NATURAL LANGUAGE PROCESSING: PART OF SPEECH TAGGER - ENGLISH

A PROJECT REPORT

Submitted to

SRI VENKATESWARA UNIVERSITY

In partial fulfilment of the

Requirements for the award of the Degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

MADURU SADAK PRAMODH

(Roll No: 1115608)

Under the guidance of

Dr.M.HUMERA KHANAM

Associate Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA UNIVERSITY

TIRUPATI-517502(A.P.), INDIA

2015 - 2017.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARAUNIVERSITYCOLLEGE OF ENGINEERING
SRIVENKATESWARAUNIVERSITY
TIRUPATI-517502(A.P.), INDIA

Certificate

This is to certify that the dissertation entitled "NATURAL LANGUAGE PROCESSING: PART OF SPEECH TAGGER - ENGLISH " is a bonafide work carried out by Mr. MADURU SADAK PRAMODH (Roll No: 1115608)in partial fulfilment of requirements for the award of the degree of MASTER OF TECHNOLOGY with the specialization in COMPUTER SCIENCE & ENGINEERING, during the academic year 2015-2017.

GUIDE

Dr.M.HUMERA KHANAM
Associate Professor
Department of CSE
SVU College of Engineering
Sri Venkateswara University
Tirupati-517502.

HOD:

Dr. M.HUMERA KHANAM
Head of the Department
Department of CSE
SVU College of Engineering
Sri Venkateswara University
Tirupati-517502.

DECLARATION

"I, MADURU SADAK PRAMODH", declare that the M.TECH dissertation work entitled **"NATURAL LANGUAGE PROCESSING: PART OF SPEECH TAGGER - ENGLISH"** submitted to Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering, Sri Venkateswara University, Tirupati in partial fulfilment of requirements for the award of the degree of **MASTER OF TECHNOLOGY.** This thesis contains no material that has been submitted previously, in whole or in part, for the award any other academic degree or diploma. Except where otherwise indicated, this dissertation work is my own work".

MADURU SADAK PRAMODH

1115608

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor **Dr.M.HUMERA KHANAM**, Associate Professor, Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering, SVU, Tirupati, for her immense help, guidance, stimulating suggestions and encouragement all the time with this dissertation work. She always provided me motivating and enthusiastic atmosphere to work with, it was a great pleasure to do this thesis under her supervision.

I am deeply obliged to **Dr.M.HUMERA KHANAM**, Head, Department of Computer Science and Engineering, S V University College of Engineering, for her constant help, encouragement and timely advice.

I acknowledge my sincere thanks to **Prof.G.PADMANABHAN,** Principal of S.V. University College of Engineering, SVU, Tirupati, for his kind cooperation in making this dissertation work a success.

I acknowledge my special thanks to all the faculty members of Department of Computer Science and Engineering, SVUCE, SVU, Tirupati, for their kind cooperation in making this dissertation work a success.

A project work of this magnitude is not possible without the help of several people, directly or indirectly. I take this as opportunity to thank all those magnanimous persons who rendered their full support to my work.

I would like to express my thanks to my parents, **M SUJATHA and M SESHAIAH**, and brother **M SIMEON PRASANTH** for their continued support & encouragement through the ups & downs encountered on the path to completing this dissertation work.

Finally, I am very thankful to my friend **PATRICK SCHUR** who helped a lot while conducting the experiments.

MADURU SADAK PRAMODH

Roll No.1115608

ABSTRACT

Part-of-speech tagging is the process of associating each word in a text with its part-of-speech category and possibly a set of morphsyntactic features. This information is represented by part-of-speech tags. We describe an implementation of a part-of-speech tagger for English based on the Brill method. The basic idea is to apply a set of rules to an initial annotation achieved using a simple algorithm. The rules are found using transformation-based learning applied to a manually tagged training corpus and also implemented built-in Tokenizer that allows one to feed raw English text directly into the tagger. This eliminates the need for pre-tokenization of English text. Initialization method for unknown numerals has been added. It ensures unknown cardinal numbers are tagged by the tag "CD" in the start-state-tagger sequence. Brill tagger's original lexicon has been replaced by a lemmatized lexicon. This allows Lemmatizer to print lemma information for each token. It also addresses the problem of tagging unknown words, i.e. words that don't appear in the training corpus. The text language detection based on ISO639 char set and returns confidence scores of each and compares the result of POS of Stanford library results. Our method guaranteed pre tagging is suitable when the tag of word is known for certain and is proposed to help improve the accuracy of classification by providing a reliable secure start around which to tag and language detection.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

TERM	DESCRIPTION	TERM	DESCRIPTION
MD	modal auxiliary	**NN**	noun, common, singular or mass
NLP	Natural Language Processing	**NNP**	noun, proper, singular
POS	Part of Speech	**NNPS**	noun, proper, plural
POST	Part of Speech Tagger	**NNS**	noun, common, plural
HMM	Hidden Markov Model	**PDT**	pre-determiner
AI	Artificial Intelligence	**POS**	genitive marker
WSD	Word Sense Disambiguation	**PRP**	pronoun, personal
IR	Information Retrieval	**PRP$**	pronoun, possessive
UML	Unified Modeling Language	**RB**	Adverb
:	colon or ellipsis	**RBR**	adverb, comparative
``	opening quotation mark	**RBS**	adverb, superlative
''	closing quotation mark	**RP**	Particle
(	opening parenthesis	**SYM**	Symbol
)	closing parenthesis	**TO**	"to" as preposition or infinitive marker
,	Comma	**UH**	Interjection
--	Dash	**VB**	verb, base form
.	sentence terminator	**VBD**	verb, past tense
$	Dollar	**VBG**	verb, present participle or gerund
CC	conjunction, coordinating	**VBN**	verb, past participle
CD	numeral, cardinal	**VBP**	verb, present tense, not 3rd person singular
DT	determiner	**VBZ**	verb, present tense, 3rd person singular
EX	existential there	**WDT**	WH-determiner
FW	foreign word	**WP**	WH-pronoun
IN	preposition or subordinating conjunction	**WP$**	WH-pronoun, possessive
JJ	adjective or numeral, ordinal	**WRB**	Wh-adverb
JJR	adjective, comparative	**JJS**	adjective, superlative

CHAPTER 1
INTRODUCTION

INTRODUCTION

1.1 Natural Language Processing

Natural Language Processing (NLP) refers to the use and ability of systems to process sentences in a natural language such as English, Telugu rather than in a specialized artificial computer language such as C, C++, java etc.; The development of NLP applications is challenging because computers traditionally require humans to speak to them in computer programming language that is precise, unambiguous and highly structured commands. Human language, however, is not always precise. It is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context. The ultimate goal of NLP is to do away with computer programming languages altogether. Instead of specialized languages such as Java or Ruby or C, there would only be "human".

1.2 History

The first use of computers to manipulate natural languages was in the 1950s with attempts to automate translation between Russian and English [Locke & Booth]. These systems were spectacularly unsuccessful requiring human Russian-English translators to pre-edit the Russian and post-edit the English. Based on World War II code breaking techniques, they took individual words in isolation and checked their definition in a dictionary. They were of little practical use. Popular tales about these systems cite many mis-translations including the phrase "hydraulic ram" translated as "water goat".

In the 1960s NLP systems started to examine sentence structure but often in an ad hoc manner. These systems were based on pattern matching and few derived representations of meaning. The most well-known of these is Eliza [Weisenbaum] though this system was not the most impressive in terms of its ability to extract meaning from language.

Serious developments in NLP took place in the early & mid 1970s as systems started to use more general approaches and attempt to formally describe the rules of the language they worked with. LUNAR [Woods 1973] provided an English interfaceto a database holding details of moon rock samples. SHRDLU [Winograd] interfaced with a virtual robot in a

world of blocks, accepting English commands to move the blocks around and answer questions about the state of the world. Since that time there has been parallel development of ideas and technologies that provide the basis for modern natural language processing systems. Research in computer linguistics has provided greater knowledge of grammar construction [GazdarJ and Artificial Intelligence researchers have produced more effective mechanisms for parsing natural languages and for representing meanings [Allen]. NLP systems now build on a solid base of linguistic study and use highly developed semantic representations.

Up to the 1980s, most NLP systems were based on complex sets of handwritten rules. Starting in the late 1980s, however, there was a revolution in NLP with the introduction of machine learning algorithms for language processing. This was due both to the steady increase in computational power resulting from Moore's Law and the gradual lessening of the dominance of Chomsky theories of linguistics (e.g. transformational grammar), whose theoretical underpinnings discouraged the sort of corpus linguistics that underlies the machine-learning approach to language processing. Some of the earliest-used machine learning algorithms, such as decision trees, produced systems of hard if-then rules similar to existing hand-written rules. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to the features making up the input data. Such models are generally more robust when given unfamiliar input, especially input that contains errors (as is very common for real-world data), and produce more reliable results when integrated into a larger system comprising multiple subtasks.

Many of the notable early successes occurred in the field of machine translation, due especially to work at IBM Research, where successively more complicated statistical models were developed. These systems were able to take advantage of existing multilingual textual corpora that had been produced by the Parliament of Canada and the European Union as a result of laws calling for the translation of all governmental proceedings into all official languages of the corresponding systems of government. However, most other systems depended on corpora specifically developed for the tasks (and often continues to be) a major limitation in the success of these systems. As a result, a great deal of research has gone into methods of more effectively learning from limited amounts of data.

Recent research has increasingly focused on unsupervised and semi supervised learning algorithms. Such algorithms are able to learn from data that has not been hand-annotated with the desired answers, or using a combination of annotated and non-annotated data. Generally, this task is much more difficult than supervised learning, and typically produces less accurate results for a given amount of input data. However, there is an enormous amount of non-annotated data available (including, among other things, the entire content of the World Wide Web), which can often make up for the inferior results.

1.3 Major tasks in NLP

The following is a list of some of the most commonly researched tasks in NLP. Some of these tasks have directly real-world applications, while others more commonly serve as subtasks that are used to aid in solving large tasks.

- **Anaphora (Linguistics):** It is the use of an expression the interpretation of which depends upon another expression in context (it's antecedent or postcedent).
- **Automatic summarization:** Produce a readable summary of a chunk of text. Often used to provide summaries of text of a known type, such as articles in the financial section of a newspaper.
- **Collocation Extraction:** It is the task of extracting collocations automatically from a corpus using a computer. Collocation is defined as a sequence of words or terms which co-occur more often than would be expected by chance. 'crystal clear', 'middle management', 'nuclear family', 'riding boots', 'cosmetic surgery', 'motor cyclist' are some of the examples of collocated pair of words.
- **Conference resolution:** Given a sentence or larger chunk of text, determine which words ("mentions") refer to the same objects ("entities").
- **Discourse analysis:** This rubric includes a number of related tasks. One task is identifying the discourse structure of connected text, i.e., the nature of the discourse relationships between sentences (Ex: elaboration, explanation, contrast). Another possible task is recognizing and classifying the speech acts in a chunk of text (Ex: yes-no question, content question, statement, assertion, etc.,).

- **Machine Translation:** Automatically translate text from one human language to another. This is one of the most difficult problems, and is a member of a class of problems colloquially termed "Al-complete", i.e., requiring all of the different types of knowledge that humans possess (grammar, semantics, facts about the real world, etc.,) in order to solve properly.
- **Morphological segmentation:** Separate words into individual morphemes and identify the class of the morphemes. The difficulty of this task depends greatly on the complexity of the morphology (i.e., the structure of words) of the language being considered. English has fairly simple morphology, especially inflectional morphology, and thus it is often possible to ignore this task entirely and simply model all possible forms of a word (Ex: "open, opens, opened, opening") as separate words. In languages such as Turkish, however, such an approach is not possible, as each dictionary entry has thousands of possible word forms.
- **Named Entity Recognition:** It is a task to discover the Named Entities in a document and then categorize these NES into diverse Named Entity classes such as Name of Person, Location, River, Organization etc.,
- **Natural Language Generation:** Convert information from computer databases into readable human language.
- **Natural Language Understanding:** It is a subtopic of NLP in artificial intelligence that deals with machine reading comprehension. It converts chunks of text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate. Natural Language Understanding involves the identification of the intended semantic from the multiple possible semantics which can be derived from a natural language expression which usually takes the form of organized notations of natural language concepts. Introduction and creation of language metamodeland ontology are efficient however empirical solutions. An explicit formalization of natural languages semantics without confusions with implicit assumptions such as closed world assumption vs. open world assumption or subjective yes/no vs. objective true/false is expected for the construction of a basis of semantics formalization.

- **Optical Character Recognition (OCR):** Given an image representing printed text, determine the corresponding text.
- **Part-of-speech (POS) tagging:** In corpus linguistics, POS tagging also called as grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context i.e., its relationship with adjacent and related words in a phrase, sentence or a paragraph.
- **Parsing:** It is also called as syntactic analysis, is the process of analyzing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar and generate a parse tree for a given sentence.
- **Question Answering:** Given a human-language question, determine its answer. Typical questions have a specific right answer (such as "What is the capital of Canada?"), but sometimes open-ended questions are also considered (such as "What is the meaning of life?").
- **Relationship Extraction:** Given a chunk of text, identify the relationships among NES (Ex: who is the wife of whom?).
- **Sentence Breaking:** It is also known as "Sentence boundary disambiguation". On giving a chunk of text, find the sentence boundaries. Sentence boundaries are often marked by periods or other punctuation marks.
- **Sentiment Analysis:** Extract subjective information usually from a set of documents, often using online reviews to determine "polarity" about specific objects. It is especially useful for identifying trends of public opinion in the social media, for the purpose of marketing.
- **Speech Recognition:** When a sound clip of a person or people speaking is given as input, then Speech Recognizer analyze the input and need to give corresponding text as output.
- **Speech Synthesis:** When a text is given as input to speech Synthesizer, it need to analytic the and give corresponding speech as output.

- **Speech Segmentation:** Given a sound clip of a text people separate it into A of Speech Recognition and typically grouped With it,
- **Topic Segmentation and Recognition:** Given chunk of tent, seperate it into segments of which is denoted to a topic. and identify the topic Of the segment.
- **Word Segmentation:** Separate a chunk of continuous text into scratatc words. For a language like English, this is fairly trival, since words arc usually separated by spaces. However, some written languages like Chinese. Japanese and Thai do not mark word boundaries in such a fashion, and In those languages text segmentation is a significant task requiring knowledge of the vocabulary and morphology of words in the language.
- **Word Sense Disambiguation (WSD):**Many words have more than one meaning; we have to select the meaning which makes the most sense in context. For this problem, we are typically given a list of words and associated word senses, Ex: from a dictionary or from an online resource such as WordNet.

In some cases, sets of related tasks are grouped into subfields of NLP that are often considered separately from NLP as a whole. Examples include:

- **Information Retrieval (IR):** This is concerned with storing, searching and retrieving information. It is a separate field within computer science (closer to databases), but IR relies on some NLP methods (for example, stemming). Some current research and applications seek to bridge the gap between IR and NLP.
- **Information Extraction (IE):** This is concerned in general with the extraction of semantic information from text. This covers tasks such as NER, Conference resolution, Relationship Extraction, etc.,

1.4 Problems

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken. This is not rare—in natural

languages (as opposed to many artificial languages), a large percentage of word-forms are ambiguous. For example, even "dogs", which is usually thought of as just a plural noun, can also be a verb:

The sailor dogs the hatch.

Correct grammatical tagging will reflect that "dogs" is here used as a verb, not as the more common plural noun. Grammatical context is one way to determine this; semantic analysis can also be used to infer that "sailor" and "hatch" implicate "dogs" as 1) in the nautical context and 2) an action applied to the object "hatch" (in this context, "dogs" is a nautical term meaning "fastens (a watertight door) securely").

Schools commonly teach that there are 9 parts of speech in English: noun, verb, article, adjective, preposition, pronoun, adverb, conjunction, and interjection. However, there are clearly many more categories and sub-categories. For nouns, the plural, possessive, and singular forms can be distinguished. In many languages words are also marked for their "case" (role as subject, object, etc.), grammatical gender, and so on; while verbs are marked for tense, aspect, and other things. Linguists distinguish parts of speech to various fine degrees, reflecting a chosen "tagging system".

In part-of-speech tagging by computer, it is typical to distinguish from 50 to 150 separate parts of speech for English. For example, NN for singular common nouns, NNS for plural common nouns, NP for singular proper nouns (see the POS tags used in the Brown Corpus). Work on stochastic methods for tagging Koine Greek (DeRose 1990) has used over 1,000 parts of speech, and found that about as many words were ambiguous there as in English. A morphosyntactic descriptor in the case of morphologically rich languages is commonly expressed using very short mnemonics, such as 'Ncmsan for Category=Noun, Type = common, Gender = masculine, Number = singular, Case = accusative, Animate = no.

Correct grammatical tagging will reflect that "dogs" is here used as a verb, not as the more common plural noun. Grammatical context is one way to determine this; semantic analysis can also be used to infer that "sailor" and "hatch" implicate "dogs" as 1) in the nautical context and 2) an action applied to the object "hatch" (in this context, "dogs" is a nautical term meaning "fastens (a watertight door) securely").

While there is broad agreement about basic categories, a number of edge cases make it difficult to settle on a single "correct" set of tags, even in a single language such as English. For example, it is hard to say whether "fire" is an adjective or a noun in

the big green fire truck

A second important example is the use/mention distinction, as in the following example, where "blue" could be replaced by a word from any POS (the Brown Corpus tag set appends the suffix "-NC" in such cases):

the word "blue" has 4 letters.

Words in a language other than that of the "main" text are commonly tagged as "foreign", usually in addition to a tag for the role the foreign word is actually playing in context.

There are also many cases where POS categories and "words" do not map one to one, for example:

David's
gonna
don't
vice versa
first-cut
cannot
pre- and post-secondary
look (a word) up

In the last example, "look" and "up" arguably function as a single verbal unit, despite the possibility of other words coming between them. Some tag sets (such as Penn) break hyphenated words, contractions, and possessives into separate tokens, thus avoiding some but far from all such problems.

It is unclear whether it is best to treat words such as "be", "have", and "do" as categories in their own right (as in the Brown Corpus), or as simply verbs (as in the LOB

Corpus and the Penn Treebank). "be" has more forms than other English verbs, and occurs in quite different grammatical contexts, complicating the issue.

The most popular "tag set" for POS tagging for American English is probably the Penn tag set, developed in the Penn Treebank project. It is largely similar to the earlier Brown Corpus and LOB Corpus tag sets, though much smaller. In Europe, tag sets from the Eagles Guidelines see wide use, and include versions for multiple languages.

POS tagging work has been done in a variety of languages, and the set of POS tags used varies greatly with language. Tags usually are designed to include overt morphological distinctions, although this leads to inconsistencies such as case-marking for pronouns but not nouns in English, and much larger cross-language differences. The tag sets for heavily inflected languages such as Greek and Latin can be very large; tagging words in agglutinative languages such as Inuit may be virtually impossible. At the other extreme, Petrov et al. have proposed a "universal" tag set, with 12 categories (for example, no subtypes of nouns, verbs, punctuation, etc.; no distinction of "to" as an infinitive marker vs. preposition, etc.). Whether a very small set of very broad tags or a much larger set of more precise ones is preferable, depends on the purpose at hand. Automatic tagging is easier on smaller tag-sets.

A different issue is that some cases are in fact ambiguous. Beatrice Santorini gives examples in "Part-of-speech Tagging Guidelines for the Penn Treebank Project", including the following, case in which entertaining can be either an adjective or a verb, and there is no syntactic way to decide:

The Duchess was entertaining last night.

1.5 Part-of-Speech Tagging

Part-of-speech tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Once performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags. POS-tagging algorithms fall into two distinctive groups: rule-based and stochastic. E. Brill's tagger, one of the first and most widely used English POS-taggers, employs rule-based algorithms.

1.6 Motivation

Part of Speech (POS) is an essential technology for a number of propelled majorities of the data management applications, such as search engines, question answering systems, text mining and business intelligence. All of these applications can profit from exact POS such as search engines and question answering systems can be improved by permitting searches and inquires for particular persons, organizations or locations. In text mining, exact POS will permit the development of databases with information wrenching out regarding specific entities. Multilingual POS applications are cross-language information retrieval, and business intelligence applications, where information about a specific person or organization has to be wrenched from textual sources in different languages.

This dissertation work is organized as follows. Literature survey for NLP: POS for English is presented in chapter 2. A detailed description NLP: POS for English is provided in chapter 3. Problem statement and proposed work is presented in chapter 4. Algorithm and System design is presented in chapter 5. Experimental results and performance metrics are presented in chapter 6.

CHAPTER 2
LITERATURE SURVEY

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

2.1 History of POS

Part-of-Speech (POS) tagging is the process of labeling a Part-of-Speech or other lexical class marker to each and every word in a sentence. It is similar to the process of tokenization for computer languages. Hence POS tagging is considered as an important process in speech recognition, natural language parsing, morphological parsing, information retrieval and machine translation.

Research on part-of-speech tagging has been closely tied to corpus linguistics. The first major corpus of English for computer analysis was the Brown Corpus developed at Brown University by Henry Kučera and W. Nelson Francis, in the mid-1960s. It consists of about 1,000,000 words of running English prose text, made up of 500 samples from randomly chosen publications. Each sample is 2,000 or more words (ending at the first sentence-end after 2,000 words, so that the corpus contains only complete sentences).

The Brown Corpus was painstakingly "tagged" with part-of-speech markers over many years. A first approximation was done with a program by Greene and Rubin, which consisted of a huge handmade list of what categories could co-occur at all. For example, article then noun can occur, but article verb (arguably) cannot. The program got about 70% correct. Its results were repeatedly reviewed and corrected by hand, and later users sent in errata, so that by the late 70s the tagging was nearly perfect (allowing for some cases on which even human speakers might not agree).

This corpus has been used for innumerable studies of word-frequency and of part-of-speech, and inspired the development of similar "tagged" corpora in many other languages. Statistics derived by analyzing it formed the basis for most later part-of-speech tagging

systems, such as CLAWS (linguistics) and VOLSUNGA. However, by this time (2005) it has been superseded by larger corpora such as the 100 million word British National Corpus.

For some time, part-of-speech tagging was considered an inseparable part of natural language processing, because there are certain cases where the correct part of speech cannot be decided without understanding the semantics or even the pragmatics of the context. This is extremely expensive, especially because analyzing the higher levels is much harder when multiple part-of-speech possibilities must be considered for each word.

2.2 Approaches for POS

Different approaches have been used for Part-of-Speech (POS) tagging, where the notable ones are rule-based, stochastic, or transformation-based learning approaches. Rule-based taggers try to assign a tag to each word using a set of handwritten rules. These rules could specify, for instance, that a word following a determiner and an adjective must be a noun. This means that the set of rules must be properly written and checked by human experts. The stochastic (probabilistic) approach uses a training corpus to pick the most probable tag for a word. All probabilistic methods cited above are based on first order or second order Markov Models.

There are a few other techniques which use probabilistic approach for POS Tagging, such as the Tree Tagger. Finally, the transformation-based approach combines the rule-based approach and statistical approach. It picks the most likely tag based on a training corpus and then applies a certain set of rules to see whether the tag should be changed to anything else. It saves any new rules that it has learnt in the process, for future use. One example of an effective tagger in this category is the Brill tagger. All of the approaches discussed above fall under the rubric of supervised POS Tagging, where a pre tagged corpus is a prerequisite. On the other hand, there is the unsupervised POS tagging technique and it does not require any pre-tagged corpora. Koskenniemi also used a rule-based approach implemented with finite-state machines.

2.2.1 Hidden Markov Models

In the mid-1980s, researchers in Europe began to use hidden Markov models (HMMs) to disambiguate parts of speech, when working to tag the Lancaster-Oslo-Bergen Corpus of British English. HMMs involve counting cases (such as from the Brown Corpus), and making a table of the probabilities of certain sequences. For example, once you've seen

an article such as 'the', perhaps the next word is a noun 40% of the time, an adjective 40%, and a number 20%. Knowing this, a program can decide that "can" in "the can" is far more likely to be a noun than a verb or a modal. The same method can of course be used to benefit from knowledge about following words.

More advanced ("higher order") HMMs learn the probabilities not only of pairs, but triples or even larger sequences. So, for example, if you've just seen a noun followed by a verb, the next item may be very likely a preposition, article, or noun, but much less likely another verb.

When several ambiguous words occur together, the possibilities multiply. However, it is easy to enumerate every combination and to assign a relative probability to each one, by multiplying together the probabilities of each choice in turn. The combination with highest probability is then chosen. The European group developed CLAWS, a tagging program that did exactly this, and achieved accuracy in the 93–95% range.

It is worth remembering, as Eugene Charniak points out in Statistical techniques for natural language parsing (1997),[1] that merely assigning the most common tag to each known word and the tag "proper noun" to all unknowns will approach 90% accuracy because many words are unambiguous.

CLAWS pioneered the field of HMM-based part of speech tagging, but was quite expensive since it enumerated all possibilities. It sometimes had to resort to backup methods when there were simply too many options.

HMMs underlie the functioning of stochastic taggers and are used in various algorithms one of the most widely used being the bi-directional inference algorithm.

2.2.2 Dynamic Programming Methods

In 1987, Steven DeRose and Ken Church independently developed dynamic programming algorithms to solve the same problem in vastly less time. Their methods were similar to the Viterbi algorithm known for some time in other fields. DeRose used a table of pairs, while Church used a table of triples and a method of estimating the values for triples that were rare or nonexistent in the Brown Corpus (actual measurement of triple probabilities would require a much larger corpus). Both methods achieved accuracy over 95%. DeRose's

1990 dissertation at Brown University included analyses of the specific error types, probabilities, and other related data, and replicated his work for Greek, where it proved similarly effective.

These findings were surprisingly disruptive to the field of natural language processing. The accuracy reported was higher than the typical accuracy of very sophisticated algorithms that integrated part of speech choice with many higher levels of linguistic analysis: syntax, morphology, semantics, and so on. CLAWS, DeRose's and Church's methods did fail for some of the known cases where semantics is required, but those proved negligibly rare. This convinced many in the field that part-of-speech tagging could usefully be separated out from the other levels of processing; this in turn simplified the theory and practice of computerized language analysis, and encouraged researchers to find ways to separate out other pieces as well. Markov Models are now the standard method for part-of-speech assignment.

2.2.3Unsupervised taggers

The methods already discussed involve working from a pre-existing corpus to learn tag probabilities. It is, however, also possible to bootstrap using "unsupervised" tagging. Unsupervised tagging techniques use an untagged corpus for their training data and produce the tagset by induction. That is, they observe patterns in word use, and derive part-of-speech categories themselves. For example, statistics readily reveal that "the", "a", and "an" occur in similar contexts, while "eat" occurs in very different ones. With sufficient iteration, similarity classes of words emerge that are remarkably similar to those human linguists would expect; and the differences themselves sometimes suggest valuable new insights.

These two categories can be further subdivided into rule-based, stochastic, and neural approaches.

2.2.4 Other taggers and methods

Some current major algorithms for part-of-speech tagging include the Viterbi algorithm, Brill tagger, Constraint Grammar, and the Baum-Welch algorithm (also known as the forward-backward algorithm). Hidden Markov model and visible Markov model taggers can both be implemented using the Viterbi algorithm. The rule-based Brill tagger is unusual in that it learns a set of rule patterns, and then applies those patterns rather than optimizing a statistical quantity. Unlike the Brill tagger where the rules are ordered

sequentially, the POS and morphological tagging toolkit RDRPOSTagger stores rules in the form of a ripple-down rules tree.

Many machine learning methods have also been applied to the problem of POS tagging. Methods such as SVM, maximum entropy classifier, perceptron, and nearest-neighbor have all been tried, and most can achieve accuracy above 95%.

A direct comparison of several methods is reported (with references) at the ACL Wiki. This comparison uses the Penn tag set on some of the Penn Treebank data, so the results are directly comparable.

However, many significant taggers are not included (perhaps because of the labor involved in reconfiguring them for this particular dataset). Thus, it should not be assumed that the results reported there are the best that can be achieved with a given approach; nor even the best that have been achieved with a given approach.

A more recent development is using the structure regularization method for part-of-speech tagging, achieving 97.36% on the standard benchmark dataset

2.3 Prior work on POST in English

Greene and Rubin have used a rule-based approach in the TAGGIT program, which was an aid in tagging the Brown corpus. TAGGIT disambiguated 77% of the corpus; the rest was done manually over a period of several years.

Derouault and Merialdo have used a bootstrap method for training. At first, a relatively small amount of text was manually tagged and used to train a partially accurate model. The model was then used to tag more text, and the tags were manually corrected and then used to retrain the model. Church uses the tagged Brown corpus for training. These models involve probabilities for each word in the lexicon and hence a large tagged corpus is required for a reliable estimation. Jelinekhas used Hidden Markov Model (HMM) for training a text tagger. Parameter smoothing can be conveniently achieved using the method of ‘deleted interpolation’ in which weighted estimates are taken from second and first-order models and a uniform probability distribution.

Kupiecused word equivalence classes (referred to here as ambiguity classes) based on parts of speech, to pool data from individual words. The most common words are still represented individually, as sufficient data exist for robust estimation.Yahya O. Mohamed Elhadjpresents the development of an Arabic part-of-speech tagger that can be used for

analyzing and annotating traditional Arabic texts, especially the Quran text. The developed tagger employed an approach that combines morphological analysis with Hidden Markov Models (HMMs) based-on the Arabic sentence structure. The morphological analysis is used to reduce the size of the lexicon tags by segmenting Arabic words in their prefixes, stems and suffixes; this is due to the fact that Arabic is a derivational language. On the other hand, HMM is used to represent the Arabic sentence structure in order to take into account the linguistic combinations.

In the recent literature, several approaches to POS tagging based on statistical and machine learning techniques are applied, including Hidden Markov Models, Maximum Entropy taggers , Transformation–based learning , Memory–based learning , Decision Trees , and Support Vector Machines . Most of the previous taggers have been evaluated on the English WSJ corpus, using the Penn Treebank set of POS categories and a lexicon constructed directly from the annotated corpus. Although the evaluations were performed with slight variations, there was a wide consensus in the late 90's that the state–of-the–art accuracy for English POS tagging was between 96.4% and 96.7%. In the recent years, the most successful and popular taggers in the NLP community have been the HMM–based TnT tagger, the Transformation–based learning (TBL) tagger and several variants of the Maximum Entropy (ME) approach.

The SVM Tools intended to comply with all the requirements of modern NLP technology, by combining simplicity, flexibility, robustness, portability and efficiency with state–of–the–art accuracy. This is achieved by working in the Support Vector Machines (SVM) learning framework, and by offering NLP researchers a highly customizable sequential tagger generator.

TnT is an example of a really practical tagger for NLP applications. It is available to anybody, simple and easy to use, considerably accurate, and extremely efficient, allowing training from 1 million word corpora in just a few seconds and tagging thousands of words per second. In the case of TBL and ME approaches, the great success has been due to the flexibility they offer in modeling contextual information, being ME slightly more accurate than TBL. In the next chapter we are going to discuss part of speech tagger – English.

CHAPTER 3

PART OF SPEECH TAGGER FOR ENGLISH

PART OF SPEECH TAGGER FOR ENGLISH LANGUAGE

3.1 English Language

English is a West Germanic language that was first spoken in early medieval England and is now a global lingua franca. Named after the Angles, one of the Germanic tribes that migrated to England, it ultimately derives its name from the Anglia (Angeln) peninsula in the Baltic Sea. It is closely related to the Frisian languages, but its vocabulary has been significantly influenced by other Germanic languages, particularly Norse (a North Germanic language), as well as by Latin and Romance languages, especially French.

English has developed over the course of more than 1,400 years. The earliest forms of English, a set of Anglo-Frisian dialects brought to Great Britain by Anglo-Saxon settlers in the 5th century, are called Old English. Middle English began in the late 11th century with the Norman conquest of England, and was a period in which the language was influenced by French.Early Modern English began in the late 15th century with the introduction of the printing press to London and the King James Bible, and the start of the Great Vowel Shift.

Through the worldwide influence of the British Empire, modern English spread around the world from the 17th to mid-20th centuries. Through all types of printed and electronic media, as well as the emergence of the United States as a global superpower, English has become the leading language of international discourse and the lingua franca in many regions and in professional contexts such as science, navigation and law.

English is the third most widespread native language in the world, after Standard Chinese and Spanish. It is the most widely learned second language and is either the official language or one of the official languages in almost 60 sovereign states. There are more people who have learned it as a second language than there are native speakers. English is the most commonly spoken language in the United Kingdom, the United States, Canada, Australia, Ireland and New Zealand, and it is widely spoken in some areas of the Caribbean, Africa and South Asia.It is co-official language of the United Nations, of the European Union and of many other world and regional international organisations. It is the most widely spoken Germanic language, accounting for at least 70% of speakers of this Indo-

European branch. English has a vast vocabulary, and counting exactly how many words it has is impossible.

Modern English grammar is the result of a gradual change from a typical Indo-European dependent marking pattern with a rich inflectional morphology and relatively free word order, to a mostly analytic pattern with little inflection, a fairly fixed SVO word order and a complex syntax. Modern English relies more on auxiliary verbs and word order for the expression of complex tenses, aspect and mood, as well as passive constructions, interrogatives and some negation. Despite noticeable variation among the accents and dialects of English used in different countries and regions – in terms of phonetics and phonology, and sometimes also vocabulary, grammar and spelling – English-speakers from around the world are able to communicate with one another with relative ease.

3.2 Problem in Part of Speech Tagger - English

The first step in implementing a part-of-speech tagger is to build a lexicon, where the part-of-speech of a word can be found. Unfortunately, many words are ambiguous, and each word can therefore have several classifications. As an example, the word "note" can be either a noun or a verb. It is the object of the part-of-speech tagger to resolve these ambiguities, using the context of the word. Another problem is the handling of words that have no entries in the lexicon. There are basically two approaches to part-of-speech tagging: rule-based tagging and stochastic tagging. This paper describes an implementation using the rule-based approach, where the rules are generated using transformation-based learning.

Transformation-based error-driven learning is a machine learning method typically used for classification problems, where the goal is to assign classifications to a set of samples. An initial classification is produced using a simple algorithm. In each iteration the current classification is compared to the correct classification and transformations are generated to correct the errors. The output of the algorithm is a list of transformations that can be used for automatic classification, together with the initial classifier.

There are two components of a transformation: a rewrite rule and a triggering environment. The rewrite rule says what should be done (e.g. change the class from A to B) and the triggering environment says when it should be done (e.g. if the preceding sample is of class C). Transformation-based learning is used in many diff erent areas, and has proven

to be a very successful method in the field of natural language processing. The algorithm was introduced for POS tagging by Eric Brill in 1995.

There has been a dramatic increase in the application of probabilistic models to natural language processing over the last few years. The appeal of stochastic techniques over traditional rule-based techniques comes from the ease with which the necessary statistics can be automatically acquired and the fact that very little handcrafted knowledge need be built into the system. In contrast, the rules in rule-based systems are usually difficult to construct and are typically not very robust.

A rule-based tagger which performs as well as taggers based upon probabilistic models. The rule-based tagger overcomes the limitations common in rule-based approaches to language processing: it is robust, and the rules are automatically acquired. In addition, the tagger has many advantages over stochastic taggers, including: a vast reduction in stored information required the perspicuity of a small set of meaningful rules as opposed to the large tables of statistics needed for stochastic taggers, ease of finding and implementing improvements to the tagger, and better portability from one tag set or corpus genre to another.

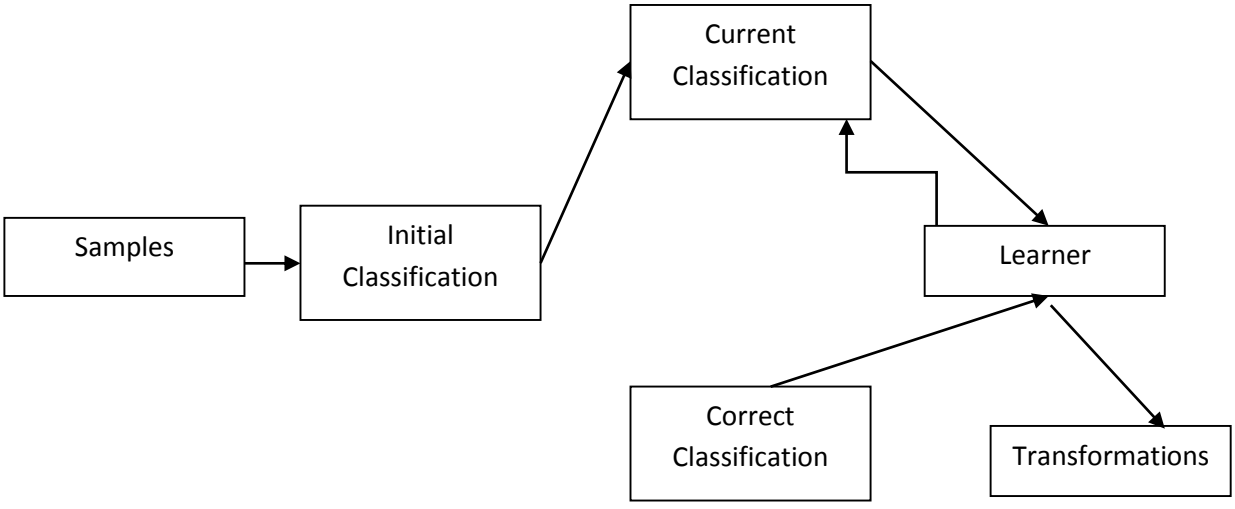


Figure 3.2: Transformation-based learning

The algorithm first assigns every word it's most likely part-of-speech, i.e. the most common tag for that word. This initial annotation is compared to a hand-annotated corpus, and a list of errors is produced. For each error, rules to correct the error are instantiated from a set of rule templates. Each instantiated rule is evaluated by computing its impact on the whole corpus. The rules are compared by assigning each rule a score, which is the diff erence between the number of good transformations and the number of bad transformations the rule produces. The rule with the highest score is applied to the text and added to the result list. The transformed corpus is then used to generate a new rule in the next iteration. The algorithm stops when a certain criteria have been fulfilled.

3.3 Design challenges in POS for English Language

POS is most challenging task in the field of NLP. Most of the text processing applications such as search systems, spelling checkers do not treat proper names correctly. This implies that names are difficult to identify and interpret in unstructured data. There are several reasons for this difficulty. There are some rules like capitalization in English language to identify named entities.

The input to a tagging algorithm is a sequence of words and a tagset, and the output is a sequence of tags, a single best tag for each word. Tagging is a disambiguation task; words are ambiguous have more than one ambiguous possible part-of-speech—and the goal is to find the correct tag for the situation. For example, the word book can be a verb (book that flight) or a noun (as in hand me that book. That can be a determiner (Does that flight serve dinner) or a complementizer (I thought that your flight was earlier). The problem of POS-tagging is to resolve resolution these ambiguities, choosing the proper tag for the context. Part-of-speech tagging is thus one of the many disambiguation tasks in language processing.

Some of the most ambiguous frequent words are that, back, down, put and set; here are some examples of the 6 different parts-of-speech for the word back:

Earnings growth took aback/JJ seat

A small building in the back/NN

A clear majority of senator's back/VBP the bill

Dave began to back/VB toward the door

Enable the country to buyback/RP about debt

I was twenty-one back/RB then

3.4 Outline of the Project

POST system is divided into three tier process. In first step system identifies input language and returns the confidence score based on it language is detected when it is English language the input is processed. The tokenizer converts input into tokens and initial tagger will tag appropriate part of speech to each token. Ambiguity will resolved using predefined sets else tags most suitable tag.

It provides multiple user interfaces for various form factor devices each transaction will perform in cloud environment and in final phase the result is compared with the Stanford NLP Library and improves machine accuracy. In next chapter we are going to discuss system analysis.

CHAPTER 4
SYSTEM ANALYSIS

SYSTEM ANALYSIS

4.1 Problem Statement

POS tagging also called as grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context i.e., its relationship with adjacent and related words in a phrase, sentence or a paragraph.

4.2 Proposed Solution

- Learner
- Contextual Rule Learner
- Unknown word Rule Learner
- Rule
- Corpus Reader
- Tagger
- Tokenizer
- Word Dictionary
- Tag Dictionary

Learner

Learner is the main class of the learning program and contains the general learning algorithm. It is an abstract class that requires the subclasses to implement some parts of the algorithm.

Contextual Rule Learner

Learner and is responsible for the contextual rule learning.

Unknown Word Rule Learner

Unknown Word Rule Learner is also a subclass of Learner and is responsible for learning the unknown word rules.

Rule

Rule is the super class of all rules. It contains the abstract methods instantiate, predicate,

Unknown Word Rule

Unknown Word Rule is the super class of all unknown word rules .Rule List is a class for maintaining a list of rules.

Change tag a to tag b when:

1. The preceding (following) word is tagged z

2. The word two before (after) is tagged z

3. One of the two preceding (following) words is tagged z

4. One of the three preceding (following) words is tagged z

5. The preceding word is tagged z and the following word is tagged w

6. The preceding (following) word is tagged z and the word two before (after) is tagged w

where a, b, z and w are tag variables.

The rule templates proposed by Brill

Corpus Reader

Its Reads tags of known words from a database.

Tag Dictionary

Tags are stored in a database for a record.

Learning Rules for Unknown Words are defined in programming.

```
While (nbr of errors> threshold)
        for (each error)
                for (each rule r correcting the error)
                        good(r) = nbr of good transformations
                        bad(r) = nbr of bad transformations
                        score(r) = good(r) - bad(r)
                        Apply the rule with highest score and append it to the rule list
```

Pseudo code for Brill's learning algorithm

4.3 Disadvantages of existing system

- Brill method differs from other part of speech tagging methods such as those using Hidden Markov Models.
- The Brill tagger was written by Fran¸coisMarier and BengtSj¨odin. It contained the basic Brill algorithm and the contextual rule templates, but was too slow for practical use.
- Long running time and no handling of unknown words
- No advanced mechanism for input language detection.

4.4 System requirement specifications

4.4.1 Hardware specifications

- Hard Disk : 150 GB
- Processor : 3.0 GHz
- Ram : 4 GB

4.4.2 Software specifications

- Operating system : Windows 7 / Android 4.4
- Coding Language : PHP, HTML 5, JQuery
- Database : MYSQL
- Server : XAMPP
- Cloud : Microsoft Azure

4.4.3 Dependencies:

- Stanford NLP Library and Microsoft Azure Cognitive services

4.5 Features of selected software

Cloud computing is the delivery of computing services servers, storage, databases, networking, software, analytics and more over the Internet (“the cloud”).

4.5.1 Azure

Azure is a comprehensive set of cloud services that developers and IT professionals use to build, deploy and manage applications through our global network of datacenters. Integrated tools, DevOps and a marketplace support you in efficiently building anything from simple mobile apps to internet-scale solutions.

Azure is the only consistent hybrid cloud

Build and deploy wherever you want with Azure, the only consistent hybrid cloud on the market. Connect data and apps in the cloud and on-premises for maximum portability and value from your existing investments. Azure offers hybrid consistency in application development, management and security, identity management and across the data platform.

Extend Azure on-premises and build innovative, hybrid apps with Azure Stack. Connect on-premises data and apps to overcome complexity and optimise your existing assets. Distribute and analyse data seamlessly across cloud and on-premises.

Azure is the cloud for building intelligent apps

Use Azure to create data-driven, intelligent apps. From image recognition to bot services, take advantage of Azure data services and artificial intelligence to create new experiences that scale and support deep learning, HPC simulations and real-time analytics on any shape and size of data.

- Develop breakthrough apps with built-in AI.
- Build and deploy custom AI models at scale, on any data.
- Combine the best of Microsoft and open source data and AI innovations.

4.5.2 PHP

PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is

integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time. PHP is forgiving: PHP language tries to be as forgiving as possible. PHP Syntax is C-Like.

Common uses of PHP

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them. It can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user. You add, delete, modify elements within your database through PHP. Access cookies variables and set cookies. Using PHP, you can restrict users to access some pages of your website. It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible

Simplicity, Efficiency, Security, Flexibility and Familiarity

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
```

```
<?php echo "Hello, World!";?>
</body>
</html>
```

It will produce following result

```
Hello, World!
```

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

```
<?php PHP code goes here ?>
<?    PHP code goes here ?>
<scriptlanguage="php"> PHP code goes here </script>
```

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

From the next chapter we will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- **Web Server** − PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- **Database** − PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

- **PHP Parser** − In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP has a total of eight data types which we use to construct our variables

- **Integers** − are whole numbers, without a decimal point, like 4195.
- **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** − have only two possible values either true or false.
- **NULL** − is a special type that only has one value: NULL.
- **Strings** − are sequences of characters, like 'PHP supports string operations.'
- **Arrays** − are named and indexed collections of other values.
- **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

PHP provides **mysql_connect** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or FALSE on failure.

Syntax

```
mysql_connect(server,user,passwd,new_link,client_flag);
```

Its simplest function **mysql_close** PHP provides to close a database connection. This function takes connection resource returned by mysql_connect function. It returns TRUE on success or FALSE on failure.

Syntax

```
mysql_close ( resource $link_identifier );
```

4.5.3 MYSQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as **Foreign Keys**.

A **Relational DataBase Management System (RDBMS)** is a software that

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

4.5.4 JQuery

jQuery is a fast and concise JavaScript library created by John Resig in 2006. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development.

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto − **Write less, do more**.

jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Loading simple data

This is very easy to load any static or dynamic data using JQuery AJAX. JQuery provides **load()** method to do the job −

Syntax

Here is the simple syntax for **load()** method −

```
[selector].load( URL, [data], [callback] );
```

Here is the description of all the parameters −

- **URL** − The URL of the server-side resource to which the request is sent. It could be a CGI, ASP, JSP, or PHP script which generates data dynamically or out of a database.
- **data** − This optional parameter represents an object whose properties are serialized into properly encoded parameters to be passed to the request. If specified, the request is made using the **POST** method. If omitted, the **GET** method is used.
- **callback** − A callback function invoked after the response data has been loaded into the elements of the matched set. The first parameter passed to this function is the response text received from the server and second parameter is the status code.

jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

Syntax:

$.get(*URL,callback*);

The required URL parameter specifies the URL you wish to request. The optional callback parameter is the name of a function to be executed if the request succeeds.

jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

Syntax:

$.post(*URL,data,callback*);

The required URL parameter specifies the URL you wish to request. The optional data parameter specifies some data to send along with the request. The optional callback parameter is the name of a function to be executed if the request succeeds.

4.6 Feasibility Study

Feasibility study is conducted once the problem is clearly understood. The feasibility study which is a high level capsule version of the entire system analysis and design process. The objective is to determine whether the proposed system is feasible or not and it helps us to the minimum expense of how to solve the problem and to determine, if the Problem is worth solving. The following are the three important tests that have been carried out for feasibility study.

4.6.1 Technical feasibility

In the technical feasibility study, one has to test whether the proposed system can be developed using existing technology or not. It is planned to implement the proposed system in PHP and Cloud Environment. The project entitled is technically feasible because of the following reasons.

- All necessary technology exists to develop the system.
- The existing system is so flexible that it can be developed further.

4.6.2 Economic feasibility

As a part of this, the costs and benefits associated with the proposed systems are to be compared. The project is economically feasible only if tangible and intangible benefits outweigh the cost. We can say the proposed system is feasible based on the following grounds.

- The cost of developing the full system is reasonable.

But in order to business and research oriented we deployed in Microsoft Azure Virtual Machines its cost is moderated and high elasticity in nature depends on users load cloud mainly reduces operational and maintenance cost.

4.6.3 Operational feasibility

The project is operationally feasible because there is sufficient support from the project management and the users of the proposed system. Proposed system definitely does not harm and will not produce the bad results and no problem will arise after implementation of the system.

User-friendly

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

Reliability

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

Security

The web server and database server should be protected from hacking, virus etc.

Availability

This software will be available always. In next chapter we are going to discuss system design and implementation.

CHAPTER 5
SYSTEM DESIGN
AND
IMPLEMENTATION

SYSTEM DESIGN AND IMPLEMENTATION

5.1 System Design

5.1.1 Block Diagram

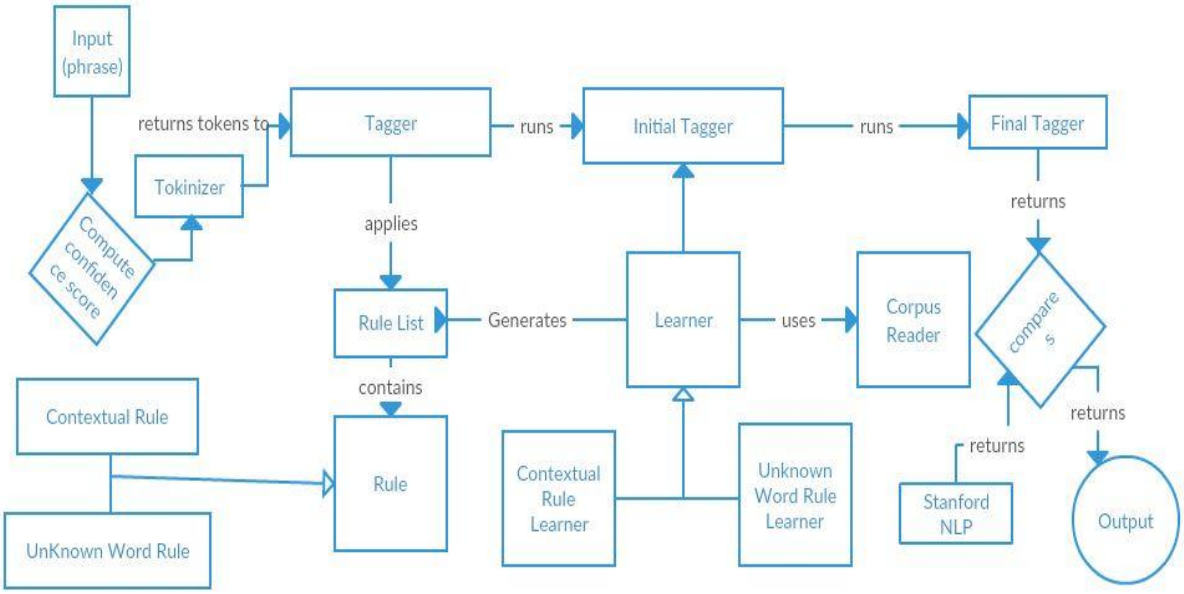


Figure 5.1.1 Block diagram of POST for English

5.1.2 Input and Output design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?

- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follows

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

5.1.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.

5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Use case diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

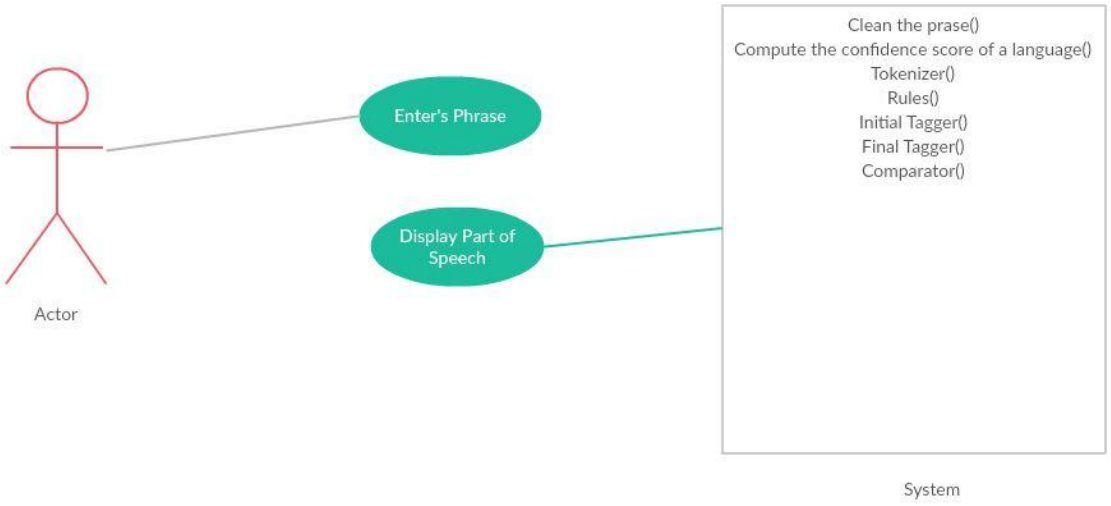


Figure 5.1.3.1: Use Case

Class diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

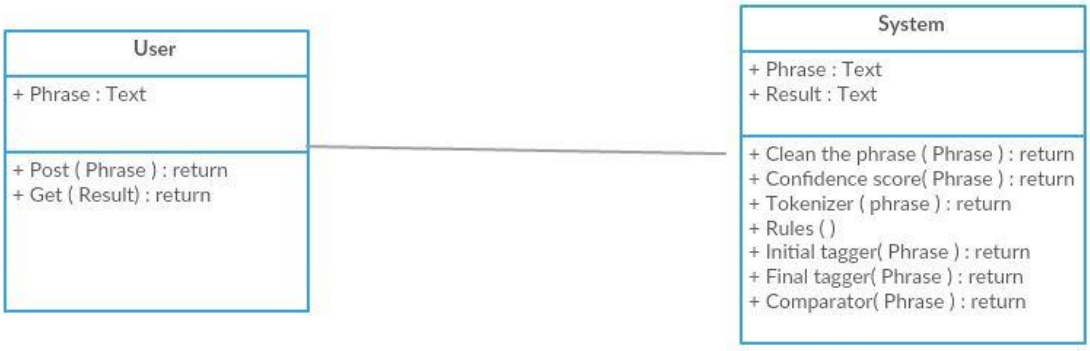


Figure 5.1.3.2: Class Diagram

Sequence diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

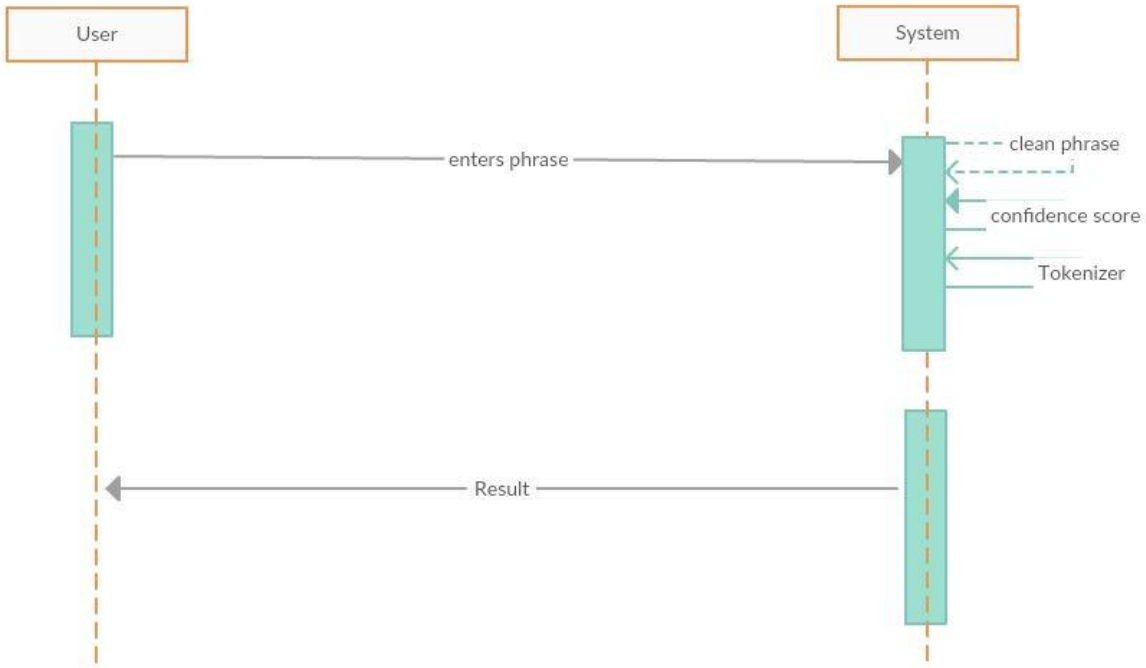


Figure 5.1.3.3: Sequence diagram

Collaboration diagram:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

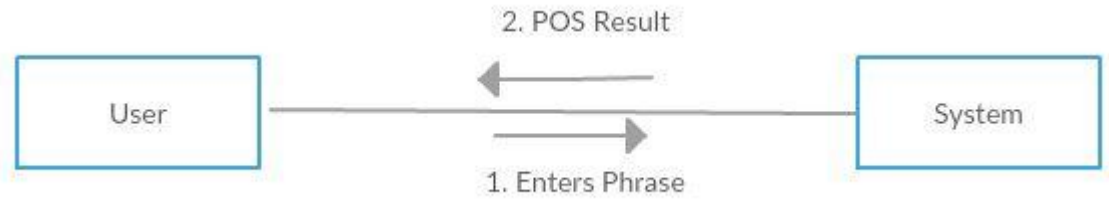


Figure 5.1.3.4: Collaboration diagram

Component diagram

A component diagram describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

Figure 5.1.3.5: Component Diagram

Activity diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling

Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

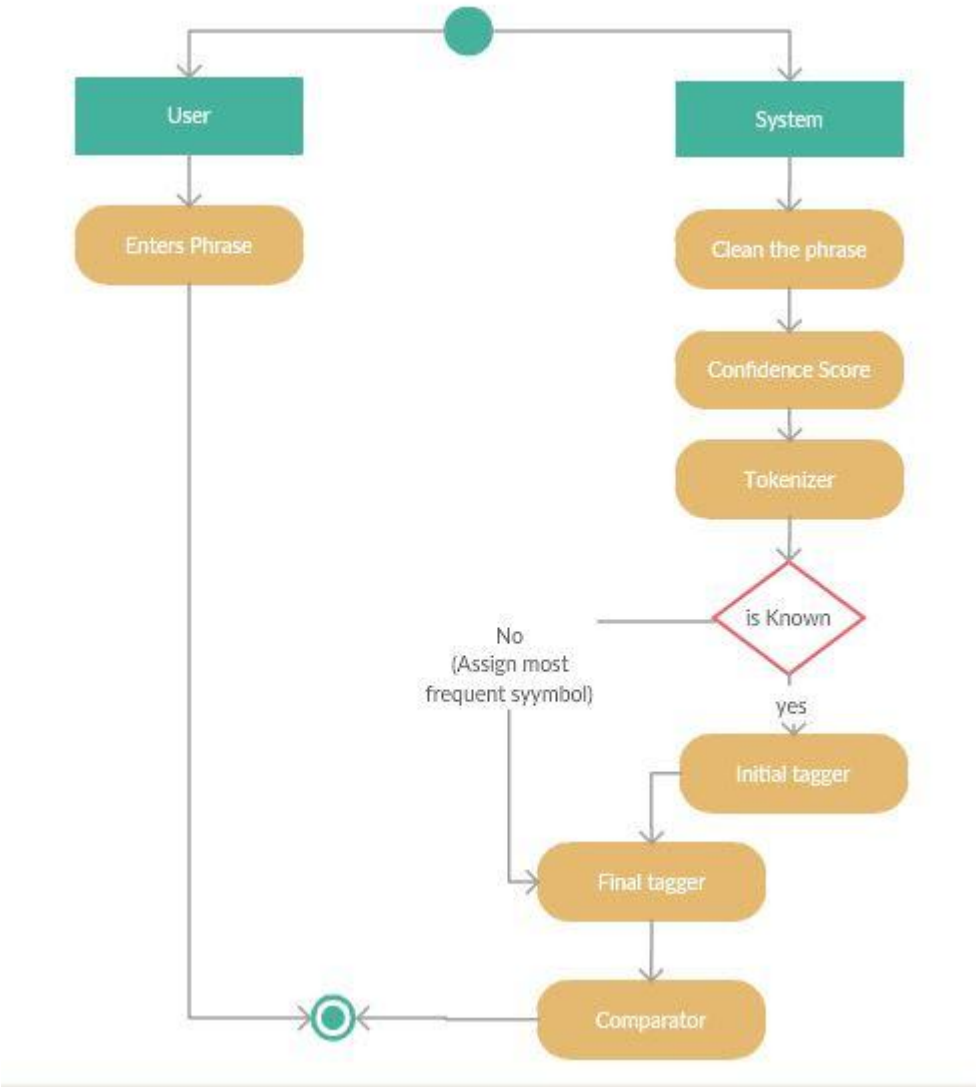


Figure 5.1.3.6: Activity Diagram

Deployment Diagram:

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server,

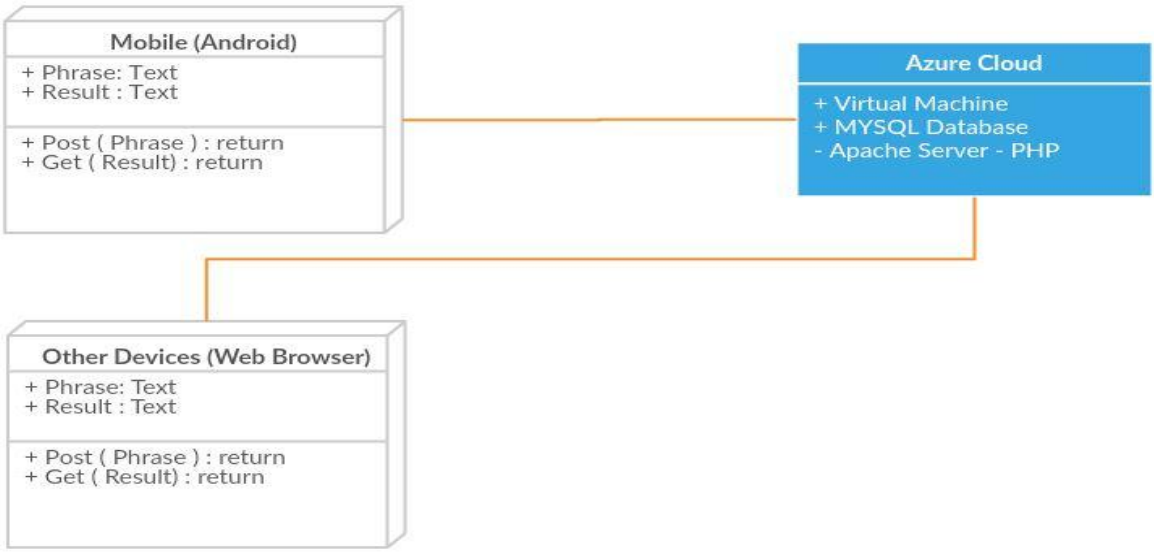
and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

1. Device Node
2. Execution Environment Node

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.



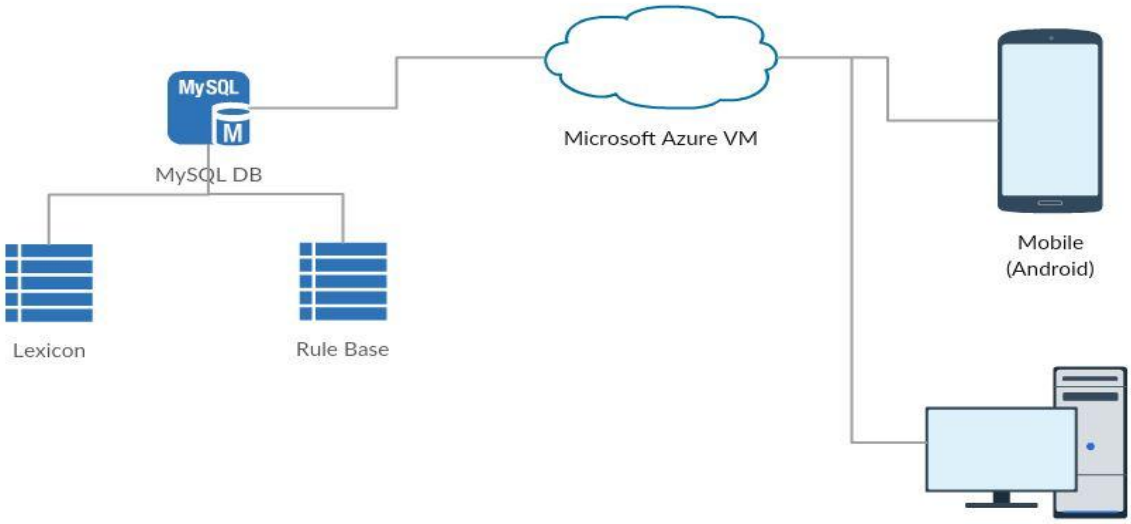


Figure 5.1.3.7: Deployment Diagram

5.2 Implementation

MODULES

1. Learner
2. Contextual RuleLearner
3. Unknown word RuleLearner
4. Rule
5. CorpusReader
6. Tagger
7. Tokenizer
8. Word Dictionary
9. Tag Dictionary

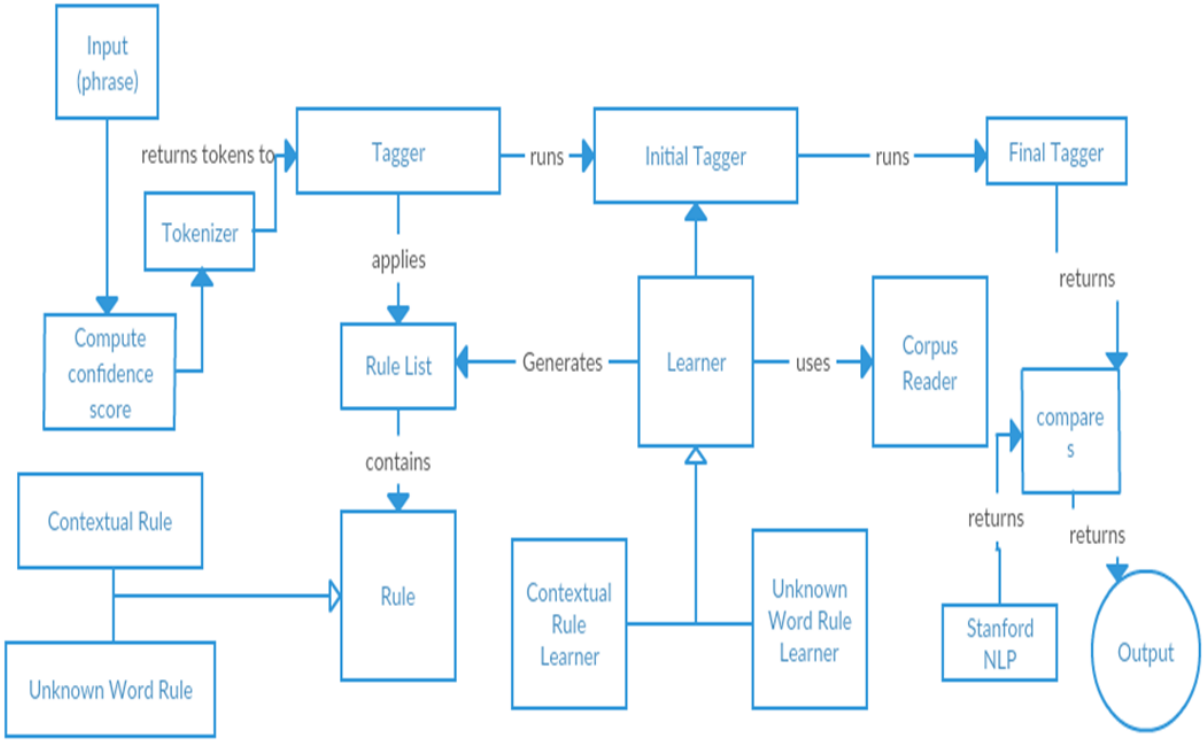


Figure 5.2: Proposed Work

5.2.1 Modules Description

Initially input phrase is processed and finds a confidence score of a language that resolves more ambiguity. After final tagger results are compared with Stanford NLP library results.

5.2.1.1 Learner

Learner is the main class of the learning program and contains the general learning algorithm. It is an abstract class that requires the subclasses to implement some parts of the algorithm.

Pseudo code for Brill's learning algorithm

```
While (nbr of errrors> threshold)
        for (each error)
                for (each rule r correcting the error)
                        good(r) = nbr of good transformations
                        bad(r) = nbr of bad transformations
                        score(r) = good(r) - bad(r)
                        Apply the rule with highest score and append it to the rule list
```

5.2.1.2 Contextual Rule Learner

Learner and is responsible for the contextual rule learning.

5.2.1.3 Unknown Word Rule Learner

Unknown Word Rule Learner is also a subclass of Learner and is responsible for learning the unknown word rules.

5.2.1.4 Rule

Rule is the superclass of all rules. It contains the abstract methods instantiate, predicate, evaluate and apply. ContextualRule is the superclass of all contextual rules. UnknownWordRule is the superclass of all unknown word rules.RuleList is a class for maintaining a list of rules.

Change tag a to tag b when: 1. The preceding (following) word is tagged z 2. The word two before (after) is tagged z 3. One of the two preceding (following) words is tagged z 4. One of the three preceding (following) words is tagged z 5. The preceding word is tagged z and the following word is tagged w 6. The preceding (following) word is tagged z and the word two before (after) is tagged w where a, b, z and w are tag variables.

The rule templates proposed by Brill

Change the tag of an unknown word from a to b when: 1. Deleting the prefix (suffix) x results in a word 2. The prefix (suffix) of the word is x 3. Adding the prefix (suffix) x results in a word 4. The preceding (following) word is w 5. The character c appears in the word where x is a string of length 1 to 4.

The rule template for unknown words

```
$noun = "NN";
$proper = "NNP";
$number = "CD";
        for($cnt=0;$cnt<sizeof($ntothash);++$cnt){
        if(ord($ntotkeys[$cnt]) > 47  && ord($ntotkeys[$cnt]) < 59)
        $ntothash[$ntotkeys[$cnt]] = $number;
        elseif(ord($ntotkeys[$cnt]) > 64 && ord($ntotkeys[$cnt]) < 91)
        $ntothash[$ntotkeys[$cnt]] = $proper;
        else
        $ntothash[$ntotkeys[$cnt]] = $noun;
        }
```

5.2.1.5 Corpus Reader

Corpus Reader is responsible for extracting words and tags from the manually annotated corpus.

5.2.1.6 Tagger

Tagger is the main class of the tagging program. It takes an untagged text as input and produces a tagged text as output.

The tagging is done in three steps.

1. Initial tagging
2. Application of unknown word rules
3. Application of contextual rules

Code for initialize

Initilizer is a function that defines lexical rules like left and right.

```
<?php
function initialize(){
        global $lexrulesright, $lexrulesleft, $bigramarray;
$lexrulesright = array("$" => "","would" => "","be" => "","it" => "","n't" => "",
```

```
"the" => "","he" => "","he" => "","are","The" => "","he" => "","Mr." => "","so" =>
"","which" => "","been" => "","a" => "","can" => "","the" => "","S-T-A-R-T" =>
"","very", "It" => "","but" => "","costs" => "","negative" => "","Engelken" => "");
$lexrulesleft = array("was" => "","is" => "","Co." => "","million" => "","their" => "","of"
=> "","economic" => "","'s" => "","Inc." => "","be" => "","people" => "","have" =>
"","may" => "","them" => "","were" => "","but" => "","ways" => "");
$bigramarray = array("NOOTHING NOOTHING");
        //lexicon and rules are loaded into databases during installation.
} ?>
```

5.2.1.7 Tokenizer

Tokenizer is used by the Tagger to divide the input text into tokens.

Code for Tokenizer

```
<?php
function tokenizer($txt){
        $txt = $txt." ";
        $token_list = array("\"", " \" ","," , " , ",";", " ; ",":", " : ","? ", " ? ","! ", " ! ", ". ",
" .","[", " [ ","]", " ] "," (", " ( ",") ", " ) ",").", " ).","<", " < ", ">", " > ","--", " -- ","'s ", " 's
","'S ", " 'S ","'m ", " 'm ","'M ", " 'M ", "'d ", " 'd ","'D ", " 'D ","'ll ", " 'll ","'re ", " 're ","'ve
", " 've ", " can't ", " can n't "," Can't ", " Can n't ","n't ", " n't ","'LL ", " 'LL ", "'RE ", " 'RE
","'VE ", " 'VE ","N'T ", " N'T "," Cannot ", " Can not "," cannot ", " can not "," D'ye ", "
D' ye "," d'ye ", " d' ye "," Gimme ", " Gim me "," gimme ", " gim me "," Gonna ", " Gon
na "," gonna ", " gon na "," Gotta ", " Got ta "," gotta ", " got ta ",Lemme ", " Lem me ","
lemme ", " lem me "," More'n ", " More 'n "," more'n ", " more 'n ","'Tis ", " 'T is ","'tis ", "
't is ","'Twas ", " 'T was ","'twas ", " 't was "," Wanna ", " Wan na "," wanna ", " wanna
",Let 's ", "Let's ",".. .", " ..."","Adm .", "Adm.","Aug .", "Aug.","Ave .", "Ave.",  "Brig .",
"Brig.","Bros .", "Bros.","CO .", "CO.", "CORP .", "CORP.","COS .", "COS.", "Capt .",
"Capt.","Co .", "Co.","Col .", "Col.","Colo .", "Colo.","Corp .", "Corp.", "Cos .",
"Cos.","Dec .", "Dec.","Del .", "Del.","Dept .", "Dept.","Dr .", "Dr.", "Drs .", "Drs.","Etc
.", "Etc.","Feb .", "Feb.","Ft .", "Ft.","Ga .", "Ga.", "Gen .", "Gen.","Gov .", "Gov.","Hon
.", "Hon.","INC .", "INC.","Inc .", "Inc.", "Ind .", "Ind.","Jan .", "Jan.","Jr .", "Jr.","Kan .",
```

```
"Kan.","Ky .", "Ky.", "La .", "La.","Lt .", "Lt.","Ltd .", "Ltd.","Maj .", "Maj.","Md .",
"Md.", "Messrs .", "Messrs.","Mfg .", "Mfg.","Miss .", "Miss.","Mo .", "Mo.", "Mr .",
"Mr.","Mrs .", "Mrs.","Ms .", "Ms.","Nev .", "Nev.","No .", "No.", "Nos .", "Nos.","Nov
.", "Nov.","Oct .", "Oct.","Ph .", "Ph.","Prof .", "Prof.",    "Prop .", "Prop.","Pty .",
"Pty.","Rep .", "Rep.","Reps .", "Reps.","Rev .", "Rev.", "S.p.A .", "S.p.A.","Sen .",
"Sen.","Sens .", "Sens.","Sept .", "Sept.","Sgt .", "Sgt.", "Sr .", "Sr.","St .", "St.","Va .",
"Va.","Vt .", "Vt.","U.S .", "U.S.","Wyo .", "Wyo.",        "a.k.a .", "a.k.a.","a.m .",
"a.m.","cap .", "cap.","e.g .", "e.g.","eg .", "eg.", "etc .", "etc.","ft .", "ft.","i.e .", "i.e.","p.m
.", "p.m.","v .", "v.", "v.B .", "v.B.","v.w .", "v.w.","vs .", "vs.","__END__", "__END__");
$i = 0;
for($i = 0; $i < 150; $i++){
$txt = str_replace($token_list[(2*$i)], $token_list[(2*$i)+1], $txt);
}
return $txt;
}
?>
```

5.2.1.8 Word Dictionary

Word Dictionary contains all words of the training corpus. It is used for finding the most likely tag for a word, investigating if a word exists and searching for prefixes or suffixes of words.

5.2.1.9 Tag Dictionary

Tag Dictionary is responsible for the translation between the string and integer representation of the part-of-speech tags.

Part of speech category

Tag	Description	Example words
$	Dollar	$
``	opening quotation mark	` ``
''	closing quotation mark	' ''
(	opening parenthesis	([{

)	closing parenthesis	)] }
,	Comma	,
--	Dash	--
.	sentence terminator	. ! ?
:	colon or ellipsis	: ; ...
CC	conjunction, coordinating	and but or yet
CD	numeral, cardinal	nine 20 1980 '96
DT	Determiner	a the an all both neither
EX	existential there	there
FW	foreign word	enfant terrible hoi polloi je ne sais quoi
IN	preposition or subordinating conjunction	in inside if upon whether
JJ	adjective or numeral, ordinal	ninth pretty execrable multimodal
JJR	adjective, comparative	better faster cheaper
JJS	adjective, superlative	best fastest cheapest
LS	list item marker	(a) (b) 1 2 A B A. B.
MD	modal auxiliary	can may shall will could might should ought
NN	noun, common, singular or mass	potato money shoe
NNP	noun, proper, singular	Kennedy Roosevelt Chicago Weehauken
NNPS	noun, proper, plural	Springfields Bushes
NNS	noun, common, plural	pieces mice fields
PDT	pre-determiner	all both half many quite such sure this
POS	genitive marker	' 's
PRP	pronoun, personal	she he it I we they you
PRP$	pronoun, possessive	hers his its my our their your
RB	Adverb	clinically only
RBR	adverb, comparative	further gloomier grander graver greater grimmer harder harsher healthier heavier higher however larger later leaner lengthier

		less-perfectly lesser lonelier longer louder lower more ...
RBS	adverb, superlative	best biggest bluntest earliest farthest first furthest hardest heartiest highest largest least less most nearest second tightest worst
RP	Particle	on off up out about
SYM	Symbol	% &
TO	"to" as preposition or infinitive marker	to
UH	Interjection	uh hooray howdy hello
VB	verb, base form	give assign fly
VBD	verb, past tense	gave assigned flew
VBG	verb, present participle or gerund	giving assigning flying
VBN	verb, past participle	given assigned flown
VBP	verb, present tense, not 3rd person singular	give assign fly
VBZ	verb, present tense, 3rd person singular	gives assigns flies
WDT	WH-determiner	that what which
WP	WH-pronoun	who whom
WP$	WH-pronoun, possessive	whose
WRB	Wh-adverb	how however whenever where

5.2.2 The contextual transformation template:

1. The preceding/following word is tagged with Z. (PREVTAG/NEXTTAG)

2. One of the two preceding/following words is tagged with Z. (PREV1OR2TAG/NEXT1OR2TAG)

3. One of the three preceding/following words is tagged with Z. (PREV1OR2OR3TAG/NEXT1OR2OR3TAG)

4. The preceding word is tagged with Z and the following word is tagged with V. (SURROUNDTAG)

5. The preceding/following two words are tagged with Z and V. (PREVBIGRAM/NEXTBIGRAM)

6. The word two words before/after is tagged with Z. (PREV2TAG/NEXT2TAG)

7. The current word is Z. (CURWD)

8. The preceding/following word is W. (PREVWD/NEXTWD)

9. One of the preceding/following words is W. (PREV1OR2WD/NEXT1OR2WD)

10. The word two words before/after is W. (PREV2WD/NEXT2WD)

11. The current word is Z and the preceding word is V. (LBIGRAM)

12. The current word is V and the following word is Z. (RBIGRAM)

13. The current word is V and the preceding/following word is tagged with Z. (WDPREVTAG/WDNEXTTAG)

14. The current word is V and the word two words before/after is tagged with Z. (WDAND2BFR/WDAND2TAGAFT)

5.3 Brill Algorithm

The algorithm starts with initialization, which is the assignment of tags based on their probability for each word (for example, "dog" is more often a noun than a verb). Then "patches" are determined via rules that correct (probable) tagging errors made in the initialization phase:

Initialization:

Known words (in vocabulary): assigning the most frequent tag associated to a form of the word

Unknown word

5.4 Learning lexical rules

The ideal goal of the lexical module is to find rules that can produce the most likely tag for any word in the given language, i.e. the most frequent tag for the word in question considering all texts in that language. The problem is to determine the most likely tags for unknown words, given the most likely tag for each word in a comparatively small set of words. The lexical learner module is weakly statistical. It uses the first half of the manually tagged corpus 3 as well as any additional large unannotated corpus containing the manually tagged corpus with the tags removed. The learner module uses three different lexicons or lists constructed (by the user) from the manually tagged corpus and the large unannotated corpus. The smallwordtaglist, built from the manually annotated corpus, serves as the goal corpus in TEL and contains lines of the form

Word Tag Frequency.

It simply contains the frequencies of each Word/Tag pair in the manually annotated corpus. Freq(W,T) denotes the Frequency of a word W with a specific tag T, and Freq(W) denotes the frequency of the word Word in the manually annotated corpus. These numbers are used by the lexical learner in two ways: Freq(W,T) is used to compute the most likely tag T for the word W (i.e. the one with the highest frequency), and

P(T|W) := Freq(W,T) / Freq(W)

is the estimated probability that the word W is tagged with the tag T. The other two lists are called bigwordlist and bigbigramlist and are constructed from the large unannotated corpus. bigwordlist is a list of all words occurring in the unannotated corpus, sorted by decreasing frequency. The bigbigramlist is a list consisting of all word pairs (hence the name bigram) occurring in the unannotated corpus. Bigbigramlist does not contain the frequencies of word pairs; it only records if a given word pair occurs in the unannotated corpus or not. Note that once the user has constructed these lists, the lexical learner does not need either the manually annotated or the unannotated corpus because the lists contain all the information the learner needs. The bigwordlist and the bigbigramlist are only used to check the trigger condition, which is completely determined by the data in these two lists. First, the learner constructs a word list from smallwordtaglist, i.e. the words with tagging information removed. Then the initial state annotator assigns to every word the default most likely tag. The default tags for

English are NN and NNP , where NNP is assigned to words which start with a capital letter and NN is assigned to words which do not start with a capital letter. The word list thus obtained is the initial temporary corpus WL0 ; it was called TC0 in the general description of TEL above. Once WL0 has been produced by the initial state annotator the learner generates the set of all permissable rules PR from all possible instantiations of all the predefined lexical templates and computes a score for every rule R in PR (see below). The rule which achieves the best score becomes rule number one in the output. Then the learner transforms WL0 to WL1 by applying this rule. This process is repeated until no rule can be found with a score greater than some threshold value, i.e. compute the new scores for all the rules in PR, pick the one with the best score, output this rule as rule number two and apply it on WL1 to get WL2 , etc. The scoring function is defined as follows: If the rule R has the template:

if Trigger then change tag X to tag Y

and w is a word in WLi with current tag X satisfying the trigger condition, then R gets the score P(Y|w) - P(X|w) for the word w. The total score for R is then obtained by adding all the 'word scores'.

score(R) := Σ P(Y|w) - P(X|w)

where the sum runs through all w in WLi with current tag X, satisfying the trigger condition. If the rule R has the template:

if Trigger then change current tag to tag Y and w is a word in WLi satisfying the trigger condition, then R gets the score P(Y|w) - P(Current tag of w |w) for the word w. The total score for R is then obtained by adding all the 'word scores'. **score(R) := Σ P(Y|w) - P(Current tag of w |w)**

where the sum runs through all w in WLi , satisfying the trigger condition. Note that the score the rule R gets for w always is of the form P(new tag|w)-P(old tag|w). A positive score means that the new tag is more likely than the old tag while a negative score means that the new tag is less likely than the old tag. The trigger condition is tested using bigwordlist and bigbigramlist, and the estimated probabilities are computed from the frequencies in smallwordtaglist., in the next chapter we are going to discuss Results.

CHAPTER 6
EXPERIMENTAL RESULTS

EXPERIMENTAL RESULTS

6.1 Output Screenshots

Figure 6.1.1 Home page

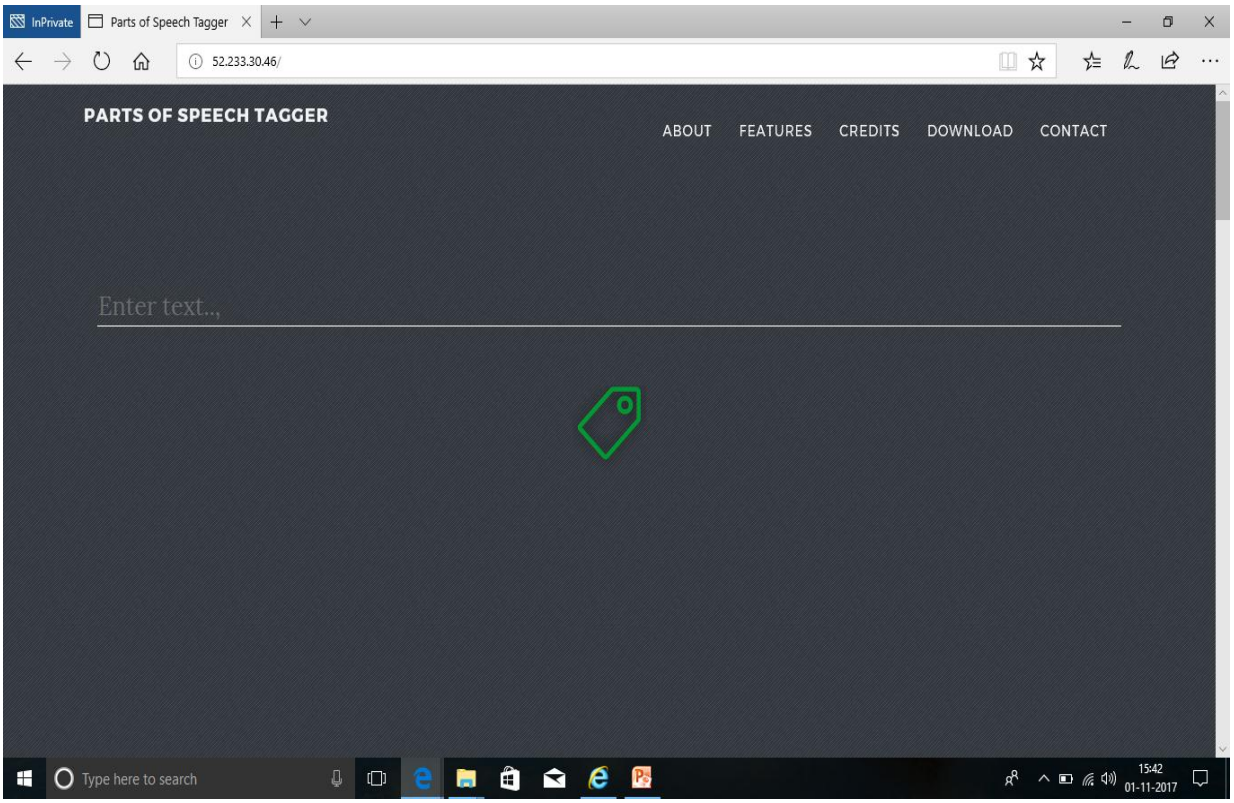


Figure 6.1.2 Input screen

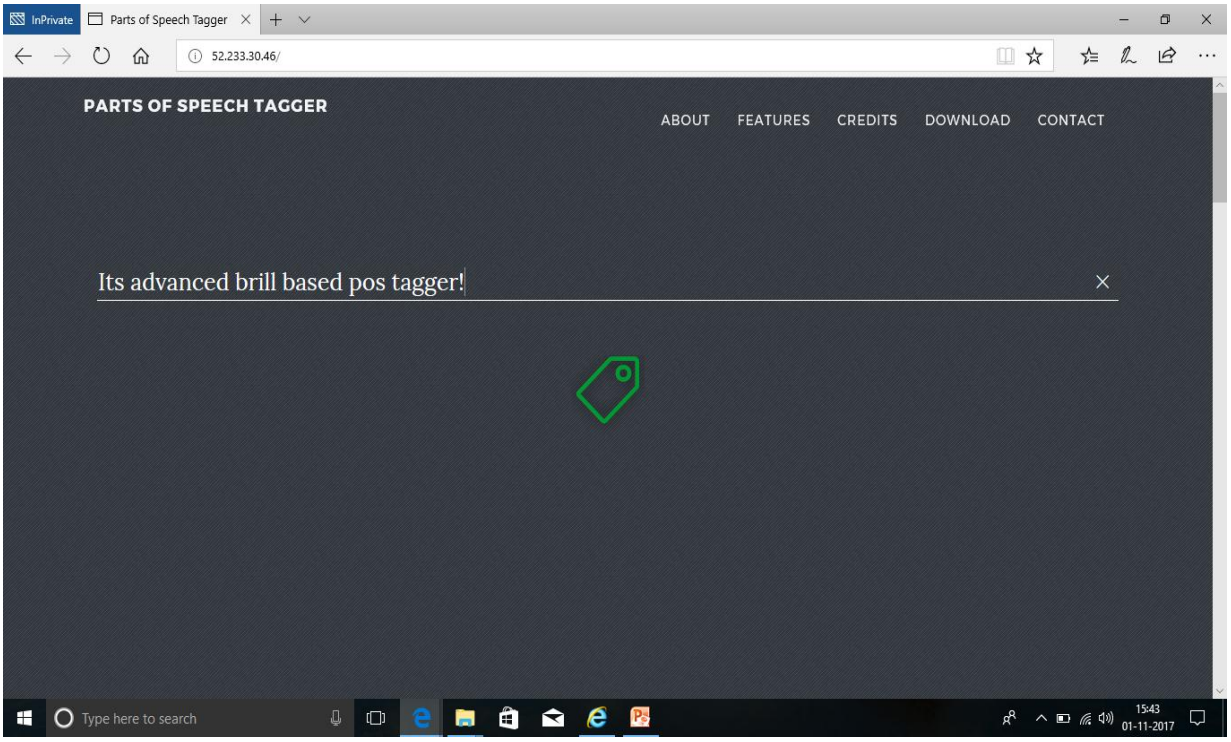


Figure 6.1.3 Output screen

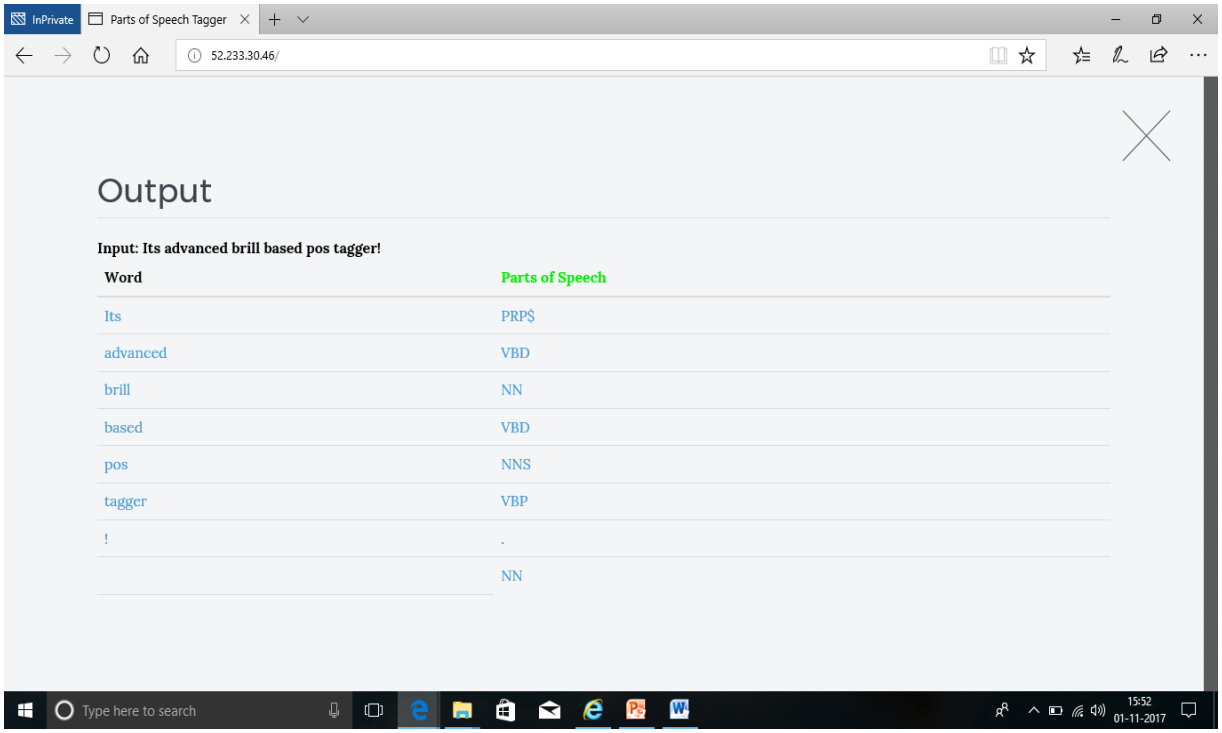


Figure 6.1.4 Mobile welcome screen

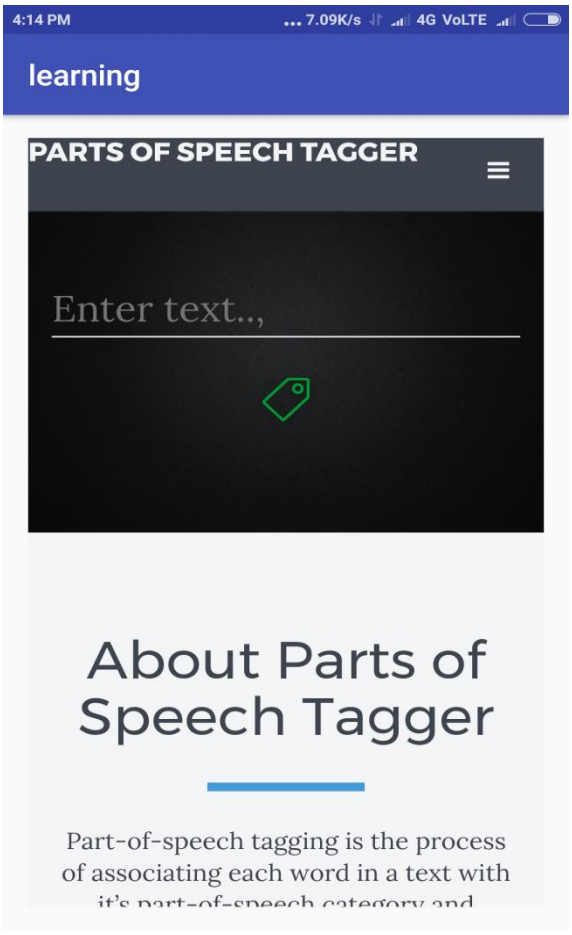


Figure 6.1.5 Mobile Output screen

6.2 Performance Metric

precision (P): Precision is the fraction of the correct tags generated by the POS to the total number of tags generated.

Precision (P) =Correct answers/answers produced

Recall(R): Recall is the fraction of the correct tags generated by the POS to the total number of correct tags.

Recall (R) = correct answers/ total possible correct answers.

F-Score: F-score is the weighted harmonic mean of precision and recall.

F-Measure = (ß2 ß2R+P).

ß is the weighting between precision and recall and typically ß=l.

F-Measure = (ß + 1) PR/ (ß R + P)

According to our observations, without window size, organizations identification gives very less performance. In this study we have taken tokens only, that is, the sentences are split into tokens using space and some predefined words. With this model we have achieved f-score of 95.18%.

CHAPTER 7
JOURNAL

JOURNAL
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Advanced Mechanism on Brill Tagger-English

Maduru Sadak Pramodh[1], Dr. M. Humera Khanam[2] , A. Khudhus [3]

[1]Department of Computer Science and Engineering, SV University College of Engineering, Tirupati, India
[2]Department of Computer Science and Engineering, SV University College of Engineering, Tirupati, India
[3]SV University, Tirupati, India

ABSTRACT

A part of speech labelling is the measurable depiction which is the procedure of partitioning each word with its grammatical feature kind. The achievement for Swedish strategy is portrayed by this paper in light of Brill Method. In this paper we are related with the client entered words with their related part of speech which are open in the corpus and furthermore we discover the part of speech of non-exist words by an event of words in an application which are entered by the client. Our technique is reasonable when the tag of a word is known for certain and is introduced with help and enhance the precision of order by giving a solid secure begin around which to tag.

Keywords : Part of Speech, Brill Method, Corpus, Non-existent words.

I. INTRODUCTION

In ordinary dialect rules, grammatical form is a class of words which have comparative syntactic properties. Current English language structure is the after effect of a slow change from an ordinary Indo-European ward checking design with a rich inflectional morphology and generally free word arrange, to a for the most part logical example with little expression, a genuinely settled Subject Verb Object (SVO) word organize and an intricate linguistic structure. Present day English depends more on helper verbs and word arrange for the declaration of complex tenses, viewpoint and mind-set and also uninvolved developments, interrogatives and some refutation. Regardless of discernible variety among the accents and lingos of English utilized as a part of various nations and areas as far as phonetics and phonology and now and again additionally vocabulary, punctuation and spelling English speakers from around the globe can speak with one. All in all Corpus is a substantial gathering of writings. It is a gathering of composed or spoken material whereupon a word investigation is based .The plural type of corpus is corpora. The corpus might be made out of composed dialect or talked dialect or both. The Brill tagger has several advantages that we propose these tagger when compare to other taggers. First the source code is distributed; this is rare at most other taggers are only distributed in the executable format. Second, the simplicity of the transformation based on the learning approach makes it possible for us to both understand and modify the process which meets our needs. And finally the tagger is accurate and it achieves accuracy without fail.

II. RELATED WORK

Brill Tagger Algorithm:

The Brill tagger is a productive method for grammatical feature labeling. It was spoken to and developed by Eric Brill in his 1993 Ph.D speculation. It is commonly outlined like blunder drove change based tagger. It is a sort of managed realizing, whose rationale is to constrict blunder; and, a change based strategy, inside the feeling that a tag is allocated to

each word and modified by employing aclassification of predefined rules.

In the change technique, if the word is known to the corpus, it initially doles out the predominant consistent tag, or if the word is obscure, it innocently appoints the label thing to it. Applying again and again these tenets, consistently changing the mistaken labels, a very high exactness will be accomplished. This approach guarantees that significant data, for example, the syntactic development of words is used in a programmed labelling process. The system of adapting such standards is typically related to as Transformation - Based Learning (TBL). Interestingly, stochastic strategies, for example, those help Hidden Markov Models would perhaps aggregate an accumulation of contingent probabilities got from n-grams of labels.

Although both straightforward and more confused stochastic taggers can be achieved appallingly high correctness's once they have doled out POS labels, they do not have leeway that control based taggers have, as stochastic taggers don't contain any unequivocal comprehensible principles, however only one thing like at least one generous likelihood lattices. Administer based taggers, on the antithetical hand, will just present the standards they use inside the labelling strategy amid an unmistakable arrangement. This straightforwardness is of extra incentive for dialects for which assets are scanty, as this empowers for an extra straight-forward investigation of the standards acquired.

Rule Based Tagging:

A control based tagger which executes and in addition taggers in view of probabilistic models. The rule based tagger conquers the restrictions regular in control based ways to deal with dialect preparing: it is hearty and the principles are consequently procured. Furthermore, the tagger has many favourable circumstances over stochastic taggers, including: an immense diminishment in put away data required the perspicuity of a little arrangement of important principles rather than the substantial tables of insights required for stochastic taggers, simplicity of finding and actualizing changes to the tagger and better movability from one label set or corpus type to another.

Administer Based labelling is the most established approach that utilizations written by hand manages for labelling, Rule construct taggers depends with respect to word reference or dictionary to get conceivable labels for each word to be labelled. Manually written tenets are utilized to sort the right label when a word has more than one conceivable tag. Vagueness will be kept away from by dissecting the highlights of that word, its previous word, its following word and different viewpoints. On the off chance that the past word is article then the word being referred to must be Noun. This data is coded as tenets. The guidelines might be setting design principles or normal articulations incorporated into limited state automata that are crossed with ambiguous sentence description.

III. PROPOSED SYSTEM

There are two parts of a change: a revise administer and an activating domain. The modify manage says once it should be done (e.g. change the class from A to B) and along these lines activating condition says when it ought to be done (e.g. on the off chance that the former specimen is of class C). Change based learning is utilized in numerous different zones and has turned out to be horrendously successful approach in the weld of common dialect preparing.

It gives various designs to various shape factor contraptions each trade will perform in cloud condition and in instantly recognizable stage the result is differentiated and the Stanford NLP Library and improves machine precision. In next part, we will inspect system examination.

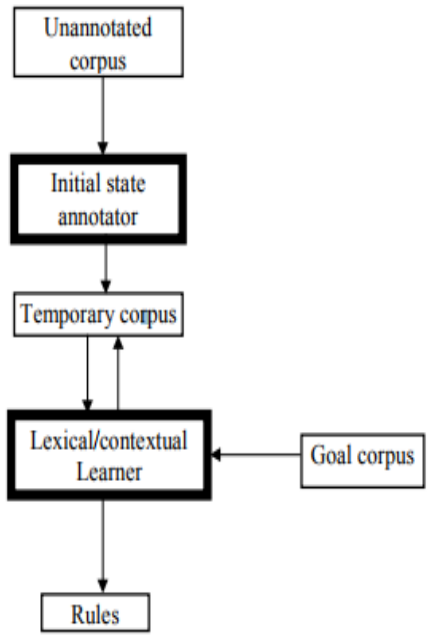
Part of Speech Tagger:

The execution of grammatical form tagger is begun by building a dictionary, wherever the grammatical feature of a word can be found. Tragically many words are questionable and each word will have numerous classifications. For instance, the word note will be either a thing or a verb. It is the object of the grammatical form tagger to determine these ambiguities, by misusing the setting of the word.

Another issue is the treatment of words that haven't any sections inside the vocabulary. There are fundamentally two ways to deal with grammatical feature labelling: control based labelling and stochastic labelling. This paper portrays an execution utilizing the oversee based approach, wherever the establishments are created utilizing change based learning.

Transformation-based error-driven learning is a machine learning strategy normally utilized for classification issues, where the objective is to dole out classifications to a gathering of tests. An underlying classification is made by utilizing a basic algorithmic rule. In each emphasis the present classification is contrasted with the best possible classification and changes are created to cure the mistakes. The yield of the algorithmic manage will be a rundown of changes which will be utilized for programmed classification, alongside the underlying more tasteful. There are two parts of a change: a revamp control and an activating domain. The modify manage says once it should be done (e.g. change the class from A to B) and subsequently activating condition says when it ought to be done (e.g. in the event that the previous example is of class C). Change based learning is utilized in numerous different regions and has ended up being dreadfully fruitful system in the weld of common dialect preparing.

The general system of Brill's corpus-based learning is gathered Transformation-based Error-driven Learning (TEL). The name mirrors the very certainty that the tagger is depended on changes or controls and learns by distinguishing blunders. For the most part, the TEL starts with an unannotated message as contribution after that goes through the initial state annotator'. It allows labels to the contribution in some heuristic way. The yield of the underlying state annotator could be a brief corpus, which is a short time later distinguished with an objective corpus, i.e. the legitimately clarified preparing corpus. For each time the transitory corpus is gone through the student, the student produces one new manage, the single decide that enhances the clarification the foremost contrasted and the objective corpus and replaces the impermanent corpus with the examination that outcomes once this lead is connected to it. By this strategy, the student delivers a requested rundown of tenets. Blunder driven learning module in Brill's tagger (information set apart by thin lines).



Figuer 1. Transformation Based Event driven learning structure

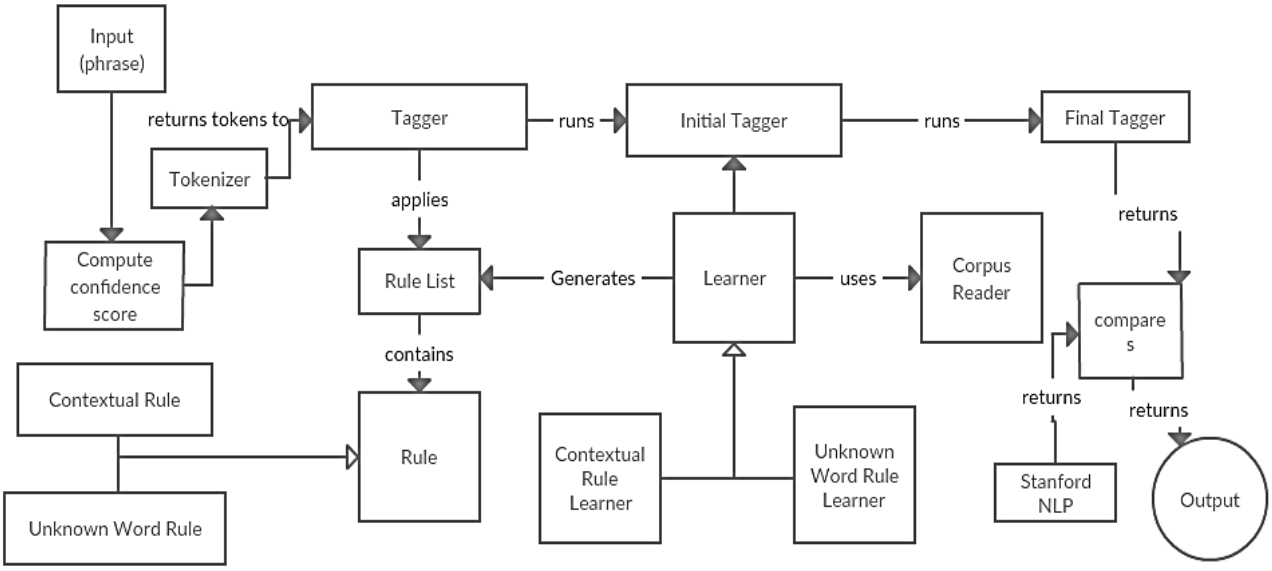


Figure 2. System Architecture

Initial Tagging:

In the Brill tagger starting labelling session words are allotted POS labels in light of non-relevant choices. To start with, every last word is apportioned to the chief regular tag of that word inside the preparation corpus, as demonstrated as follows.

(1)	Every	minute	counts
	DT	**NN**	**VBZ**
(2)	Every	minute	Details
	DT	***NN**	**NN**

The tagger each time picks the premier regular tag to a word; this winds up in blunders of the kind that which can be viewed inside the case above, where minute is erroneously marked as a thing in (2),

Wherever it should be a modifier. Words that don't appear to be found in the corpus are dealt with independently and can be allotted labels relying upon the highlights of words. Now and again words could be marked relying upon their additions (or option dialect subordinate uncovering signs). Words not reasonable to any class after this procedure are doled out to the most continuous tag inside the corpus. Since no pertinent data is utilized amid the each stage, a few words are conceivable to be labeled erroneously.

Transformation-Based Tagging

After the finish of introductory stage, the relevant mistake driven tagger is connected. This tagger makes an endeavor to utilize change manages in order to lessen the measure of labeling mistakes. Considering that principles revise the slip-ups made by the underlying labeling, they are regularly named as patches. These tenets are accepted consequently and are made to suit one in every one of the few decided setting subordinate rule formats.

The above procedure is shown in the figure 3

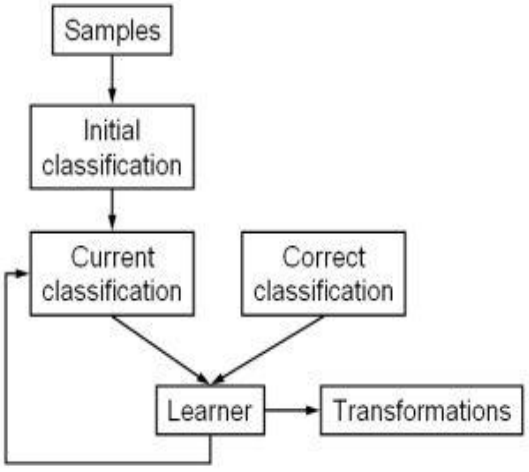


Figure 3. Transformation Based Tagging

Algorithm

The contextual transformation template proposed by brill:

1. The preceding/following word is tagged with A (PRETAG/NEXTTAG)
2. Anyone of the two preceding/following words arelabelled with A (PREV1OR2TAG/NEXT1ORTAG)
3. One of the three preceding/following words are tagged with A. (PREV1OR2OR3TAG/NEXT1OR2OR3TAG)
4. The preceding word is tagged with A and the following word is tagged with V. (SURROUNDTAG)
5. The preceding/following two words are tagged with A and V. (PREVBIGRAM/NEXTBIGRAM)
6. The word two words before/after are tagged with A. (PREV2TAG/NEXT2TAG)
7. The present word is Z. (CURWD)
8. The preceding/following word is W. (PREVWD/NEXTWD)
9. One of the preceding/following words are W. (PREV1OR2WD/NEXT1OR2WD)
10. The word two words before/after are W. (PREV2WD/NEXT2WD)
11. The present word is A and the preceding word is V. (LBIGRAM)
12. The present word is V and the following word is A. (RBIGRAM)
13. The present word is V and the preceding/following word is tagged with A. (WDPREVTAG/WDNEXTTAG)
14. The present word is V and the word two words before/after is tagged with A. (WDAND2BFR/WDAND2TAGAFT)

In the calculation we at first instate the qualities that are the way toward appointing the marks (labels) in view of their likelihood for each word (for instance, pooch is more regularly a thing than a verb). At that point patches will be computed by means of principles that right (likely) labelling mistakes made in the introduction stage:

The System Architecture is shown in the figure 2.

Initialization:

Known words (in phrasing): allocating the most incessant label related to a type of the word Unknown word.

Learner:

Learner is the main class of the learning program and contains the general learning algorithm. It is an abstract class that requires the subclasses to implement some parts of the algorithm.

Contextual Rule Learner:

Learner and is responsible for the contextual rule learning.

Unknown Word Rule Learner:

Unknown Word Rule Learner is also a subclass of Learner and is responsible for learning the unknown word rules.

Rule:

Rule is the superclass of all rules. It contains the abstract methods instantiate, predicate, evaluate and apply. Contextual Rule is the super class of all contextual rules. UnknownWordRule is the super class of all unknown word rules.RuleList is a class for maintaining a list of rules.

Corpus Reader:

Corpus Reader is responsible for extracting words and tags from the manually explained corpus.

Tagger:

Tagger is the main class of the tagging program. It takes an untagged text as input and produces a tagged text as output.

The tagging is done in three steps.

1. Initial tagging
2. Application of unknown word rules
3. Application of contextual rules

Tokenizer:
Tokenizer is used by the Tagger to divide the input text into tokens.

Word Dictionary:
Word Dictionary contains all words of the training corpus. It is used for finding the most suitable tag for a word, investigating if a word exists and searching for prefixes or suffixes of words.

Tag Dictionary
Tag Dictionary is responsible for the translation between the string and integer representation of the part-of-speech tags.

Initially input phrase is processed and finds a confidence score of a language that resolves more ambiguity. After final tagger results are compared with Stanford NLP library results.

IV. EXPERIMENTAL RESULTS

Precision (P): Precision is the portion of the correct tags generated by the POS to the total number of tags generated.
Precision (P) =Correct answers/answers produced
Recall(R): Recall is the fraction of the correct tags generated by the POS to the total number of correct tags.

Recall (R) = correct answers/ total possible correct answers.
F-Score: F-score is the weighted harmonic mean of precision and recall.

F-Measure = (ß2 ß2R+P).
ß is the weighting between precision and recall and typically ß=l.
F-Measure = (ß + 1) PR/ (ß R + P)

According to our observations, without window size, organizations identification gives very less performance. In this study we have taken tokens only, that is, the sentences are split into tokens using space and some predefined words. With this model we have achieved f-score of 95.18%.
Score(R): number of errors corrected - number of errors Occurred.

V. CONCLUSION

This work has been displayed how Eric Brill's administer based POS tagger, which consequently procures rules from a preparation corpus, which works by depending on change based blunder driven learning, for upgrading the outcomes the tagger is exceptionally powerful and this is extremely fruitful and quicker when contrasted and the corpus which contains the 23,000 words, extremely poor outcomes are created by the current framework when the words are obscure in the corpus. The last consequences of the tagger are processed to 95.18% and 92.14% with a shut and open corpus individually.

Additionally, here we are utilizing a vast label set stamping inflectional highlights of a word in the preparation and grouping process which enhances the precision.

VI. REFERENCES

[1]. Guaranteed pre tagging for the Brill Tagger saif Mohammad and Ted Pedersen. university of Minnesota
[2]. Part of speech tagging using the Brill Method Maria Larsson and mans norelius.
[3]. Genetic Algorithms in the Brill Tagger Moving Towards language independence. Johannes Bjerva.
[4]. Brill's POS tagger with extended Lexical templates for Hungarian beata Megyesi.
[5]. Brill, E. 1992. A Simple Rule-Based Part of Speech Tagger. In Proceedings of the DARPA Speech and Natural Language Workshop. pp. 112-116. Morgan Kauffman. San Mateo, California.

[6]. Brill, E. 1994. A Report of Recent Progress in Transformation-Based Error-Driven Learning. ARPA-94.

[7]. Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in POS Tagging. In Computational Linguistics. 21:4.

[8]. Brill, E. & Marcus, M. 1992. Tagging an Unfamiliar Text with Minimal Human Supervision. In Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language. 1992.

[9]. Megyesi, B. 1998. Brill's Rule-Based Part of speech Tagger for Hungarian. Master's Degree Thesis in Computational Linguistics. Department of Linguistics, Stockholm University, Sweden

[10]. B. Revathi, Dr.M.HumeraKhanam Hindi to English part of speech Tagger By using CRF Method

[11]. A. Ragini, Dr. M. HumeraKhanam "Machine Translation System for English to Telugu Language: A Rule Based Complex Sentence Simplification" North Asian

[12]. International Research Journal of Sciences, Engineering &I.T.Smt. M. HumeraKhanam, Prof. K. V. Madhumurthy "Dependency parsing for Telugu" International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622, Vol.1, Issue 4, pp.1751-1754, Nov-Dec 2011, ACM, ISO 3297

CERTIFICATE
OF
PUBLICATION

ISSN : 2456-3307 website : www.ijsrcseit.com

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Scientific Journal Impact Factor = 4.032

IJSRCSEIT/Certificate/Volume 3/Issue 1/1548 18 January 2018

CERTIFICATE OF PUBLICATION

This is to certify that **Maduru Sadak Pramodh** has published a research paper entitled "Advanced Mechanism on Brill Tagger-English" in the International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Volume 3, Issue 1, January-February-2018

This Paper can be downloaded from the following IJSRCSEIT website link
http://ijsrcseit.com/CSEIT11726269
IJSRCSEIT Team wishes all the best for bright future

Editor in Chief
International Journal of Scientific Research in Computer Science, Engineering and Information Technology

UGC Approved Journal [Journal No : 64718]

CONCLUSION AND FUTUREWORK

CONCLUSION AND FUTURE WORK

This is the end of the technical material in this book. We've explored the big ideas of Artificial Intelligence and NLP. As a review, you might enjoy reading Chapter 1 to see how the formerly mysterious problems are solved now make sense.

We have proposed handling of Unknown Words Unfortunately, most of the rules for unknown words turned out to give poor results. The optimization of the original tagger was very successful, resulting in a running time almost 15 times faster when using a training corpus of 23000 words.

The final result of our work can be summarized in the figures showing the accuracy of the tagger. The resulting accuracy was computed to 95.18 % and 92.94 %, with a closed and open vocabulary respectively.

This application is deployed in Microsoft Azure Cloud and achieved client server model for business and research purposes. This dissertation work is intended for people whose main interest is in something else and as preparation for those who intend to continue in NLP Applications. The final chapter is directed mainly at the latter group, to tell them what to expect. We confess, though, to the hope that some of the former may have enjoyed this experience enough to be lured into further study. If you're in that category, you'll be particularly interested in my work please scan QR code and part in my work.

For future extenders of this work, the baseline will presumably be the focal point of attention. A way to further increase the accuracy of the tagger would be to introduce the lexicalized rules also suggested by Brill. However, according to Brill, these rules only improve accuracy slightly (0.2 percentage points). Handling of unknown words may improve using supervised learning methods and train corpus accordingly.

REFERENCES

REFERENCES

1. Transformation-Based Learning by Christian Siefkes.
2. Brill Tagging on the Micron Automata Processor by Keira Zhou; Jeffrey J. Fox; Ke Wang; Donald E. Brown.
3. Part-of-Speech Tagging Using the Brill Method Maria Larsson and M˚ansNorelius.
4. B. Revathi, Dr. M. HumeraKhanam "Hindi To English Part Of Speech Tagger By Using CRF Method".
5. Eric Brill. 1995. Transformation-Based ErrorDriven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging.
6. Brill, E. 1993b. Transformation-Based Error-Driven Parsing. In Proceedings of the Third International Workshop on Parsing Technologies. Tilburg, The Netherlands.
7. Brill, E. 1994a. A Report of Recent Progress in Transformation-Based Error-Driven Learning. ARPA-94.
8. Brill, E. 1994b. Some Advances in Rule-Based Part of speech Tagging. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, Wa.
9. Brill, E. 1995a. Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. Very Large Corpora Workshop. 1995.
10. Brill, E. & Marcus, M. 1992a. Automatically Acquiring Phrase Structure Using Distributional Analysis. DARPA Workshop on Speech and Natural Language. 1992.
11. Brill, E. & Marcus, M. 1992b. Tagging an Unfamiliar Text With Minimal Human Supervision. In Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language. 1992.
12. Brill, E. &Resnik, P. 1994. A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation. In Proceedings of Fifteenth International Conference on Computational Linguistics COLING-1994. Kyoto, Japan.
13. F. Jelinek. 1997. Statistical Methods for Speech Recognition. MIT Press.
14. NavneetGarg, Vishal Goyal, SumanPreet. "Rules Based Part of Speech Tagger" in the proceedings of COLING 2012:, Mumbai, December 2012.