**Chapter 1**

**Introduction**

In this age of machines where we came across classes of data, structured, semi-structured and unstructured. Retrieving information from Structured is far too easy and can be done though a database management system. But for data that is in simple running text, its difficult to process query and retrieve information. However this has been made possible by the branch of Artificial Intelligence, Natural Language and Processing (NLP), which enables computers to derive meaning from human or natural language input, and others involve natural language generation.

* 1. **Understanding NLP**

The history of NLP generally starts in the 1950s, although work can be found from earlier periods. In 1950, Alan Turing published an article titled "**Computing Machinery and Intelligence**" which proposed what is now called the **Turing test** as a criterion of intelligence.

***“Natural language processing (NLP) is the ability of a computer to understand what a human is saying to it.”***

It is the technology for dealing with our most ever-present product: human language, as it appears in emails, web pages, tweets, product descriptions, newspaper stories, social media, and scientific articles, in thousands of languages and varieties. In the past decade, successful natural language processing applications have become part of our everyday experience, from spelling and grammar correction in word processors to machine translation on the web, from email spam detection to automatic question answering, from detecting people's opinions about products or services to extracting appointments from your email. It’s hard to even to think of preparing document without spell check or grammar check on MS-Word. Now-a-days use of NLP has gone way too smarter than just spell check, it plays major role in summarizing content, sentiment analysis and automatic information generation.

**Some Interesting Application**

1) Automatic Event Generation: Imagine a mail in your Inbox notifying in running content about an Event. The task will be made easier if NLP can mark Event on your Calendar automatically by analyzing text.

2) Information Extraction and Sentiment Analysis: Now, if a person e-shop for a cell phone, and instead of reading all the reviews if he can get reviews by dynamic appearing bar chart which depends on running text reviews. This will not only save time of consumer, but also give an edge to commercial sites.

3) Machine Translation: The translation of text from one language to another can be made automatic or manual which involves suggestion of next word in destination language can also be achieve by using NLP

4) Task like Spam Detection, Parts-Of-Speech Tagging, and Name Entity Recognition are some of the easy application which has been solved in NLP. Difficult task involves Question-Answering, Paraphrase, Summarization.

**Major tasks in NLP**

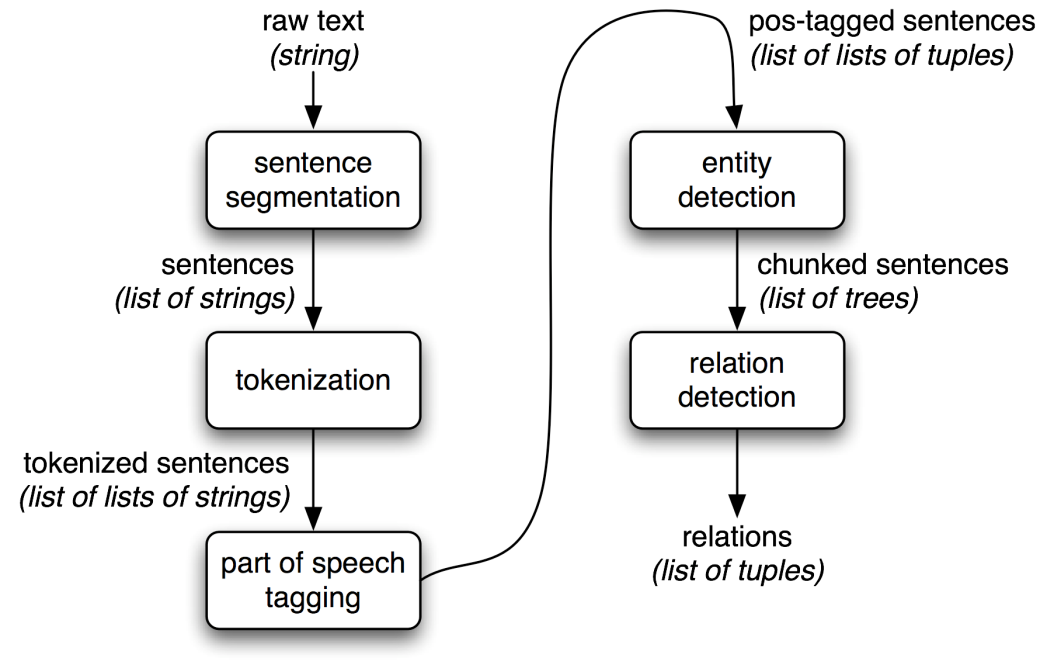
* Parts-of Speech Tagging
* Relationship Extraction
* Sentence Breaking
* Word Sense Disambiguation
* [Natural language Generation](http://en.wikipedia.org/wiki/Natural_language_generation)
* Natural Language Understanding
* Optical Character Recognition
* Information Retrieval
* Information Extraction
* Stemming
* Automatic Summarization
* Co-Reference Resolution
* Machine Translation
* Discourse Analysis
* Parsing
* Morphological Segmentation
* Name-Entity Recognition
* Question Answering
* Speech Recognition & Processing
  1. **Information Retrieval**

The meaning of Information Retrieval can be very wide. Just taking out a road map from your hand bag to see directions can be a form of information retrieval. But to be precise to academics our project deals with retrieving information from a web page by putting a query. Here we assume that a web page is known kind and thus query that could be asked will be similar or of known types. Therefore our projects works in achieving answer to such queries by making certain assumption in querying and corpus.

According to Dan Manning Information Retrieval is defines as

“***Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)”***

Fig 1.1 shows the architecture for a simple information extraction system. It begins by processing a document using several of the procedures. First, the raw text of the document is split into sentences using a sentence segmenter, and each sentence is further subdivided into words using a tokenizer. Next, each sentence is tagged with part-of-speech tags, which will prove very helpful in the next step, **named entity detection**. In this step, we search for mentions of potentially interesting entities in each sentence. Finally, we use **relation detection** to search for likely relations between different entities in the text.



*Figure 1.1 Architecture of Information Retrieval*

**1.2.1 Tokenization and Sentence Segmentation**

**Tokenization** is the process of breaking up the sequence of characters in a text by locating the word boundaries, the points where one word ends and another begins. For computational linguistics purposes, the words thus

*Input:**Tezpur University was established by an Act of Parliament in 1994*

**1994**

**in**

**Parliament**

**of**

**Act**

**an**

**by**

**established**

**was**

**University**

**Tezpur**

These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction.

* A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. A *type* is the class of all tokens containing the same character sequence. A *term* is a (perhaps normalized) type that is included in the IR system's dictionary.
* The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in a taxonomy, but in practice in modern IR systems they are strongly related to the tokens in the document. However, rather than being exactly the tokens that appear in the document, they are usually derived from them by various normalization processes.

**Sentence segmentation** is the process of determining the longer processing units consisting of one or more words. This task involves identifying sentence boundaries between words in different sentences. Since most written languages have punctuation marks which occur at sentence boundaries, sentence segmentation is frequently referred to as sentence boundary detection, sentence boundary disambiguation, or sentence boundary recognition. All these terms refer to the same task: determining how a text should be divided into sentences for further processing.

**1.2.2 Parts-Of-Speech Tagging**

The process of classifying words into their **parts of speech** and labeling them accordingly is known as **part-of-speech tagging**, **POS-tagging**, or simply **tagging**. Parts of speech are also known as **word classes** or **lexical categories**. The collection of tags used for a particular task is known as a **tagset**.

*Input:**Tezpur University was established by an Act of Parliament in 1994*

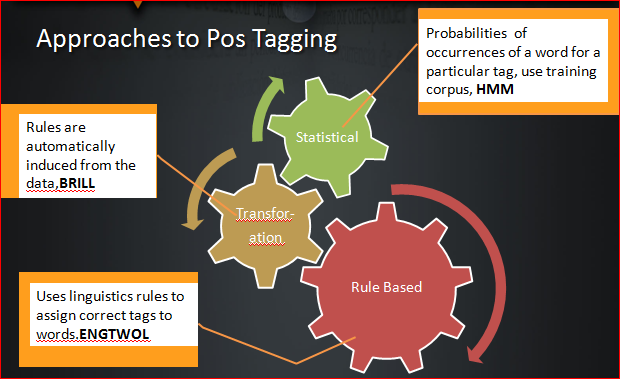
*Output:* *Tezpur/****NNP*** *University/****NNP*** *was/****VBD*** *established/****VBN*** *by/****IN*** *an/****DT*** *Act/****NNP*** *of/****IN*** *Parliament/****NNP*** *in****/IN*** *1994/****CD***

Here we see that *Tezpur* and *University* are NNP, or proper noun; *was* is VBM or main verb; *by* is IN, or a preposition; *an* is DT, or a determinant; 1994 is CD, a cardinal number and likewise.

* This task of assigning a syntactic category to each word in a text, thereby resolving some ambiguities. E.g., the tagger decides whether the word “owned” is used as a main verb in present tense or past tense verb.
* Part of Speech Tagger is an important tool that is used to develop language translator and information extraction. .
* Gives significance amount of information about words and its neighbours
* Can help in Speech Recognition eg- CONtent (N), conTENT (V).
* Stemming for IR(morphological affixes)
* Partial Parsing, quickly finding names or other phrases for IE application
* A variety of techniques have been used, e.g. statistical (Ratnaparkhi 1996, Brants 2000), memory based (Daelemans et al. 1996), rule-based (Brill 1992) and many more. The accuracies for small and medium sized tagsets are usually in the middle or high 90s.

There are 3 ways to POS tagging-

1. Statistical Based
2. Rule Based
3. Transformation Based



*Figure 1.2 Showing Different types of Tagging*

**1.2.3 Entity Detection using POS Tags**

Entity Detection or Named Entity Recognition is a key part of all information extraction systems as named entities are the main building bricks of relations and events. The classification of the entities is a more challenging problem than the simple recognition and it often needs information based on the environment of the token.

The recognition of named entities may be token or sequence-based. The token based approach assigns a label to each individual token independently of the labels of the other token.

CONLL 2003 defines the chunk borders of an entity by using IOB tags, where I-TYPE means that the word is inside an entity, B-TYPE means a beginning of a new entity if the previous token is part of an entity of the same type and O-TYPE for anything that is not part of an entity. For classification, the task defines four different types: Person(PER), Organization(ORG), Location(LOC) and Miscellaneous(MISC) .

*Input:**Tezpur University was established by an Act of Parliament in 1994*

*Output: [Tezpur]*I-LOC *[University]*B-LOC *[was]*O *[established]*O *[by]*O *[an]*O *[Act]*MISC *[ of]*B-MISC *[Parliament]*B-MISC *[in]*O *[1994]*O*.*

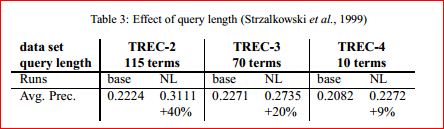
One of the basic techniques to detect entity is Chunking, that is segmenting tokens into phrases. This would be discussed in section 2.2.4. The rules that make up a chunk grammar use **tag patterns** to describe sequences of tagged words. A tag pattern is a sequence of part-of-speech tags delimited using angle brackets, e.g. <DT>?<JJ>\*<NN>. Tag patterns are similar to regular expression patterns.

**1.2.4 .Query Processing**

The next phase is to derive relation between neighboring tokens. This includes concept of Relationship Extraction. Looking for relations between specified types of named entity. One way of approaching this task is to initially look for all triples of the form (*X*, α, *Y*), where *X* and *Y* are named entities of the required types, and α is the string of words that intervenes between *X* and *Y*. Use of regular expressions to pull out just those instances of α that express the relation that we are looking for.

*Query: Search for strings that contain the word in.*

*Solution: The special regular expression (?!\b.+ing\b) is a negative lookahead assertion that allows us to disregard strings such as****and in the emerging areas in Science and Technology****, where in is followed by a gerund.*

**

**Query length** has an effect on the impact of natural language processing.(Strzalkowski et al. 1999b) found the correlation shown in table 3, which compares a baseline system with a system that uses various NLP techniques. In general, the impact of NLP is larger for longer queries. This seems to have a simple explanation: short queries lack a lot of the context information that is used in NLP. The same authors also confirmed this effect when focusing more on the individual components.

*[\*]The Text REtrieval Conference (TREC) is an on-going series of workshops focusing on a list of different information retrieval (IR) research areas, or tracks. It is co-sponsored by the National Institute of Standards and Technology (NIST) and the Intelligence Advanced Research Projects Activity*

**Chapter 2**

**Getting Deeper into Information Retrieval**

Information retrieval (IR) is the task of representing, storing, organizing, and offering access to information items. IR is different from data retrieval, which is about finding precise data in databases with a given structure.

But one might think that Information Retrieval (IR) is same as Information Extraction(IE). But then there lies a clear difference between them.**IR** is ***based on a query*** - you specify what information you need and it is returned in human understandable form. **IE** is about structuring unstructured information - given some sources ***all of the (relevant) information*** is structured in a form that will be easy for processing. This will not necessary be in human understandable form - it can be only for use of computer programs.

**2.1 Major** **challenges** while retrieving information or processing a query includes huge dynamic collections of diverse (multi-modal) material, users expectation of immediate answers, relevance ranking & personalization, access to everything everywhere (in various languages).

The basic issue is to find answers to these problems:-

* How do we match queries with documents?
* How do we measure relevance?
* What is a document?
* How can we make this really fast?
* How can we deal with huge diverse collections and millions of users?

However relevance can be found out by making use of Precision and Recall utility,discussed in section ***2.3.2*** exploring document contents, tagging, entity detection, finding links between document and query.

**2.2 Types of Information Retrieval**

1. Boolean Retrieval
2. Ranked Retrieval

**2.2.1 Boolean Retrieval**

The **Boolean model** of information retrieval (BIR) is a classical information retrieval (IR) model and, at the same time, the first and most adopted one.

The BIR is based on Boolean logic and classical set theory in that both the documents to be searched and the user's query are conceived as sets of terms. Retrieval is based on whether or not the documents contain the query terms. Given a finite set T of elements called index terms (e.g. words or expressions - which may be stemmed - describing or characterizing documents such as keywords given for a journal article), a finite set D , where Di is an element of the power set of T of elements called documents. Given a Boolean expression - in a normal form - Q called a query.

T = {

D = {},

Q = ( ...) ( ...),

With , or ­ ­

where means that the term is present in document , whereas means that it is not.

Equivalently, Q can be given in a disjunctive normal form, too. An operation called **retrieval**, consisting of two steps, is defined as follows:

1. The sets of documents are obtained that contain or not term (depending on whether or

) :

= {| element of }

2. Those documents are retrieved in response to Q which are the result of the corresponding sets operations,the answer to Q as follows:-

UNION ( )

The BIR model is cleanly formulated and is easy to implement and is intuitive concept. *Major disadvantage* are:-

* [Exact matching](http://en.wikipedia.org/wiki/String_search_algorithm) may retrieve too few or too many documents
* Difficult to rank output, some documents are more important than others
* Hard to translate a query into a Boolean expression
* All terms are equally weighted
* More like [data retrieval](http://en.wikipedia.org/wiki/Data_retrieval) than information retrieval

**2.2.2 Ranked Retrieval**

This is a scholastic approach to find relevance of information retrieved. Given below is a **tf-idf Model.** It is probably the best known weighting scheme in IR.

The Term Frequency (tf) is a measure of relevance and Inverse Document Frequency (idf) is ameasure of informativeness of the term.The given formula is

formula.JPG Where N is the no of terms in corpus.

Weight increases with:-

* number of occurrences within a document
* rarity of the term in collection

**2.3 Concepts, Tool and Terminology**

**2.3.1 Vector Space Model**

As Query language is expressive but complicated, therefore a model is needed that consider individual independent of each other. The model discussed in section 2.2.2 is a kind of vector space model only. Vector-space models were developed to eliminate many of the problems associated with exact, lexical matching techniques. The model assumes documents and queries both as vector.

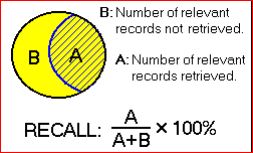
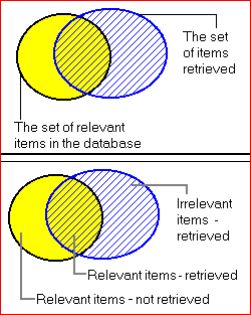
* Each dimension corresponds to a separate term.
* Both the document vectors and the query vector provide the locations of the objects in the term-document space.
* By computing the distance between the query and other objects in the space, objects with similar semantic content to the query presumably will be retrieved.
* If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed eg *tf-idf*.
* If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus).
* Vector operations can be used to compare documents with queries.

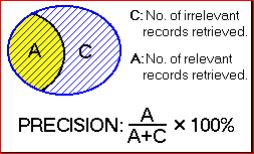
**2.3.2 Precision and Recall**

Precision and recall are the basic measures used in evaluating search strategies. As shown in the first figure on the left, these measures assume: There is a set of records in the database which is relevant to the search topic .Records are assumed to be either relevant or irrelevant (these measures do not allow for degrees of relevancy).The actual retrieval set may not perfectly match the set of relevant records.

**Recall** is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. It is usually expressed as a percentage.

**Precision** is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage.

**

**

*Figure 2.1 Using Set Theory to show Precision and Recall*

**2.3.3 Stopwords**

Almost all IR applications remove stopwords (function words, low-content words, very high frequency words) before processing documents and queries. This usually increases system performance. But there are many counter-examples that are handled poorly after stopword removal, e.g.:

1. To be or not to be
2. New Year celebrations
3. Will and Grace
4. On the road again

(Words in italics are considered stopwords). Adjusting the stopword list to the given task can significantly improve results (Farahat et al. 2003). Creating stopword lists is not generally considered to be NLP, but NLP techniques can help to create specific lists and to deal with examples 1 – 4 above.

**2.3.4 Stemming**

Stemming is the task of mapping words to some base form.The two main methods are

1. *Linguistic/dictionary-based stemming*- has higher stemming accuracy, but also higher implementation and processing costs and lower coverage,
2. *Porter-style stemming*-has lower accuracy, but also lower implementation and processing costs and is usually sufficient for IR.

Stemming maps several terms onto one base form, which is then used as a term in the vector space model. On average, it increases similarities between documents or documents and queries because they have an additional common term after stemming, but not before. *This results in an increase in recall, but sacrifices precision*. Stemming has a relatively low processing cost, especially when using Porter style stemming. It reduces the index size, and it usually slightly improves results.

**2.3.5 Compounds and Statistical Phrases**

Compounds and statistical phrases index multi token units instead of single tokens. The technique used in SMART (Buckley et al. 1993) is to collect pairs of adjacent non-stopwords and then use all pairs with a frequency above some threshold.

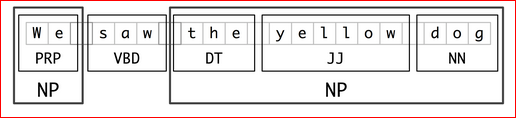
* It is possible to use longer n-grams, but this is expensive because of the large number of longer n-grams. Bigrams already significantly increase the index size, even when pruning by frequency.
* Single tokens alone match documents that should not match (e.g. matching *Tezpur* in *Tezpur University*). Using multi-token units alone adds a very high penalty for slight variations, e.g. documents containing *Mihir K. Choudhry* suddenly would not match anymore when the query is *Mihir Choudhry*.
* Adding both single-token units and multi-token units to the document vector alleviates these problems. The net benefit of using compounds is positive, and it is likely to be further improved if the cases above can be separated automatically.
* Making these distinctions is related to determining compositionality of compounds, but it is not the same. Processing needs to be tailored to the retrieval task in order to identify those compounds that are improve retrieval accuracy.

**2.3.6 Compound Splitting**

Many languages like Dutch, German, Swedish, and many more, form words by *concatenating* other words in a productive process. Being able to separate compounds should improve retrieval quality. A simple algorithm for compound splitting is to consider all other words found in the lexicon as possible parts. Optionally, one can require a minimum length of parts (e.g. length ≥ 4), allow linking elements (e.g. *-e-, -en-, -n-, -s-* in German), and require that the frequency of each part is larger than the frequency of the compound. The net benefit of compound splitting is usually positive. However, it is a research question to automatically determine which split is beneficial for retrieval.

**2.3.7 Chunking and Shallow Parsing**

Chunking and Shallow Parsing aim at separating words in a sentence into basic phrases, e.g. noun phrases or simple verb phrases. A large number of technique has been tried. The best system in the CoNLL 2000 shared task evaluation for chunking. Chunks are used in the vector space model the same way as n-grams or compounds: both the individual terms as well as the whole chunk are added as separate dimensions to the vector.



To find the chunk structure for a given sentence, the *RegexpParser* chunker begins with a flat structure in which no tokens are chunked. The chunking rules are applied in turn, successively updating the chunk structure. Once all of the rules have been invoked, the resulting chunk structure is returned.

**3.3.8 Co-reference Resolution**

Coreference resolution is the task of finding all expressions that refer to the same entity in a text.

* Not all discourse entities are created equal. Some lead long lives and appear in a variety of discourse contexts (**coreferent**), whereas others never escape their birthplaces, dying out after just one mention (singletons).
* The ability to make this distinction based on properties of the NPs used to identify these referents (mentions) would benefit not only coreference resolution, but also topic analysis, textual entailment, and discourse coherence document summarization, question answering, and information extraction.
* When exploring coreference, there are numerous distinctions that can be made, e.g. anaphora, cataphora, split antecedents, coreferring noun phrases, etc *Input: The* ***musici****was so loud that****iti****couldn't be enjoyed.*

The **anaphor** *it* follows the expression to which it refers (its **antecedent**).

**Chapter 3**

**Implementation**

Here we present a starting implementation that will constitute downloading Web Page and separating text and table. Then removing HTML tags, converting it to corpus. The next step involves POS tagging on corpus. Next getting query as input and POS tagging it too. The idea is to find similar structure in corpus also. Finally retrieving best possible answer to query at this stage and developing an interface for Guest User.

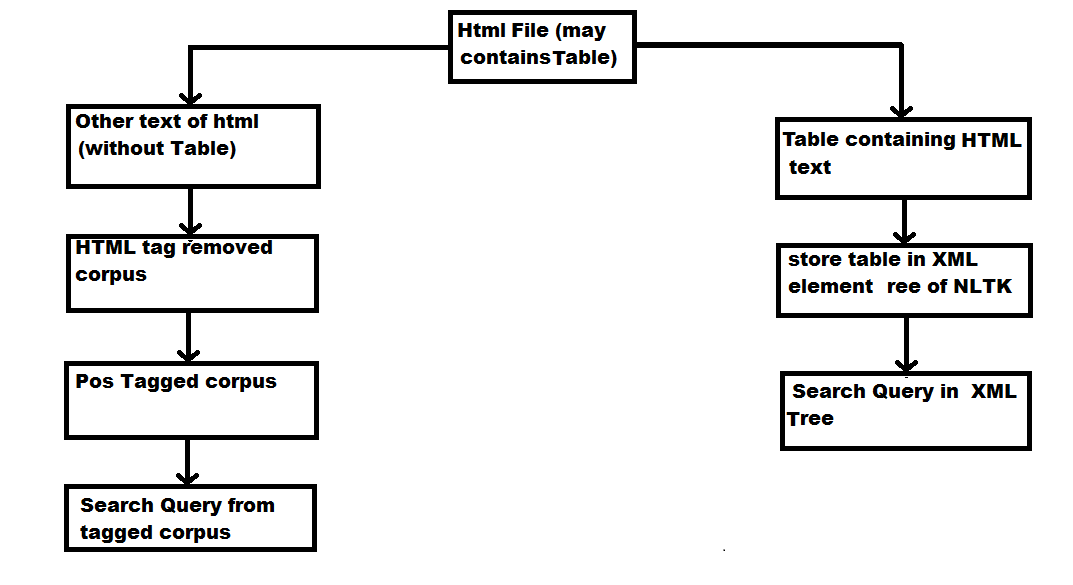
**3.1 Steps involved in implementation**

*Following steps assumes that NLTK, Python and PHP has been installed and configured on user’s work machine.*

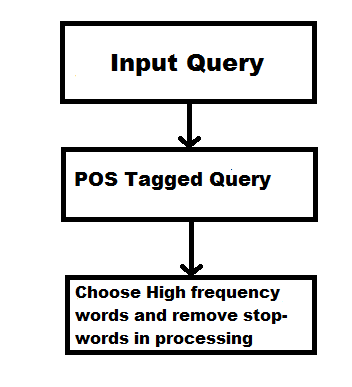
1. First step is downloading Web Page. Such as we choose [www.tezu.ernet.in](http://www.tezu.ernet.in) as input.
2. We separate the web page into two parts or files. First contains all tables and the left is stored in second file as running text.
3. First part containing table is stored in XML-Element Tree of NLTK and query processing is done by traversing the XML Tree.
4. Next second file consisting of running text is kept as it is. A query in taken as input to information retrieval from User Interface developed using css and PHP.
5. Here the query is assumed to be type of “*Who is …….”* for running text or “*Detail of…..”* for getting information from table.
6. POS tagging of Query is performed using **Maxent-Treee Bank POS tagger** .It has maximum entropy.
7. Next step involves removal of Stopwords from query. This reduces search time in IR.
8. According to POS tagging done for query,
   1. For query like “who is …”

* POS tagging is done on corpus for running text in second file,, which involves tagging token sentence wise, i .e differentiating sentence too.
* Once the sentence contacting the maximum no of exactly same words from query are acknowledged, that particular sentence is given as output, considering it to be possible answer to query.
  1. For query – “details of……”
* The term after detail of is taken as key and is searched in the tree. Matching node is shown as output along with its following nodes in its branch.

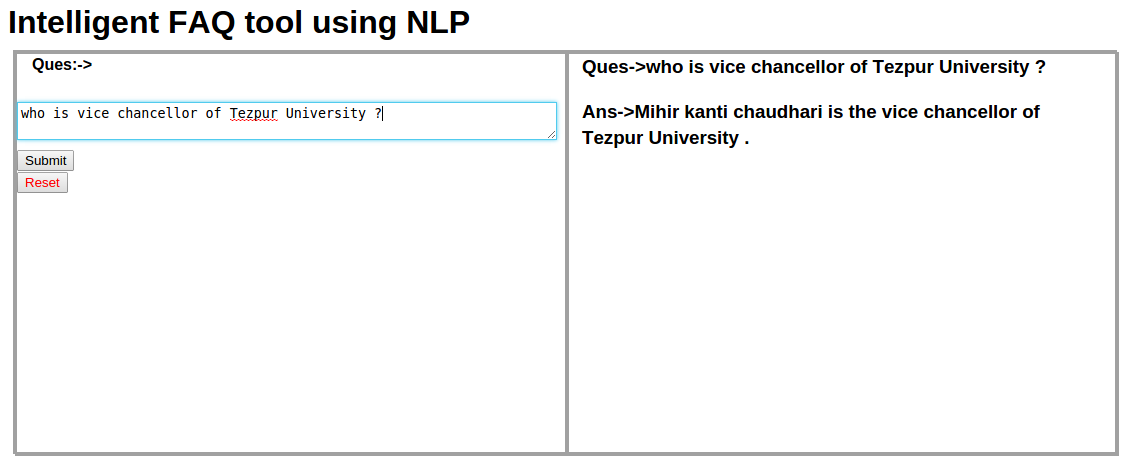
**3.2 Corpus Processing ( Flow chart)**

****

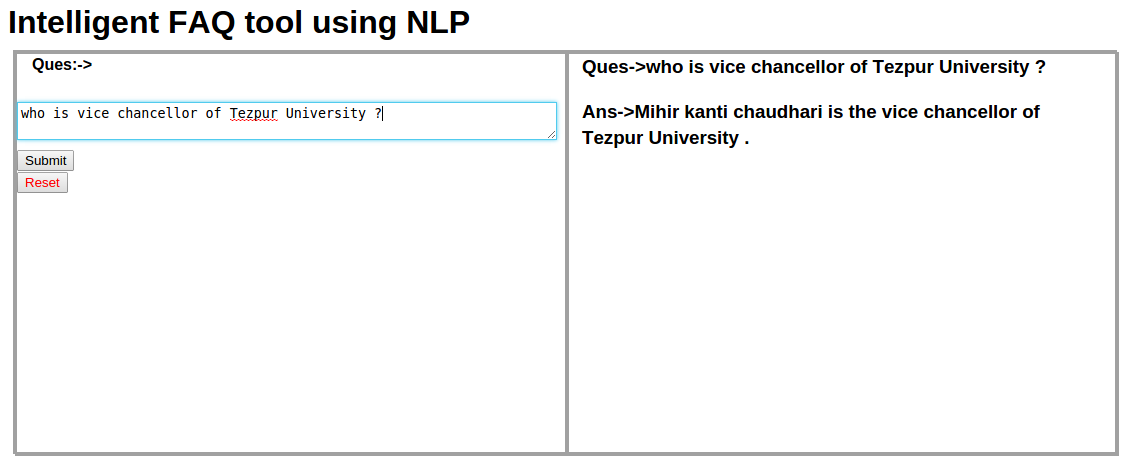
**3.2 Query Processing (Flow chart)**

****

**3.2 Snapshots of result**

****

**Fig 3.1: Snapshots during information retrieval from corpus**

****

**Fig 3.1: Snapshots during information retrieval from HTML Table**

**Conclusion and Future Findings**

In this initial phase of project we have been able to achieved fair knowledge of NLP and particularly IR. Our research on our project area is still continuing. Though, from the knowledge acquired till now we have developed a demo of our project. It gives answer for Wh-question, of type, *Who*, only and give details of student when searched by name . The detail of students is extracted by converting HTML table into a running text of corpus at basic level. In this way we are able to retrieve information from a known type corpus and so the query asked are expected to be of similar nature and assumed to be coming from a set of defined question types.

Following implementation are needed to be done in next phase of project

* The FAQ tool is expected to give answers to query for other Wh Question like- What, When etc.
* The corpus can be dynamically upgraded.
* Improving efficiency of Query searching while retrieving information from table and for query of question types, making query not case sensitive i.e. answer for a query if present should be independent of case sensitive nature of query and corpus.
* Improving query searching by Scoring Term, Co-reference Resolution, Working on Query Length.
* Further concepts by using ongoing research on topics, features on tool may be added.

**References**

Palmer David D., *Handbook of natural language processing ,Chapter 2,Tokenisation and Sentence Segmentation*, Year-2000.

Bird S., Klein E., and Loper E*., Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit, Chapter-07 ,Extracting Information from Text*, First ed.

Conference on Computational Natural Language Learning (CoNLL-2003), *Language-Independent Named Entity Recognition (II).*

Oliver Bender, Franz Josef Och and Hermann Ney, *Maximum Entropy Models for Named Entity Recognition* In: *Proceedings of CoNLL-2003*, Edmonton, Canada, 2003, pp 148-151.

Erik F. Tjong Kim Sang and Fien De Meulder, *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. In: *Proceedings of CoNLL-2003*, Edmonton, Canada, 2003, pp. 142-147.

Jörg Tiedemann*, Information Retrieval*, *Information Extraction,Question Answering*, slide 4-17,Department of Linguistics and Philology, Uppsala University.

Todd A. Letsche, Michael W. Berry, *Large-Scale Information Retrieval with Latent Semantic Indexing, Chapter-02. Vector-Space Models for Information Retrieval*, Submitted to Information Sciences - Applications, 1996.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. *The Life and Death of Discourse Entities: Identifying Singleton Mentions*. In Proceedings of NAACL 2013.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval, Chapter 05, Dropping common terms: stop words* ,Cambridge University Press. 2008.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, Christopher Manning. 2010. A *Multi-Pass Sieve for Coreference Resolution* EMNLP-2010, Boston, USA. 2010.

Rami Al − Rf , Steven Skiena, SpeedRead: *A Fast Named Entity Recognition Pipeline*,In *Proceedings of COLING 2012: Technical Papers*, pages 51–66,COLING 2012, Mumbai, December 2012 pp 52-29.

György Móra, Veronika Vincze, *Joint Part-of-Speech Tagging and Named Entity-Recognition Using Factor Graphs*,pp 233-236.

**Wiki Links**

* *en.wikipedia.org/wiki/Standard\_Boolean\_model*
* [*https://www.creighton.edu/fileadmin/user/HSL/docs/ref/Searching\_-\_Recall\_Precision.pdf*](https://www.creighton.edu/fileadmin/user/HSL/docs/ref/Searching_-_Recall_Precision.pdf)
* [*http://en.wikipedia.org/wiki/Natural\_language\_processing*](http://en.wikipedia.org/wiki/Natural_language_processing)
* *http://en.wikipedia.org/wiki/Information\_Retrieval*

**Appendix**

1. **Installing NLTK, Python and PHP**

* NLTK requires Python versions 2.6-2.7 or 3.2+

For Mac/Unix

1. Install Setuptools: http://pypi.python.org/pypi/setuptools
2. Install Pip: run sudo easy\_install pip
3. Install Numpy (optional): run sudo pip install -U numpy
4. Install NLTK: run sudo pip install -U nltk
5. Test installation: run python then type import nltk

* PHP

From a command shell, run the following commands:

sudo apt-get install apache2

sudo apt-get install php5

sudo apt-get install libapache2-mod-php5

sudo /etc/init.d/apache2 restart

1. **Configuring NLTK with required Corpus**

For central installation on a multi-user machine, do the following from an administrator account.

1.Run the Python interpreter and type the commands:

import nltk

nltk.download()

2.A new window should open, showing the NLTK Downloader.Click on the File menu and select Change Download Directory. For central installation, set this to

/usr/share/nltk\_data

3.Select the packages or collections needed to download.

` 4.Test that the data has been installed as follows.(This assumes you downloaded the Brown Corpus):

from nltk.corpus import brown

brown.words()