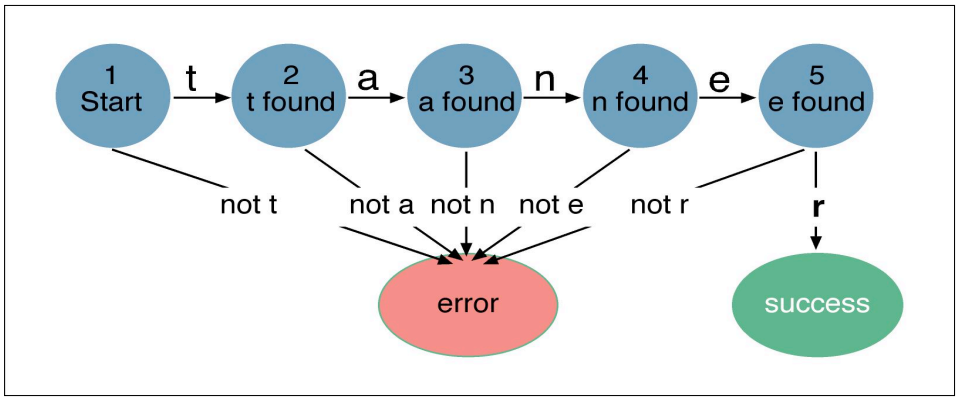


Figure 8. Finite state machine parsing of word *taner*

The algorithm starts processing the input data by the initial unit and goes on one by one. Each unit is processed as the result returns as correct. As the algorithm reaches the target, the process is terminated. In the diagram above, the algorithm began processing the word taner by the first unit “t”. If step 1does not return as “t”, the algorithm then tries the second unit. This process keeps on working until the target unit is reached. For each unit in the input data, algorithm runs the same process and tries to reach the target unit.

After this stage is completed 95-96% percent of the input text will be tagged correctly.

III.1.11. Idiomtagging

During this process, words and tags are matched with a previously prepared list. Leech and Smith (2000) call this list as a *template.* A sample of the content of this template is as following:

> • a list of multi-words (of course, because of, etc.)
>
> • a list of place name expressions (e.g. Mount X , where X is some word beginning with a capital)
>
> • a list of personal name expressions (e.g. Dr., Mr. etc.)
>
> • a list of foreign or classical language expressions used in English (e.g. de jure, hoi polloi)
>
> • a list of grammatical sequences where there are typically 'slots' in the sequence which may or may not be filled: e.g. Modal + (adverb/negative) + (adverb/negative) + Infinitive.

Leech and Smith (2000) states the importance of idiomtagging.

The correct tag can only be selected if CLAWS looks at a word+tag sequence as a whole. In other words, idiomtagging is more powerful than the Viterbi disambiguation algorithm because it is able to operate on a “window” of several word tokens at once.

III.1.12. Post Claws -Template Tagger

CLAWS has a success rate of 97%. The left 3% percent of POStagging is completed in this stage. *Template Tagger* is an other software which is capable of processing the output of CLAWS as input data.

Rather than developing/working on CLAWS4 software, builders of BNC decided to focus on the rule-side of tagging process.

> "For the BNC Tagging Enhancement project, we decided to concentrate our efforts on the rule-based part of the system, where most of the inroads in error reduction had been made. This involved (a) developing software with more powerful pattern-matching capabilities than the CLAWS Idiomlist, and (b) carrying out a more systematic analysis of errors, to identify appropriate error-correcting rules." (Leech:2000)

III.2 Corpus Processing - Corpus Workbench (CWB)

As all the stages stated above are done, the chosen texts are ready to use. The usage of this data needs another bunch of software including background applications that are not seen by users and others that supply an interface for users.

Here, we will investigate the BNCweb XML edition and the software that processes it.

CWB is the combination of open sourced software that can handle and process large amounts of data up to two billion words with linguistic annotation.

The main part of CWB is corpus query processor (CQP) and CQPWeb. CQPweb serves the web interface of the corpus.

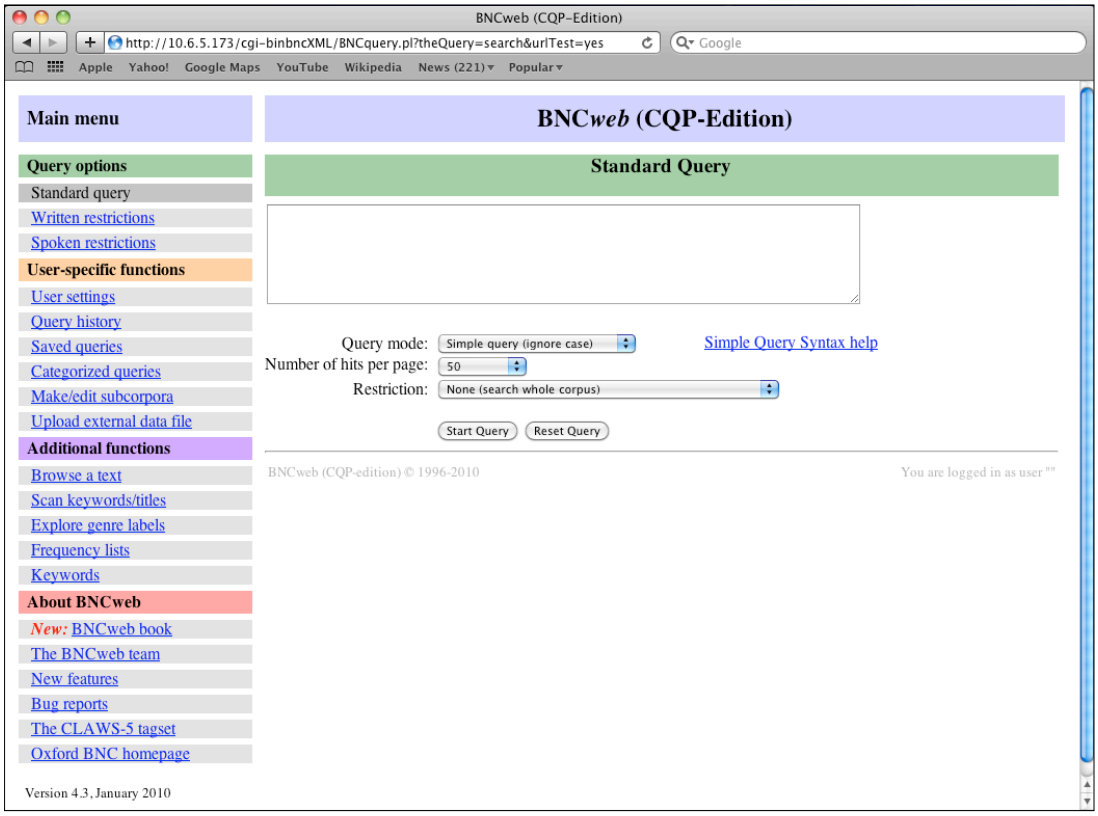


Figure 9. BNC Interface

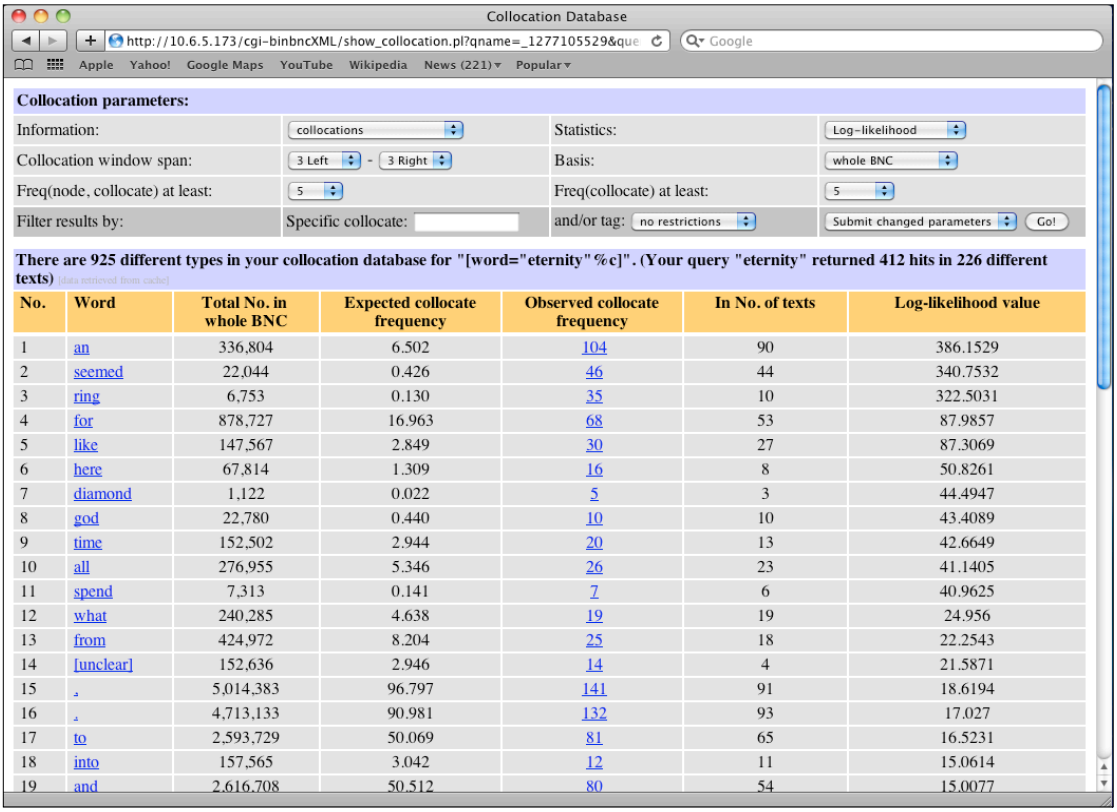
CQP also administrate other jobs including caching of the pre-searched data, query-sorting, collocations, distribution, creating sub-corpora, usage of regular expressions, etc.

III.2.1. Caching

Each query is cached by CQP. This is a very useful solution as researcher works with the same token. Caching helps the software for not making the same query each time it is called, but also calling the results from the cached base. By this way, processor usage is reduced and the results can reach faster.

III.2.2. Collocations

One of the main features of CQP is representing the collocations of the searched token. Collocations represent the environment of the token searched for. CQP can handle a range of different collocation statistics. Below are the collocation results from BNCWeb for the token eternity.



No.	Word	Total No. in whole BNC	Expected collocate frequency	Observed collocate frequency	In No. of texts	Log-likelihood value
1	an	336,804	6.502	104	90	386.1529
2	seemed	22,044	0.426	46	44	340.7532
3	ring	6,753	0.130	35	10	322.5031
4	for	878,727	16.963	68	53	87.9857
5	like	147,567	2.849	30	27	87.3069
6	here	67,814	1.309	16	8	50.8261
7	diamond	1,122	0.022	5	3	44.4947
8	god	22,780	0.440	10	10	43.4089
9	time	152,502	2.944	20	13	42.6649
10	all	276,955	5.346	26	23	41.1405
11	spend	7,313	0.141	7	6	40.9625
12	what	240,285	4.638	19	19	24.956
13	from	424,972	8.204	25	18	22.2543
14	[unclear]	152,636	2.946	14	4	21.5871
15	,	5,014,383	96.797	141	91	18.6194
16	.	4,713,133	90.981	132	93	17.027
17	to	2,593,729	50.069	81	65	16.5231
18	into	157,565	3.042	12	11	15.0614
19	and	2,616,708	50.512	80	54	15.0077

Figure 10. BNC Collocation

III.2.3. Distribution

CQP can serve the results for the searched token by means of its distribution compared to the other categories in the corpus. The figure below shows how BNCWeb serves the distribution for the token eternity.

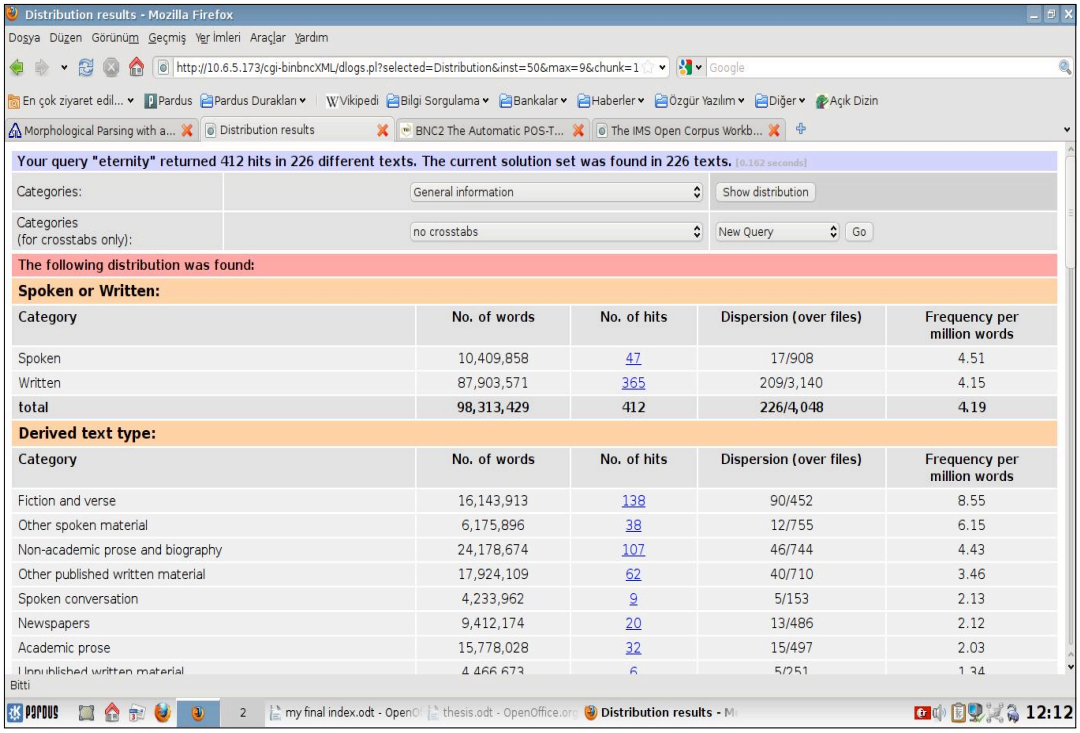


Figure 11. BNC Distribution

III.2.4. Sub-corpora

Sub-corpus is a part of the corpus created by using a limited part of the main corpus. If the user makes queries only from the data taken from newspapers in whole BNC, we can say that the user is using a sub-corpus.

CQP enable users to create sub-corpora from the main corpus.

III.2.5. Regular Expressions

CQP has a built-in tool called CEQL that can serve powerful tools for making queries using regular expressions.

A regular expression is string that represents a specific pattern. CQP can handle both universal regular expressions and the ones associated to the corpus.

The asterisk (*) for instance represents any character. It is universally in use ranging operating systems to web, programming languages to console emulators. In a

terminal emulator window, the regular expression “*.txt” represent any file that has the extension “.txt”.

The command below samples how we can count the total number of the words in all the text files in a directory.

wc -w *.txt

The command wc (word count) with the parameter -w (word) counts the words in the files that we limited to as to have any name but the extension “.txt”.

The characters “? * + [ ] ( ) | _ { } < >” are all used to describe patterns for regular expressions.

BNCWeb has also its own marks for queries with regular expressions. The query below shows the results for token “etern*” that derives all the results for every hit that begins with “etern”.

No	Filename	Hits 1 to 50 Page 1 / 28		
1	GW2 2892	dreamed of once, where I wandered through its rooms for an	eternity	.’ The young man glanced behind himself dubiously, into the
2	A04 657	extreme simplicity of character. For he had revealed one of the	eternally	popular effects of nature, one which is still quoted by simple
3	A68 1512	he himself and the other hearers were electrified. Ramsey spoke of	eternal	life. The witness judged part of the secret to lie in
4	EWR 463	. In bringing into play this powerful discourse on art and the	eternal	"qualities of the human spirit to justify the distinctiveness of En
5	H9L 1643	sensed would last a long, long time, if not an	eternity	. His hand toyed mercilessly with her breasts, as if he
6	EEM 1265	more than once, or that their inhabitants might be restored to	eternal	life without a knowledge of the Son of God. But Genesis
7	CAW 766	perception, and as a repetition in the finite mind of the	eternal	act of creation in the infinite I AM’ (Biographia Literaria
8	CB8 824	mascara and the office experience shouldn't mix, and that spraying	Eternity	behind your ears before a conference devalues your contributio
9	FB4 228	witness that a binding contract was being undertaken. The subjects vowed	eternal	fealty to the ‘White Tsar’, to be loyal and
10	FET 2229	she was the beloved, the tended saint — dying gladioli,	eternal	silk blown roses, immortelles. Behind her on the altar was
11	G10 1046	had turned sooner than he had thought. Alone, in the	eternal	light of his timeless Prison, Fergus winced away from the mem
12	BNF 557	sometimes ‘No’. One ‘No’ does not mean	eternal	‘Noes’. The child who does not learn to ask
13	B13 73	on Napoleon. In 1907 the Reverend C S Greaves discussed the	Eternal	City, Eldorado and his own travels in Palestine. The Reverend
14	J2B 223	miracle to those of us who have been contemplating what seemed an	eternal	building site. An advance group of undergraduates moved in fo
15	CL6 1178	Trinity is presented as intrinsically bound up with the incarnation of the	eternal	Son as Jesus Christ, and as supplying the ultimate framework fo
16	CM1 308	beast. Spirits of the dead drift over the battlefield locked in	eternal	battle. The mist itself is red-tinted and smells of blood.
17	A6S 33	Marx first had to show that capitalism is not based on some	eternal	immutable truth, as presented by economists, but is the product
18	ASF 723	the latter's famous metaphor of time as the moving image of	eternity	, since he was more concerned to stress the difference between,
19	B0P 1217	it. Within four or five minutes, yet seeming like an	eternity	, the line begins to peel off the open spool, slowly
20	CEJ 1186	the question, ‘Master, what must I do to inherit	eternal	life?’ 5 Who were the two characters who ‘went
21	CKN 876	Wooster, after all, in the inter-war Jeeves stories, was	eternally	trying to do the Right Thing and succeeding (if at all
22	ALJ 1403	to come in after her. Her heart stopped. For an	eternal	moment, she waited for its next beat … thirteen ANNE HAD

Figure 12. BNC Regular Expression

CHAPTER IV: A Simple Sample

In this chapter we will form our own corpus with the features that we will explain below. The corpus will be called TsC. The corpus can be reached from http://derlem.tanersezer.com. The features of TsC are:

- TsC allows users to make queries within 15.000 words chosen from free licensed texts from Turkish
- TsC allows administrators to add and delete data using web interface
- TsC uses UTF-8 character encoding for database, queries and results
- TsC shows results as in list and Keyword in Context (KWIC)
- The data list that formed TsC could be seen by title, genre, author and date

IV.1. What's in The Box?

TsC build using PHP, HTML and SQL database connection. Why we choose this two? Although, programming languages like Python and Perl are much capable of processing language, in TsC, we just in need of making searches and displaying hits on screen. Almost every host provides their users php-mysql based services. Also using php and mysql locally on a computer is an easy task even for amateur users on most popular operating systems like Mac OS Classic, Mac OS X, Windows and Linux distributions.

Therefore, we preferred these two in our sample work. By this choose, we aimed our code to be applicable by other users.

Below we will focus on how the code works.

IV.1.1. SQL Database

TsC keeps its data on a SQL server. All the data is stored in UTF-8 character encoding. Each data has a unique *ID number* that is assigned to data during uploading to database.

Administrators can upload the data using a web interface. This web interface is granted by user authentication. Below is the figure of TsC data upload screen.

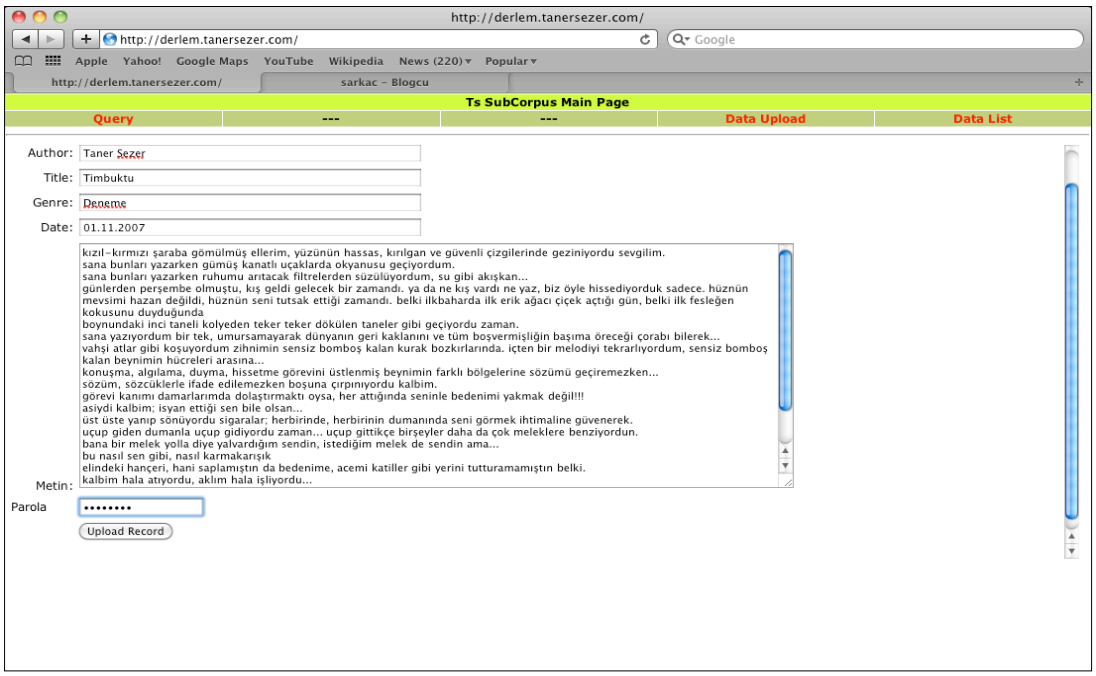


Figure 13. TsC Data upload

Authorized users can use TsC Data upload screen to add new texts to the corpus data list. 5 main text filed, including the text itself, should be filled. As the user fills all the text fields, by clicking “Upload Record” button, the new data is sent to the TsC database. Each new data automatically gets a unique id on the server side.

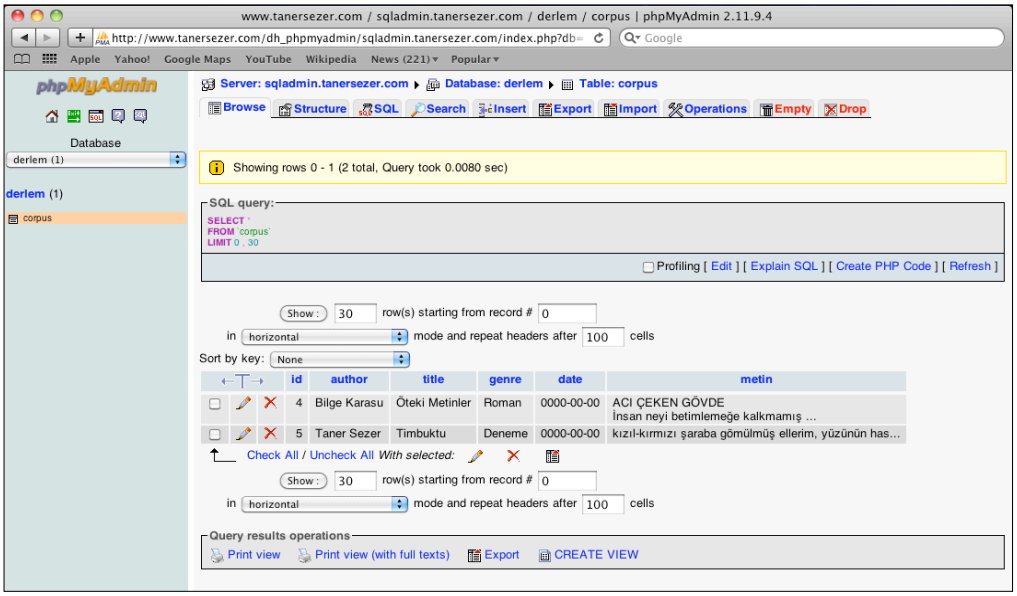


Figure 14. TsC MySQL Database

The database of TsC is so simple. It has a table with five rows keeping data of "*author*", "*title*", "*genre*", "*date*", "*metin* (text)" and "*id*". The "id" is a unique data and given to the data during upload.

IV.1.2. PHP Code

Both the search and serving the results processes are built by using PHP. The code is compiled of 9 php different files where 1 file supplies the connection to SQL database, 1 supplies user authentication, 2 supplies the query actions and the rest serves the results.

The query action code is given below. The code is given by line numbers in order to point which action is done by which part of the code easily.

```
1. <?php
2. require_once("db.php");
3. if (strlen(trim($_GET["q"])) > 1) {
4.     $aranan = mysql_real_escape_string($_GET["q"]);
5. } else {
6.     header("Location: query.php?$aranan");
7. }
8. $sorgu = "SELECT id, metin FROM corpus WHERE metin like '% $aranan %'";
9. $query = mysql_query($sorgu);
10. print "<html><head><meta http-equiv='Content-type' content='text/html; charset=utf-8'></head><body>";
11. print "<p>";
12. print ($_GET["t"] == 2) ? "<a href='?q=$aranan'>Liste Görünümü</a>":"<a href='?q=$aranan&t=2'>KWIC Görünümü</a>";
13. print " | <a href='query.php'>Yeni Arama</a></p>";
14. if (@mysql_num_rows($query) > 0) {
15.     if ($_GET["t"] == 2) {
16.     while ($result = mysql_fetch_assoc($query)) {
17.         $kelimeler = explode(" ", $result["metin"]);
18.         $index = 0;
19.         foreach ($kelimeler as $kelime){
20.             $punc = substr($kelime, -1);
21.             if ($punc == "." or $punc == "!" or $punc == ":" or $punc == "?"){
22.                 $yeni_cumle[] = $kelime;
23.                 if ($found){
24.                     foreach ($found_index as $i) {
25.                         $start = (($i - 5) >= 0) ? $start = ($i - 5):0;
```

```
26.                     for ($start; $start < $i; $start++){
27.                         $before[] = $yeni_cumle[$start];
28.                     }
29.                     $word = $yeni_cumle[$i];
30.                     $start = $i + 1;
31.                     $end = (($start + 6) > count($yeni_cumle)) ? count($yeni_cumle):$i + 5;
32.                     for ($start; $start <= $end; $start++){
33.                         $after[] = $yeni_cumle[$start];
34.                     }
35.                     $sentences[] = array("id" => $result["id"],"before" => @implode(" ",
$before), "word" => $word, "after" => @implode(" ", $after));
36.                     $before = array();
37.                     unset($word);
38.                     $after = array();
39.                 }
40.                 $found = 0;
41.                 $index = 0;
42.                 unset($found_index);
43.                 $yeni_cumle = array();
44.             } else {
45.                 $found = 0;
46.                 $index = 0;
47.                 unset($found_index);
48.                 $yeni_cumle = array();
49.             }
50.         } else {
51.             if (preg_match("/$aranan/i", $kelime)){
52.                 $found = 1;
53.                 $found_index[] = $index;
54.             }
55.             $yeni_cumle[] = $kelime;
56.             $index += 1;
57.         }
58.     }
59. }
60. } else {
61.     while ($result = mysql_fetch_assoc($query)) {
62.         $kelimeler = explode(" ", $result["metin"]);
63.         foreach ($kelimeler as $kelime){
```

```
64.             $punc = substr($kelime, -1);
65.             if ($punc == "." or $punc == "!" or $punc == ":" or $punc == "?"){
66.                 $yeni_cumle[] = $kelime;
67.                 if ($found){
68.                     $sentences[] = array("id" => $result["id"], "cumle" => implode(" ", $yeni_cumle));
69.                     $yeni_cumle = array();
70.                     $found = 0;
71.                 } else {
72.                     $found = 0;
73.                     $yeni_cumle = array();
74.                 }
75.             } else {
76.                 if (preg_match("/$aranan/i", $kelime)){
77.                     $found = 1;
78.                 }
79.                 $yeni_cumle[] = $kelime;
80.             }
81.         }
82.     }
83. }
84. } else { echo "<p>Sonuç bulunamadı!</p>"; exit(0);}
85. if ($_GET["t"] == 2) {
86.     $farklı = (count($sentences) > 1) ? "farklı":"";
87.     print "<p>'$aranan' araması sonucunda ".mysql_num_rows($query)." ".$farkli." metin içinde ".count($sentences)." sonuç bulundu.</p>";
88. } else {
89.     $farklı = (count($sentences) > 1) ? "farklı":"";
90.     print "<p>'$aranan' araması sonucunda ".mysql_num_rows($query)." ".$farkli." metin içinde ".count($sentences)." cümle bulundu.</p>";
91. }
92. if ($_GET["t"] == 2) {
93.     print "<table align='center' width='100%'>
94. <tr><td style='border-width: 0 0 2px 0; border-style: solid'>ID</td><td colspan='3' style='border-width: 0 0 2px 0; border-style: solid'> </td></tr>";
95.     foreach($sentences as $cumle){
96.         print"<tr><td>".$cumle["id"]."</td><td align='right'>".$cumle["before"]."</td><td align='center'>".$cumle["word"]."</td><td>".$cumle["after"]."</td></tr>";
97.     }
```

```
98. } else {
99.     print "<table align='center' width='100%'>
100.        <tr><td style='border-width: 0 0 2px 0; border-style: solid'>ID</td><td
style='border-width: 0 0 2px 0; border-style: solid'> </td></tr>";
101.        $ci = 0;
102.        foreach($sentences as $cumle){
103.            $color = (($ci % 2) == 0) ? "FFF":"FFC";
104.            print "<tr style='background-color:
$color'><td>".$cumle["id"]."</td><td>".$cumle["cumle"]."</td></tr>";
105.            $ci += 1;
106.        }
107.    }
108.    print "</table>";
109.    ?>
```

Lines 1 to 9 deal with connection to the database. Lines 10 to 14 draws the HTML visuals that is served to user including tables and buttons. Note that all between the tags “<>” and “</>” are shown on screen and are HTML codes which parsed by the web browser.

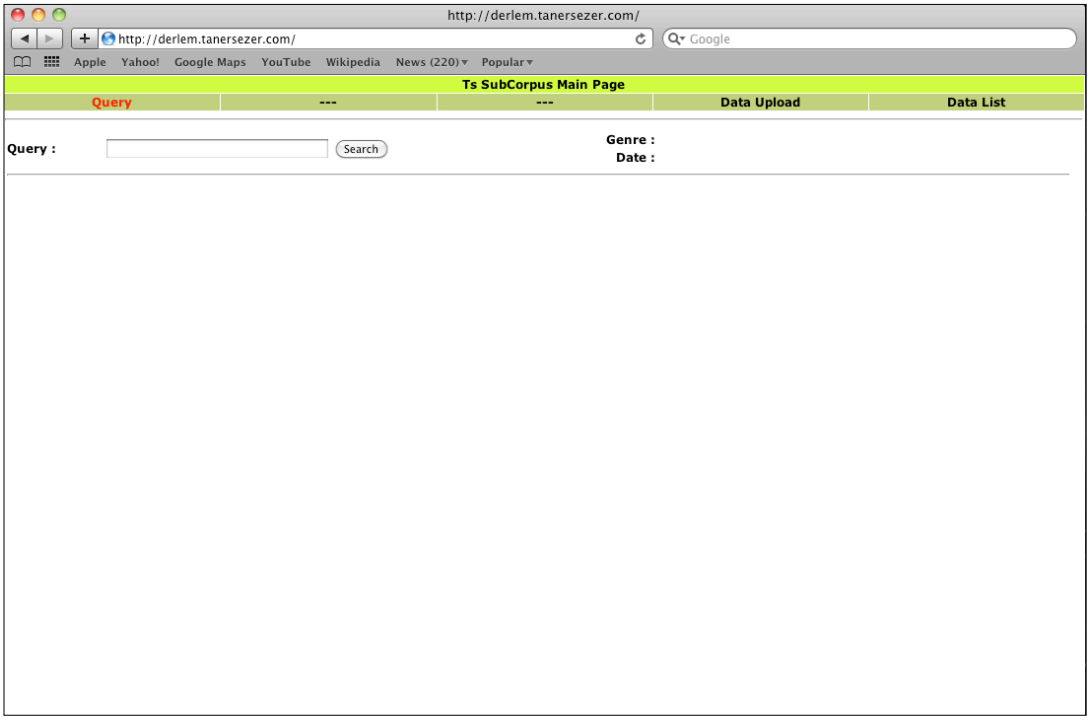


Figure 15. TsC Main Screen

The lines 17 to 22 defines the classes *kelimeler*, *kelime*, *punc* and *yeni_cumle*.

The sentence boundaries are determined by using only four punctuation marks which are “., !, : and ?”. Every sentence tokenized by using the built-in PHP command *explode*.

Line 25 defines the proximity of the searched token. The default for TsC is five words before and after token.

The lines 26 to 83 makes the query action.

The line 84 serves the “*Sonuç bulunamadı!*” warning in case no result has been found on TsC database.

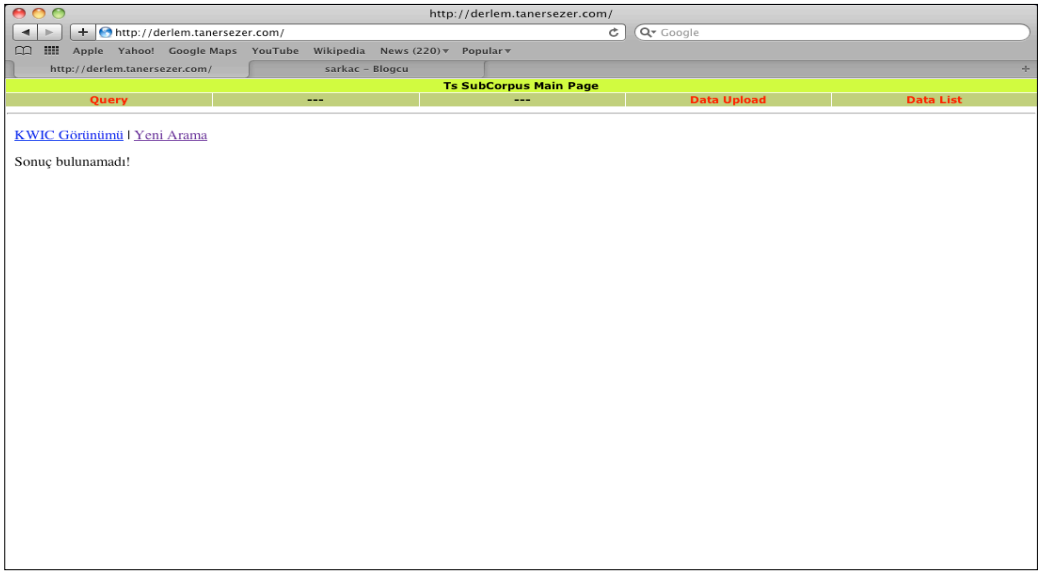


Figure 16. TsC No Result Screen

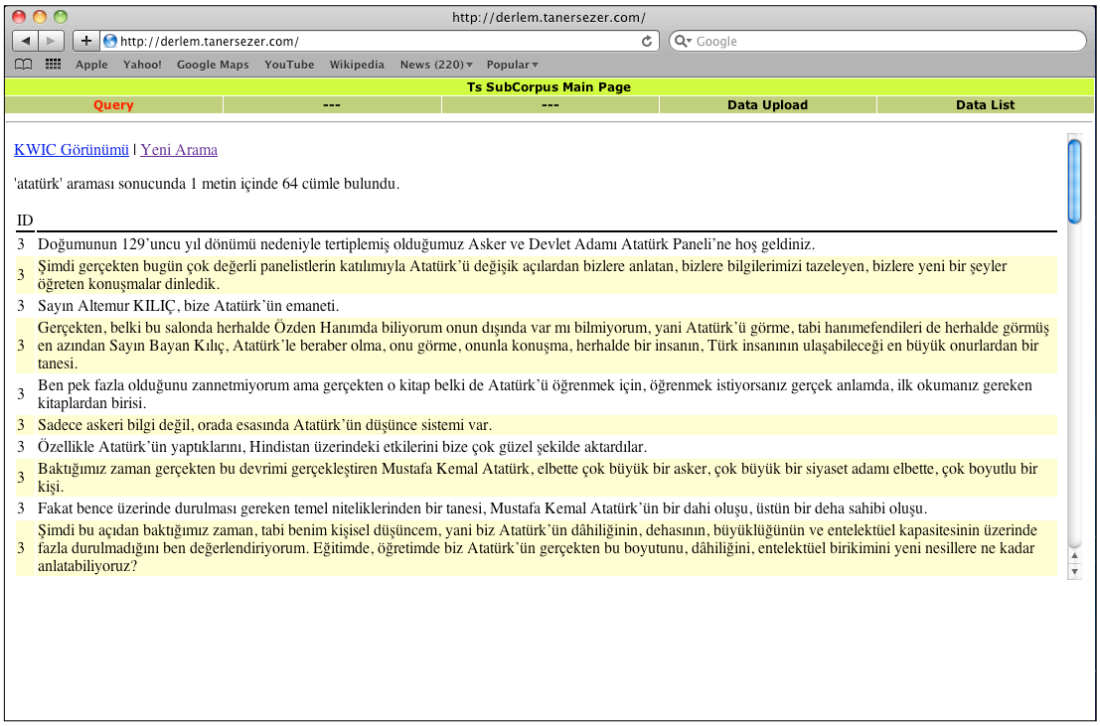


Figure 17. TsC Result Screen

Line 86 to 88 writes the result of the search to the screen. The result is given as: "[*searched token*]" araması sonucunda [*the number of texts*] metin içinde [*the number of results*] sonuç bulundu.

And finally lines 92 to 109 writes the KWIC form of the query on the screen.

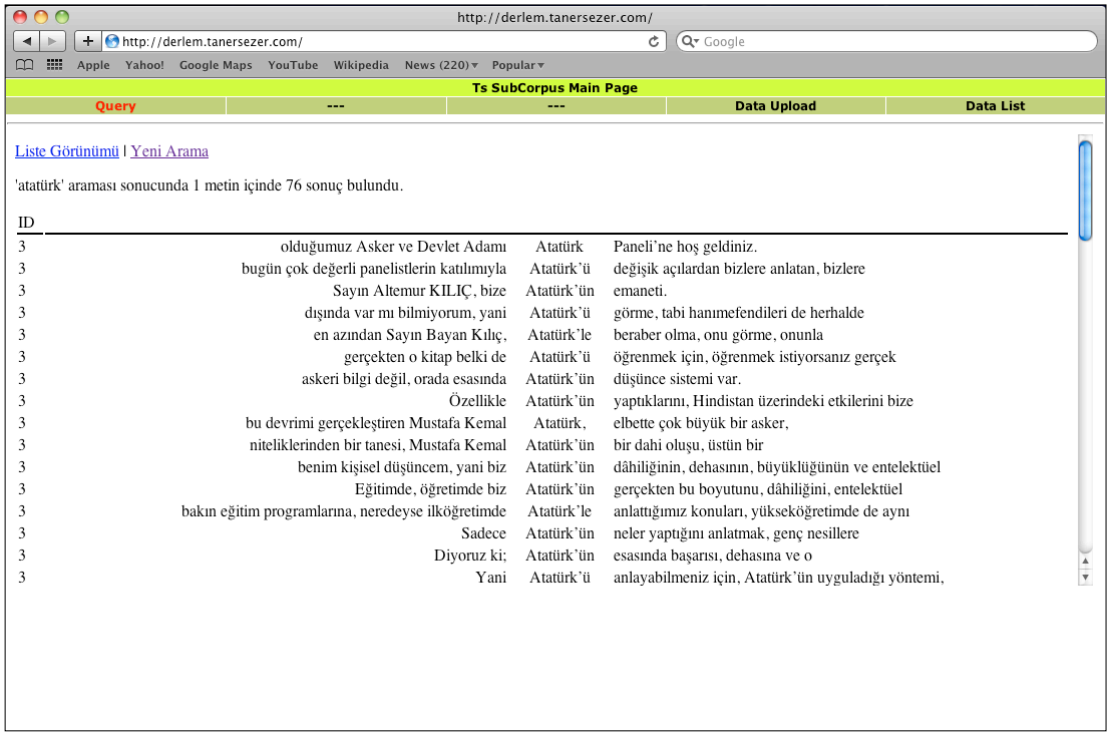


Figure 18. TsC KWIC Screen

CONCLUSION

At the time of this study is being held, the only Turkish corpus is METU corpus, consisting of 2 million words, from 14 genres. Most of the words are collected from newspapers. METU corpus is platform free as its build by using Java[7]. However METU Corpus strangely just can handle Turkish characters on Windows OS and only while the system language is set to English. Also it has a static database, which can not be modified. It runs locally as software.

The TsC tries to handle Turkish characters platform free by using UTF-8 character encoding. Both the database and user interface are capable enough to serve these characters with no problem.

The ongoing project "*Building Turkish National Corpus (TNC)*" aims to build a general reference corpus consisting of 50 million words. TNC will be POSTagged and will be served via internet.

TsC that we build within this study could not be a measure of building a Turkish corpus, but it may humbly show that it is not an extremely difficult job to construct a corpus by means of computational work. TsC is not a tagged corpus because a free or available postagger for Turkish is not available yet.

The main problem of constructing a general reference corpus of Turkish is the difficulty in bringing the two disciplines together. If the scholars and researches of computer engineers, software developers and linguists can work together a well-built corpus can be achieved.

We also try to make a work, which had never done before. The whole code, which runs the TsC, is given in this paper as open source software.

---

7 Java is a programming language aiming to create executable software for every platform.

The corpus linguistic studies in our country will reach a better point if the results are shared and the software developed are licensed as open-source software.

REFERENCES

Atkins, S., Jeremy, C. and Nicholas, O. (1992) *"Corpus design criteria"*, Literary and Linguistic Computing 7:1-16.

Biber, D. 1993. *Representativeness in Corpus Design*. Literary and Linguistic Computing 8/4:243-57

Biber, Douglas, Susan Conrad and Randi Reppen. (1998) *Corpus Linguistics*, Cambridge University Press.

Brill, E. (1992) *'A simple rule-based part-of-speech tagger'.* Proceedings of the 3rd conference on Applied Natural Language Processing. Italy: Trento.

McEnery, T., Xiao, R. and Tono, Y. 2006. *Corpus-based language studies : an advanced resource book* London: Routledge.

McEnery, T. and Wilson, A. 2001. *Corpus Linguistics.* (2nd Ed.) Edinburgh: Edinburgh University Press

McEnery, T. and A. Wilson. *Corpus Linguistics.* Edinburgh University Press, 2001

McEnery, T, R. Xiao, and Y. Tono. Corpus-Based Language Studies: *An Advanced Resource Book*, 2006

McEnery, T., Baker, P. and Hardie, A. (2006) *A Glossary of Corpus Linguistics*. Edinburgh: Edinburgh University Press

Fligelstone S., Pacey M., and Rayson P. (1997) *'How to Generalize the Task of Annotation'*. In Garside et al.

Garside, R., G. Leech and T. McEnery (eds.). *Corpus Annotation: Linguistic Information from Computer Text Corpora*, Longman, 1997.

Garside R., Leech G. and McEnery A. (eds.) (1997) *Corpus Annotation.* London: Longman.

Garside R., and Smith N. (1997) *'A hybrid grammatical tagger: CLAWS4*'. In Garside et al. (1997)

Leech, G., Garside, R., and Bryant, M. (1994). CLAWS4: *The tagging of the British National Corpus.* In Proceedings of the 15th International Conference on Computational Linguistics (COLING 94). Japan: Kyoto. (pp.622-628.)

Marshall, I. (1983). *'Choice of Grammatical Word-class without Global Syntactic Analysis: Tagging Words in the LOB Corpus'*. Computers and the Humanities 17, 139-50.

Sinclair, J. (1991). *Corpus, concordance, collocation.* Oxford: Oxford University Press.

Oflazer, K. (1994). *Two-level Description of Turkish Morphology*. Literary and Linguistic Computing, 9(2).

Oflazer, K., Eryiğit, G. (2006) Statistical Dependency Parsing for Turkish. EACL 2006

Oflazer, K., Eryiğit, G., and, Nivre, J. (2008). *Dependency Parsing of Turkish.* Computational Linguistics 34(3): (pp.357-389.)

Smith, N. (1997) *'Improving a Tagger'.* In Garside et al.

http://info.ox.ac.uk/bnc

http://www.bncweb.info

A. Genre classification for spoken texts of BNC

Code	Text
S_brdcast_discussn	54
S_brdcast_documentary	10
S_brdcast_news	12
S_classroom	59
S_consult	128
S_conv	153
S_courtroom	13
S_demonstratn	6
S_interview	13
S_interview_oral_history	119
S_lect_commerce	3
S_lect_humanities_arts	4
S_lect_nat_science	4
S_lect_polit_law_edu	7
S_lect_soc_science	13
S_meeting	132
S_parliament	6
S_pub_debate	16
S_sermon	16
S_speech_scripted	26
S_speech_unscripted	51
S_sportslive	4
S_tutorial	18
S_unclassified	44

B. Genre classification for written texts of BNC

Code	Text
W_ac_humanities_arts	87
W_ac_medicine	24
W_ac_nat_science	43
W_ac_polit_law_edu	187
W_ac_soc_science	142
W_ac_tech_engin	23
W_admin	12
W_advert	60
W_biography	100
W_commerce	112
W_email	7
W_essay_school	7
W_essay_univ	4
W_fict_drama	2
W_fict_poetry	31
W_fict_prose	485
W_hansard	4
W_institut_doc	43
W_instructional	15
W_letters_personal	6
W_letters_prof	11
W_misc	501
W_news_script	32
W_newsp_brdsht_nat_arts	51
W_newsp_brdsht_nat_commerce	44
W_newsp_brdsht_nat_editorial	12
W_newsp_brdsht_nat_misc	95
W_newsp_brdsht_nat_report	49
W_newsp_brdsht_nat_science	29
W_newsp_brdsht_nat_social	36
W_newsp_brdsht_nat_sports	24
W_newsp_other_arts	15
W_newsp_other_commerce	17
W_newsp_other_report	27
W_newsp_other_reportage	12
W_newsp_other_science	23
W_newsp_other_social	37
W_newsp_other_sports	9
W_newsp_tabloid	6
W_non_ac_humanities_arts	116
W_non_ac_medicine	17
W_non_ac_nat_science	62
W_non_ac_polit_law_edu	93
W_non_ac_soc_science	128
W_non_ac_tech_engin	123
W_pop_lore	211
W_religion	35

C. Genre and Domain of BNC

Written Domain of BNC

	texts	w-units	%	s-units	%
Imaginative	476	16496420	18.75	1352150	27.10
Informative: natural & pure science	146	3821902	4.34	183384	3.67
Informative: applied science	370	7174152	8.15	356662	7.15
Informative: social science	526	14025537	15.94	698218	13.99
Informative: world affairs	483	17244534	19.60	798503	16.00
Informative: commerce & finance	295	7341163	8.34	382374	7.66
Informative: arts	261	6574857	7.47	321140	6.43
Informative: belief & thought	146	3037533	3.45	151283	3.03
Informative: leisure	438	12237834	13.91	744490	14.92

Written Medium of BNC

	texts	w-units	%	s-units	%
Book	1411	50293803	57.18	2887523	57.88
Periodical	1208	28609494	32.52	1487644	29.82
Miscellaneous published	238	4233135	4.81	287700	5.76
Miscellaneous unpublished	249	3538882	4.02	220672	4.42
To-be-spoken	35	1278618	1.45	104665	2.09

D. THE BNC BASIC TAGSET (also known as the "C5" tagset)

Tag	Description
AJ0	Adjective (general or positive) (e.g. good, old, beautiful)
AJC	Comparative adjective (e.g. better, older)
AJS	Superlative adjective (e.g. best, oldest)
AT0	Article (e.g. the, a, an, no)
AV0	General adverb: an adverb not subclassified as AVP or AVQ (see below) (e.g. often, well, longer (adv.), furthest.
AVP	Adverb particle (e.g. up, off, out)
AVQ	Wh-adverb (e.g. when, where, how, why, wherever)
CJC	Coordinating conjunction (e.g. and, or, but)
CJS	Subordinating conjunction (e.g. although, when)
CJT	The subordinating conjunction that
CRD	Cardinal number (e.g. one, 3, fifty-five, 3609)
DPS	Possessive determiner-pronoun (e.g. your, their, his)
DT0	General determiner-pronoun: i.e. a determiner-pronoun which is not a DTQ or an AT0.
DTQ	Wh-determiner-pronoun (e.g. which, what, whose, whichever)
EX0	Existential there, i.e. there occurring in the there is ... or there are ... construction
ITJ	Interjection or other isolate (e.g. oh, yes, mhm, wow)
NN0	Common noun, neutral for number (e.g. aircraft, data, committee)
NN1	Singular common noun (e.g. pencil, goose, time, revelation)
NN2	Plural common noun (e.g. pencils, geese, times, revelations)
NP0	Proper noun (e.g. London, Michael, Mars, IBM)
ORD	Ordinal numeral (e.g. first, sixth, 77th, last) .
PNI	Indefinite pronoun (e.g. none, everything, one [as pronoun], nobody)
PNP	Personal pronoun (e.g. I, you, them, ours)
PNQ	Wh-pronoun (e.g. who, whoever, whom)
PNX	Reflexive pronoun (e.g. myself, yourself, itself, ourselves)
POS	The possessive or genitive marker 's or '
PRF	The preposition of
PRP	Preposition (except for of) (e.g. about, at, in, on, on behalf of, with)
PUL	Punctuation: left bracket - i.e. (or [
PUN	Punctuation: general separating mark - i.e. . , ! , : ; - or ?
PUQ	Punctuation: quotation mark - i.e. ' or "
PUR	Punctuation: right bracket - i.e.) or]
TO0	Infinitive marker to
UNC	Unclassified items which are not appropriately considered as items of the English lexicon.
VBB	The present tense forms of the verb BE, except for is, 's: i.e. am, are, 'm, 're and be [subjunctive or imperative]
VBD	The past tense forms of the verb BE: was and were
VBG	The -ing form of the verb BE: being
VBI	The infinitive form of the verb BE: be
VBN	The past participle form of the verb BE: been
VBZ	The -s form of the verb BE: is, 's
VDB	The finite base form of the verb BE: do
VDD	The past tense form of the verb DO: did

VDG	The -ing form of the verb DO: doing
VDI	The infinitive form of the verb DO: do
VDN	The past participle form of the verb DO: done
VDZ	The -s form of the verb DO: does, 's
VHB	The finite base form of the verb HAVE: have, 've
VHD	The past tense form of the verb HAVE: had, 'd
VHG	The -ing form of the verb HAVE: having
VHI	The infinitive form of the verb HAVE: have
VHN	The past participle form of the verb HAVE: had
VHZ	The -s form of the verb HAVE: has, 's
VM0	Modal auxiliary verb (e.g. will, would, can, could, 'll, 'd)
VVB	The finite base form of lexical verbs (e.g. forget, send, live, return) [Including the imperative and present subjunctive]
VVD	The past tense form of lexical verbs (e.g. forgot, sent, lived, returned)
VVG	The -ing form of lexical verbs (e.g. forgetting, sending, living, returning)
VVI	The infinitive form of lexical verbs (e.g. forget, send, live, return)
VVN	The past participle form of lexical verbs (e.g. forgotten, sent, lived, returned)
VVZ	The -s form of lexical verbs (e.g. forgets, sends, lives, returns)
XX0	The negative particle not or n't
ZZ0	Alphabetical symbols (e.g. A, a, B, b, c, d)

Total number of word class tags in the BNC basic tagset = 57, plus 4 punctuation tags.

E. BNC ambiguity tagset

Ambiguity tag	Ambiguous between	More probable tag
AJ0-NN1	AJ0 or NN1	AJ0
AJ0-VVD	AJ0 or VVD	AJ0
AJ0-VVG	AJ0 or VVG	AJ0
AJ0-VVN	AJ0 or VVN	AJ0
AV0-AJ0	AV0 or AJ0	AV0
AVP-PRP	AVP or PRP	AVP
AVQ-CJS	AVQ or CJS	AVQ
CJS-AVQ	CJS or AVQ	CJS
CJS-PRP	CJS or PRP	CJS
CJT-DT0	CJT or DT0	CJT
CRD-PNI	CRD or PNI	CRD
DT0-CJT	DT0 or CJT	DT0
NN1-AJ0	NN1 or AJ0	NN1
NN1-NP0	NN1 or NP0	NN1
NN1-VVB	NN1 or VVB	NN1
NN1-VVG	NN1 or VVG	NN1
NN2-VVZ	NN2 or VVZ	NN2
NP0-NN1	NP0 or NN1	NP0
PNI-CRD	PNI or CRD	PNI
PRP-AVP	PRP or AVP	PRP
PRP-CJS	PRP or CJS	PRP
VVB-NN1	VVB or NN1	VVB
VVD-AJ0	VVD or AJ0	VVD
VVD-VVN	VVD or VVN	VVD
VVG-AJ0	VVG or AJ0	VVG
VVG-NN1	VVG or NN1	VVG
VVN-AJ0	VVN or AJ0	VVN
VVN-VVD	VVN or VVD	VVN
VVZ-NN2	VVZ or NN2	VVZ

F. List of Tags in BNC Enriched Tagset (Claws7)

!	punctuation tag - exclamation mark
“	punctuation tag - quotation marks
(	punctuation tag - left bracket
)	punctuation tag - right bracket
,	punctuation tag - comma
-	punctuation tag - dash
-----	new sentence marker
.	punctuation tag - full-stop
...	punctuation tag - ellipsis
:	punctuation tag - colon
;	punctuation tag - semi-colon
?	punctuation tag - question-mark
APPGE	possessive pronoun, prenominal (my, your, our etc.)
AT	article (the, no)
AT1	singular article (a, an, every)
BCS	before-conjunction (in order (that), even (if etc.))
BTO	before-infinitive marker (in order, so as (to))

CC	coordinating conjunction (and, or)
CCB	coordinating conjunction (but)
CS	subordinating conjunction (if, because, unless)
CSA	as as a conjunction
CSN	than as a conjunction
CST	that as a conjunction
CSW	whether as a conjunction
DA	after-determiner, capable of pronominal function (such, former, same)
DA1	singular after-determiner (little, much)
DA2	plural after-determiner (few, several, many)
DAR	comparative after-determiner (more, less)
DAT	superlative after-determiner (most, least)
DB	before-determiner, capable of pronominal function (all, half)
DB2	plural before-determiner, capable of pronominal function (both)
DD	determiner, capable of pronominal function (any, some)
DD1	singular determiner (this, that, another)
DD2	plural determiner (these, those)

DDQ	wh-determiner (which, what)
DDQGE	wh-determiner, genitive (whose)
DDQV	wh-ever determiner (whichever, whatever)
EX	existential there
FO	formula
FU	unclassified
FW	foreign word
GE	germanic genitive marker - (' or 's)
IF	for as a preposition
II	preposition
IO	of as a preposition
IW	with; without as preposition
JJ	general adjective
JJR	Rgeneral comparative adjective (older, better, bigger)
JJT	general superlative adjective (oldest, best, biggest)
JK	adjective catenative (able in be able to; willing in be willing to)
MC	cardinal number neutral for number (two, three...)

MCGE	genitive cardinal number, neutral for number (twos, 100's)
MCMC	hyphenated number (40-50, 1770-1827)
MC1	singular cardinal number (one)
MC2	plural cardinal number (tens, twenties)
MD	ordinal number (first, 2nd, next, last)
MF	fraction (quarters, two-thirds)
ND1	singular noun of direction (north, southeast)
NN	common noun, neutral for number (sheep, cod)
NNA	following noun of title (M.A.)
NNB	preceding noun of title (Mr, Prof)
NN1	singular common noun (book, girl)
NN2	plural common noun (books, girls)
NNL1	singular locative noun (street, Bay)
NNL2	plural locative noun (islands, roads)
NNO	numeral noun, neutral for number (dozen, thousand)
NNO2	plural numeral noun (hundreds, thousands)
NNT	temporal noun, neutral for number (no known examples)

NNT1	singular temporal noun (day, week, year)
NNT2	plural temporal noun (days, weeks, years)
NNU	unit of measurement, neutral for number (in., cc.)
NNU1	singular unit of measurement (inch, centimetre)
NNU2	plural unit of measurement (inches, centimetres)
NP	proper noun, neutral for number (Phillipines, Mercedes)
NP1	singular proper noun (London, Jane, Frederick)
NP2	plural proper noun (Browns, Reagans, Koreas)
NPD1	singular weekday noun (Sunday)
NPD2	plural weekday noun (Sundays)
NPM1	singular month noun (October)
NPM2	plural month noun (Octobers)
PN	indefinite pronoun, neutral for number (none)
PN1	singular indefinite pronoun (one, everything, nobody)
PNQO	whom
PNQS	who
PNQV	whoever, whomever, whomsoever, whosoever

PNX1	reflexive indefinite pronoun (oneself)
PP	nominal possessive personal pronoun (mine, yours)
PPH1	it
PPHO1	him, her
PPHO2	them
PPHS1	She, she
PPHS2	they
PPIO1	me
PPIO2	us
PPIS1	I
PPIS2	we
PPX1	singular reflexive personal pronoun (yourself, itself)
PPX2	plural reflexive personal pronoun (yourselves, ourselves)
PPY	you
RA	adverb, after nominal head (else, galore)
REX	adverb introducing appositional constructions (namely, viz, eg.)
RG	degree adverb (very, so, too)

RGA	post-nominal/adverbial/adjectival degree adverb (indeed, enough)
RGQ	wh- degree adverb (how)
RGQV	wh-ever degree adverb (however)
RGR	comparative degree adverb (more, less)
RGT	superlative degree adverb (most, least)
RL	locative adverb (alongside, forward)
RP	prep. adverb; particle (in, up, about)
RPK	prep. adv., catenative (about in be about to)
RR	general adverb (actually)
RRQ	wh- general adverb (where, when, why, how)
RRQV	wh-ever general adverb (wherever, whenever)
RRR	comparative general adverb (better, longer)
RRT	superlative general adverb (best, longest)
RT	nominal adverb of time (now, tommorow)
TO	infinitive marker (to)
UH	interjection (oh, yes, um)
VB0	be

VBDR	were
VBDZ	was
VBG	being
VBM	am
VBN	been
VBR	are
VBZ	is
VD0	do
VDD	did
VDG	doing
VDN	done
VDZ	does
VH0	have
VHD	had (past tense)
VHG	having
VHN	had (past participle)
VHZ	has

VM	modal auxiliary (can, will, would etc.)
VMK	modal catenative (ought, used)
VV0	base form of lexical verb (give, work etc.)
VVD	past tense form of lexical verb (gave, worked etc.)
VVG	-ing form of lexical verb (giving, working etc.)
VVN	past participle form of lexical verb (given, worked etc.)
VVZ	-s form of lexical verb (gives, works etc.)
VVGK	-ing form in a catenative verb (going in be going to)
VVNK	past part. in a catenative verb (bound in be bound to)
XX	not, n't
ZZ1	singular letter of the alphabet (A, a, B, etc.)
ZZ2	plural letter of the alphabet (As, b's, etc.)