
University of Eastern Finland

School of Computing

Master's Thesis

# Using Part of Speech for Analyzing Language

Asma Komal

29th of June, 2019

# ABSTRACT

English is the most commonly spoken language in the world, its vocabulary has increased with the rise of technology and meanwhile it has become more complicated for the speech (language) processing tasks. Every word in this language has a meaning and function according to which we can specify the categories based on how these words have been used in speech. Words having similar grammatical characteristics are labeled with the same part of speech category.

Part of speech (POS) tagging is assigning a tag or label to each word in the text. This is done based on the meaning and context of each word relative to its adjacent words in the sentence. POS tagging is very useful for information retrieval, classification purposes and for a variety of natural language processing tasks.

Originally, POS taggers were designed for the *newswire* (a service to transmit news to media or public) text and not for the tweet text (from Twitter). Recently, different approaches have been proposed to handle Twitter data and provide robust POS tagging methods. The goal of this thesis is to provide a detailed and up-to-date study of state-of-the-art POS tagging approaches in the scope of Twitter data and English language.

**Keywords:** part of speech, tagger, performance, Twitter, natural language processing, POS tagging.

# ACKNOWLEDGMENTS

I am obliged to the University of Eastern Finland for accepting me in the IMPIT program and giving me a chance to learn under the supervision of the best teachers. I am thankful for being a part of such a multicultural and learned group of people in IMPIT 2017.

I am grateful to my supervisor, Professor Pasi Fränti, for his endless guidance and support. Under his supervision I have improved my presentation and technical writing skills along with the deep analysis and problem solving techniques. I would like to thank him for giving me the confidence with his advices at every step of my research and work.

I am thankful to Professor Mikko Laitinen for his guidance and support in my research. I am also very thankful to my project researcher Masoud Fatemi who helped me in all the experimental problems and answered all of my technical questions.

I would like to express my deepest gratitude to my husband who was always there by my side and helped me continuously at every step of my studies. I am also grateful to my family who have shown me great moral support and encouraged me to do my best.

In the end, I would like to dedicate this thesis to my mother. Her prayers and struggles have made me what I am today.

# TABLE OF CONTENTS

# 1 Introduction

We humans, always communicate using a common language in which we can understand each other. *Language* is defined as a set of words and a common system to use those words – which is adapted by the people living in the same area, geographical region or a nation (Webster, 2019).

*Speech* is another term for spoken language, it is defined as an ability to express our feelings and thoughts using voice (Collins, 2012). People from different places of the world have their own different languages, accents and dialects such as Finnish language, Swedish language, Arabic language. Of all the languages in the world, English is the most common international language because nowadays people belonging from different countries mostly communicate in English.

*Grammar* is the base of any language system designed by the people. It has a set of rules defined for correctly reading, writing and speaking the words. Grammar defines the use, function and classification of words in speech or text (Sayce, 1911). The process of classifiying the words according to their grammatically correct use and function in the sentence is called *part of speech* (POS). (Nordquist, 2019)

*POS tagging* is introduced by linguistics by which they assign a tag to each word in the text. In a sentence, every word is assigned a tag in context of its adjacent words. Figure 1 shows an example of such a sentence in which the words are tagged with their POS tags. Generally, eight POS tags are considered as the main tags to determine the use of a word in a sentence. For example, the words can be used as nouns, pronouns, verbs, adverbs, adjectives, conjunctions, prepositions and interjections in a sentence (D'Souza, 2018). The brief definitons of POS tags are shown with examples in Table 1.

**Table 1**: Eight basic POS tags with examples (Francis, 2019)

| Part of speech | Definition | Examples |
|---|---|---|
| **Noun** | Names of person, place, thing or idea. | Mona, tree, Finland, love, home |
| **Pronoun** | Replaces a noun | I, me, we, ours, he, she, her, they, them |
| **Adjective** | Describes a noun | Good, huge, black, attractive |
| **Verb** | Action or state | To be, have, do, sing, cook, work, play |
| **Adverb** | Describes a verb, adjective or adverb | Loudly, quickly, easily, badly, very, too |
| **Preposition** | Links a noun or pronoun to another word | To, on, after, at, through, from |
| **Conjunction** | Joins words or group of words (clauses or sentences) | And, either, or, neither, nor, but |
| **Interjection** | Expresses strong feelings or emotions | Oh! Wow! Great! Oops! Ouch! Hey! |

The POS categories described in Table 1 also have subcategories. For example, nouns have types like common nouns, proper nouns, abstract nouns; verbs have types like main verbs, auxiliary verbs; adverbs have comparative and superlative forms.

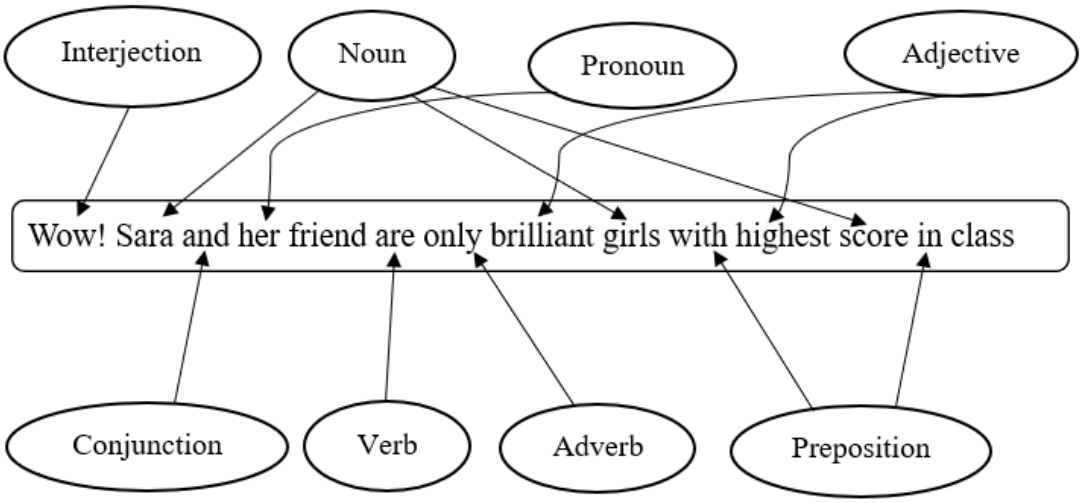



**Figure 1**: An example of a sentence tagged with eight basic POS tags.

POS tagging is done with the help of *tagsets*, which is a list of POS tags. An example is *Penn Treebank* tagset for English language developed by Beatrice Santorini in 1989. It consists of tags shown in Table 2. These tags are used to specify a category/class for each word in a text *corpus*, where a large collection of text is called a corpus.

POS tagging is an important component in the development of *natural language processing* (NLP). NLP is an emerging field of computer science and it is considered as a form of *artificial intelligence.* NLP creates a way of interaction between humans and computers. Computers use NLP to understand and process human language in order to draw insights from human created texts. These days NLP is behind the idea of providing customized advertisements and predictions to the users on their devices. For example, when we write a text message in our mobile phones we see word suggestions related to what we are typing or according to the words we frequently use in our daily conversations. All this is being done by NLP.

**Table 2**: List of 36 POS tags in Penn Treebank tag set (Marcus et al., 1993).

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

The importance of POS tagging can be determined by the fact that it is the baseline of many NLP applications pipelines. Examples of how it is used in different applications are listed below.

- *Sentiment analysis*, it analyzes if the user has commented a negative or a positive feedback

- *Spam detection,* it detects irrelevant search results that appear in email systems

- *Text to speech conversion,* it assigns different tags to the same word according to the different contexts used for that word in the same text

- *Topic tagging,* it generates topics for articles and blogs automatically

- S*peech recognition,* it creates subtitles of movies and tv shows

- W*ord sense disambiguation,* it distinguishes the meaning of a word within the scope of a phrase (Moreno-Monteagudo et al., 2006)

- *Information retrieval*, *parsing* (Watson., 2006) and *semantic classification* (Buitelaar et al., 2005), it analyzes the language structure in the text.

The above examples of applications depict that POS tagging is a fundamental step in the development of all those applications in which the understanding of language and context is needed to help computers interpret what has been said in a certain text given as an input to the computer. POS tagging simplifies the NLP task being solved by performing as a pre-requisite step in the application pipelines. Such an NLP application pipeline is shown in Figure 2 in which POS tagging is used to classify news text. It classifies the articles into different categories of sports, politics and entertainment with the help of tagged words.


**Figure 2:** A simple pipeline of news text classification (Belinić. 2018).

A POS tagger is the software that performs the tagging on a text corpus. POS taggers fall into two groups:

- Rule-based POS taggers
- Stochastic POS taggers

A *rule-based* tagger assigns tag to a word on the basis of the already defined rules in context of that word. For example, the rule-based tagger follows a rule from tag dictionary that if a word has the preceding word as an *article*[1] by grammar, then the word in question must be tagged as a *noun* (Jurafsky and Martin, 2005). Generally, the rule based tagger classifies the words in a corpus using either these hand-written rules or a *tag dictionary* using the context of a word with respect to its preceding and following words in a sentence. A tag dictionary is simply a list of several hundred words and their possible tags (Moore, 2015). The size of a tag dictionary can vary from few hundred to several thousand words and their corresponding tags.

On the other hand, the *stochastic* taggers assign tags to known words according to the structure and formation of the words analyzed in the text. Stochastic tagger is trained on a given training text data. It analyzes the language and determines the grammar and pattern of words in training text automatically. Then it applies the

[1] https://study.com/academy/lesson/what-are-articles-in-english-grammar-definition-use-examples.html

learned knowledge of grammar and language on the test data to assign tags. If an unknown or ambiguous word is encountered then the tagger searches for the most frequent tag found with that particular word in the training data and assigns that tag to the word. This is called the *word frequency approach*.

There are many other approaches and different models for stochastic POS tagging. One such model is *n-gram* (Jurafsky and Martin, 2018) which means a sequence of *n* words and is also known as *tag sequence probability* model. According to the n-gram approach, a word is given a tag which has the highest probabilty to occur with the *n* previous tags in the given text, and *n* can refer to one, two, three or more tags in the sequence. In other words, the context of the words is considered in order to assign tags and the language structure is analyzed to predict which word is most likely to follow the given word in the text data.

*Hidden Markov Model* (HMM) is an example of a stochastic POS tagger (Lee et al., 2000). It combines the two approaches of word frequency measurement and tag sequence probability (n-gram approach). This model suggests that the probability of choosing a tag for a given word depends on the previous words and their tags in the text.

Another effective tagging approach is a transformation based approach. *E. Brill's tagger* (Brill, 1994) is the most widely used tagger based on this approach. The Brill's tagger finds out the rules in a corpus by going through the training data of that corpus and applies those rules to the test data. According to Brill (1994), the tagger works on a transformation-based error-driven learning approach. According to this approach shown in Figure 3, the tagger takes raw text as an input and assigns tags to the words at the initial state. Then the tagged or annotated text is compared with the manually tagged text called the *ground truth*. After this, the text is passed through the *learner* which produces a new rule based on the ground truth and applies it to the text. Then it compares the text with the truth again until the best tagging accuracy is achieved. In this way, the learner transforms the tagged text again and again by learning through the errors found in the text at the previous step and produces a list of rules for the tagger.

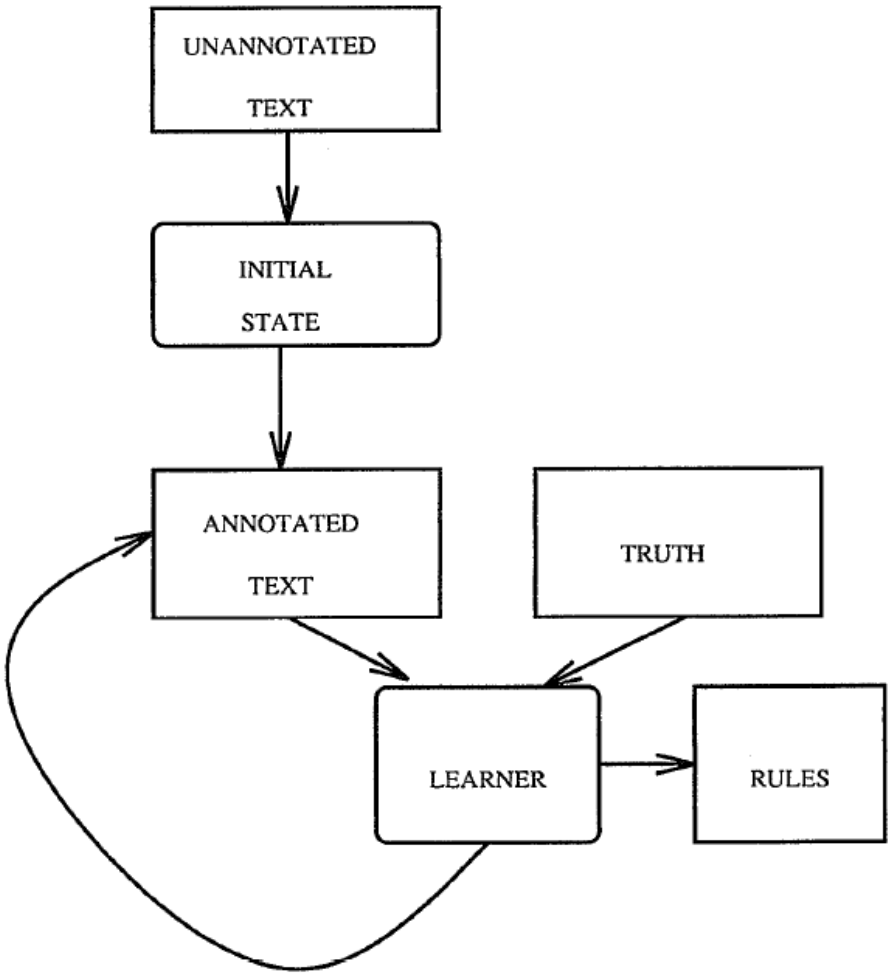



**Figure 3**: Transformation-based error-driven learning process (Jishan et al., 2016).

In this thesis, the goal is the comparison of some existing state-of-the-art POS taggers to highlight the impact of different text datasets on their tagging performances. We will also discuss the strengths and weaknesses of these POS taggers from the structural and architectural point of view to provide a comprehensive study on POS taggers. It includes the discussion of POS tagging models like HMM, hardware and software dependencies and available training data sets for each tagger. Experiments have been performed on a newswire corpus to check the performance of the taggers on a formal and grammatically correct text as well as on the conversational corpus to assess the tagging performance on a text which contains informal use of language and also has grammatical errors.

We also focus on highlighting the challenges and importance of POS tagging on *Twitter* dataset. Twitter[2] is a social networking site for people to communicate with each other in the form of short messages called *tweets*. It is a great source of information and a lot of analysis can be performed to extract useful statistics from the daily tweets, which is why linguistics have keen interest in processing the tweets data these days. Social scientists and linguistics are using the tweets to analyze the stock market behavior (Pagolu et al., 2016), predict the consequences of a political movement (Conover et al., 2011) or results of an election (Ramteke et al., 2016) and also to analyze the evolution in English language (Grieve et al., 2018). For this reason, we discuss POS tagging in detail for Twitter data (Section 2) and carry out experiments on conversational text data having similar characteristics as the Twitter text data.

[2] https://twitter.com/

# 2 POS Tagging for Online Chat Text

According to a recent research by Cooper, P. [2019], the number of people using Twitter has reached upto 326 million till date. These huge number of Twitter users post 500 million tweets every day, which means 5,787 tweets are produced every second. This data generated by Twitter is not in the form of a standard English text because it contains abbreviations, webpage links/email addresses, username mentions (@), hashtags (#), location tags, foreign words, emoticons, discourse markers (~) and also grammatical errors due to the restriction of 280 characters per tweet. An example of a tweet is shown in Figure 4. These type of special characters and grammatical errors present in the tweets make the POS tagging a challenging task because the terms and language in this data changes everyday, increasing the number of unknown words on such a vast scale.


**Figure 4**: An example of a tweet with grammatical errors and special characters[3]

Different approaches have been proposed to handle Twitter data and provide robust POS tagging methods (discussed in Section 2.1 and 2.2). The general architecture of

[3] https://twitter.com/ssempanyijr1997/status/1127091449491853312

a POS tagger is shown in Figure 5. A tagger takes raw text as an input and divide it into sentences. It then further breaks the sentences into words called *tokens*. Finally POS tags are assigned to these tokens to give tagged sentences as an output. However, these general POS taggers were designed to perform best for the *newswire* text produced by the news media services and usually contains grammatically perfect sentences without spelling errors. For this reason, different approaches were introduced to improve the general architectural design and tagging accuracy of existing POS taggers, discussed in detail in Section 2.2.

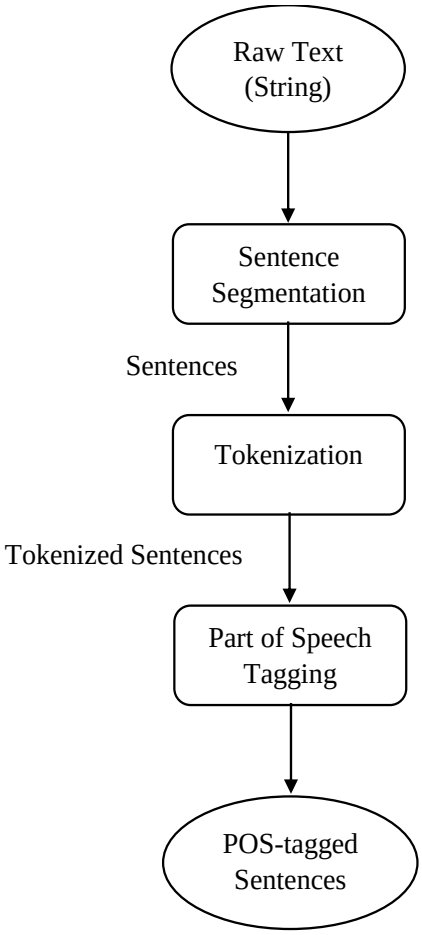
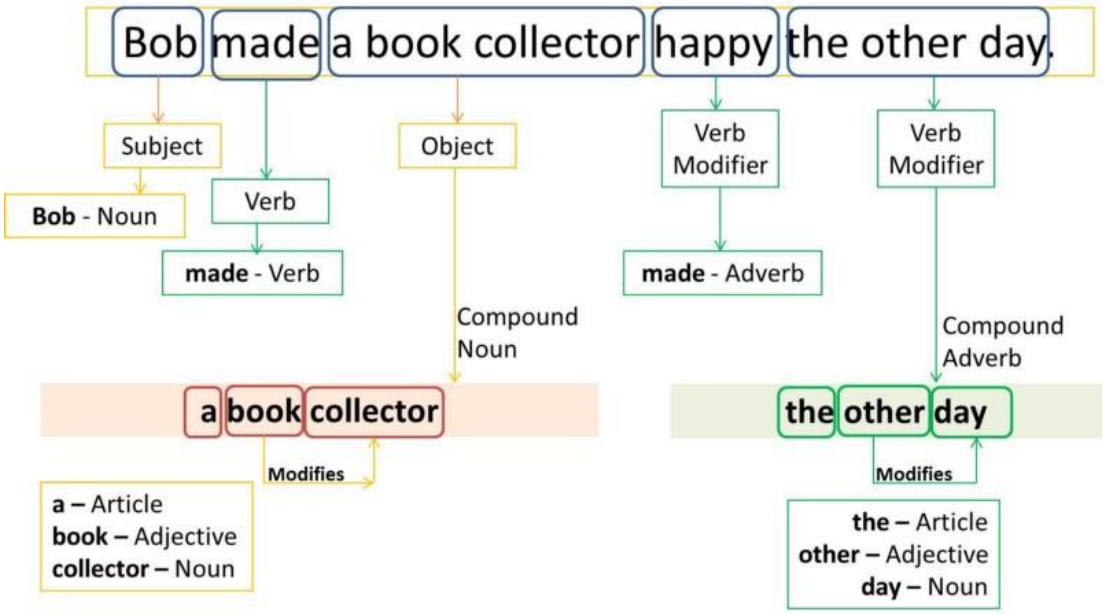



**Figure 5**: Simple pipeline architecture for a POS tagger (Bird et al., 2009).

Figure 6 shows an example of POS tagging process implemented on a sentence following the pipeline architecture in Figure 5. It shows a sentence segment and the tokenization of sentence into words. Then POS tags are assigned to the tokens and a POS tagged sentence is generated at the end.




Bob/*NN* made/*VB* a/*AT* book/*JJ* collector/*NN*

happy/*RB* the/*DT* other/*JJ* day/*NN*

**Figure 6:** POS tagging example of a sentence (Godayal, 2018).

## 2.1 Data Handling Approaches

A *data intensive approach* has been proposed by Derczynski et al. (2013) to handle Twitter data in context of POS tagging for English. They have suggested methods to deal with missing words and errors in the data due to *tokenization*, use of slang/informal language, rare words, and spelling mistakes. Tokenization is a process of chopping a text document into words/tokens to make the data easy to be tagged. For examlple, Figure 7 shows a sentence and its tokenized version.

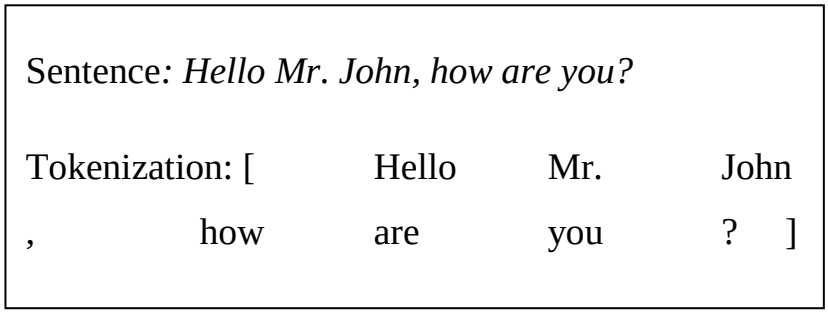



**Figure 7**: An example of a sentence and its tokenized version.

Jatav et al. (2017) have used linguistics-based rules to handle the data that has been tagged incorrectly. Tagged data is compared with the *ground truth* and the linguistics rules to find out incorrectly tagged words. They have suggested to divide the incorrectly tagged words as *critical errors* and *non-critical errors*. Critical error is the one in which the meaning of a sentence is changed due to misinterpretation of a word by giving it a wrong POS tag. These errors badly affect the overall language structure in the text. Non-critical errors do not affect the sentence structure significantly, the overall meaning of a sentence does not change, and the POS tag does not fall out of its category in this case.

Gimpel et al. (2011) have proposed to develop a POS tagset having special features to efficiently handle the Twitter related data. The feature set includes the following

- *Twitter orthography* attribute which handles the at-mentions (@), hashtags (#) and webpage links (https://www.example.com/) in tweets
- *Frequently-capitalized tokens* attribute is to check capitalization patterns in the word; for example, *sMarT*
- *Traditional tag dictionary* to look for the standard glossary of tagged words such as Penn Treebank tagset (Table 2)
- *Distribution similarity* to find and categorize similar words together and choose tag for an unknown word by looking at its similar words. For

example, by looking at the suffix of the words like *softly* and *quickly*. Both words have the same suffix *-ly*, so such words can be categorized together.

- *Phonetic normalization* to handle the words used with many alternative spellings in the tweets such as *thanks, thanx, thankssss*

Owoputi et al. (2013) have used the work of Gimpel et al. (2011) to design a POS tagger with new large-scale distributional features (Brown et al., 1992). A typical POS tagger will assign POS tags to an informal conversational text as shown in Figure 6. Here, the tagger is unable to identify tags for some abbrevated words like *ikr* and *lolololol*. For this reason, Owoputi et al. (2013) created clusters of such rare words found in the chat data and assign correct tags to them. Their proposed new features distribute the words frequently occurring together into same groups as follows

- *Acronyms,* for example using *lol* as an expression of *laughing out loud* or *haha*

- *Interjections,* such as *ikr* is used as a short term for *I know, right*

- *Grammatical category variants,* such as *going to, trying to, gonna* are variants having similar meanings and used alternatively

- *Orthographic categories* to handle words like *so, sooo, soooo* or *happpy, happyyyy* in the same clusters

- *Emoticons category* to handle emojis

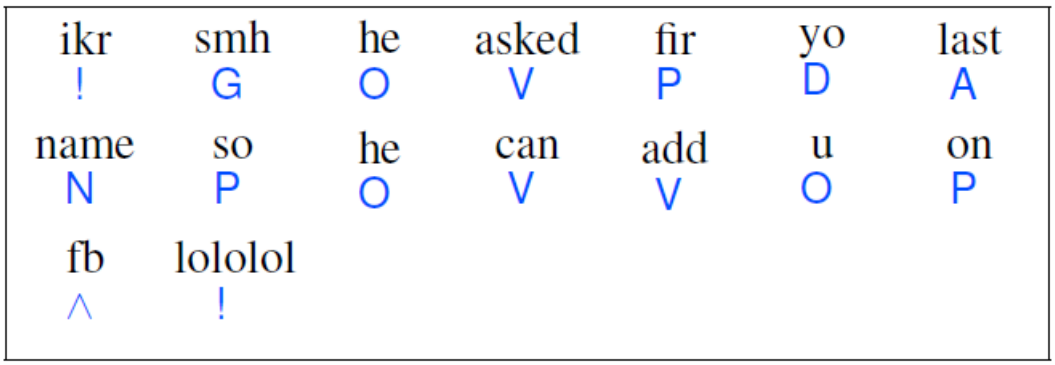



**Figure 8**: An example of an automatically tagged tweet. (Owoputi et al., 2013)

Since the Twitter data includes unusual and informal words there is a lack of large scale tagged data in this domain to train the POS taggers accurately. Gui et al. (2017) have suggested to use neural networks to handle this lack of training data challenge as well as other tweets related data challenges like phonetic terms. For example, *gr8* is used for "*great*"; abbreviations like "*the*" is shortened as *da*; a continuous addition of new words or slangs in tweets and use of emoticons.

Weerasooriya et al. (2017) have also tried to handle the similar conversational data challenges implementing methods proposed by Tregex (Levy and Andrew, 2006) and Penn Treebank (Marcus et al., 1994). Both these methods implement the concept of analyzing word context in a sentence which states that, the tag of a word is predicted in context of its neighbor words and their tags.

## 2.2 Approaches to Improve POS Tagging Accuracy

In past few years, many approaches have been introduced to improve the POS tagging accuracy and reduce the error rate in taggers. Supervised and unsupervised approaches of POS tagging are summarized as in Figure 9 by Guilder (1995). In a *supervised* approach, a pre-tagged corpus such as a tag dictionary is used to train a tagger and POS tagging is performed according to that pre-tagged training data. In an *unsupervised* approach, no such pre-tagged corpus is used. Instead the tagger checks the language structure in a given corpus and automatically learns to create word groups called *tag sets* and perform POS tagging based on its own tag set (Buchholz,

2002). In this section, some of these approaches are discussed which have been implemented by researchers to improve the POS tagging accuracy.

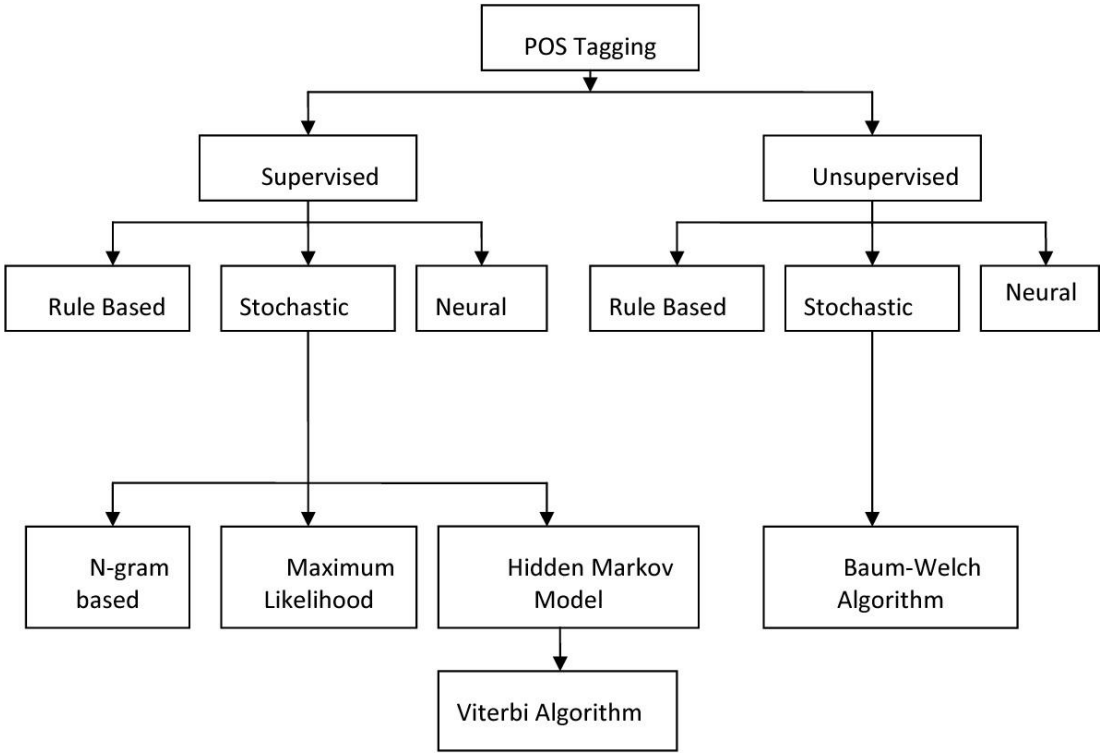



**Figure 9**: Classification of POS tagging approaches (Guilder, 1995).

Derczynski et al. (2013) have introduced a supervised model to improve the POS tagging accuracy over 90.5% for Twitter related chat data. They have also focused on error reduction techniques by analyzing the difference in tagging accuracies for chat text as well as news text. They reduced the error rate of taggers at ~5% (elevating the accuracy from 84.43% to 89.37%) by categorizing the errors found in tagged data and designed rules to handle each error category accordingly. For example, they have handled errors that occur due to tagging of unknown words which were not present in the training data using a normalization technique. In this technique, similar words related to an unknown word are searched in the training data to assign a similar tag to that unknown word such as the terms *yea*, *yeah*, *ya* and *yes* were treated as similar words and same tags suggested for these words.

Jatav et al. (2017) have introduced an unsupervised rule-based approach to improve the POS tagging accuracy of the existing state-of-the-art POS taggers by approximately 3%. They argued that the POS tagging errors generated by other taggers are due to reasons like the lack of enough training data or the occurrence of unknown words in test data. They have developed a set of linguistic rules to be applied on POS taggers in the training step. These rules helped in significantly improving the performance of taggers and handled challenges like grammatical mistakes, size and quality of the training data and rare words. Using their proposed set of rules they have designed a rule-based/statistical tagger called *rapid application generation engine* (RAGE) AI hybrid POS tagger, which achieves over 80% tagging accuracy. The tagger is preprocessed with a set of large scale language rules as shown in Figure 10 to assign tags to the words in text by following those rules. Then another set of POS correction rules is applied to check incorrect tags. Finally, one more set of grammar rules is applied to get rid of the grammatical errors.

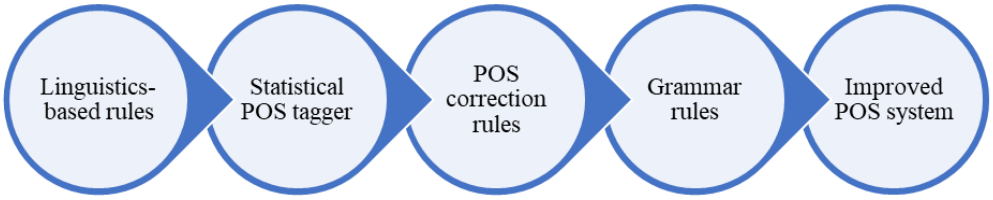



**Figure 10**: Step-by-step rules of the RAGE POS tagger (Jatav et al., 2017).

Gimpel et al. (2011) have used a supervised rule-based approach to create tagged dataset/tag set for specifically improve the POS tagging accuracy in Twitter domain. On the other hand, Owoputi et al. (2013) used an unsupervised clustering approach to develop a tag set (discussed in Section 2.1) using clustering method (Brown et al., 1992) and improved the POS tagging accuracy of taggers. Using the tag set, Gimpel et al. (2011) have developed a POS tagger to correctly tag the chat related data with 90% accuracy. The tag set was created by manually tagging the tweets and it includes all the Twitter specific characteristics such as hastags, emoticons, web links, at-mentions. This tag set is useful to train the POS taggers for an accurate POS tagging of conversational text. The stochastic POS tagger developed by Owoputi et

al. (2013) is based on the *maximum entropy Markov model* (MEMM) and it also achieves 90% tagging accuracy. MEMM is a variation of the hidden markov model discussed in Section 1.1.

A semi supervised neural network approach called *target preserved adversarial neural network* (TPANN) is introduced by Gui et al. (2017) to improve the POS tagging accuracy. In this approach, POS tagger is trained on pre-tagged data from different conversational and news text domains and it also extracts common features from the given test corpus to learn the grammatical structure in the corpus automatically. In this way, the tagger combines both supervised and unsupervised approaches to achieve more than 90% accuracy.

Ninomiya and Mozgovoy (2012) and Jørgensen and Søgaard (2016) both have used supervised approaches to improve the POS tagging accuracy. They have focused on improving the tagging accuracy for grammatically incorrect text because most taggers show poor performance on such type of corpus. Ninomiya and Mozgovoy (2012) have suggested to add an error corpus into the training data of a tagger. For this, they developed a tagged error corpus which contains grammatically incorrect sentences to train the existing POS taggers and improve their tagging performance by nearly 3% higher accuracy. Jørgensen and Søgaard (2016) suggested to add multiple corpora into training set from different domains such as subtitles from movies and television shows, conversational data, songs lyrics and tweets can help in better training of POS taggers.

# 3 POS Tagger Models

In this section, four state-of-the-art English language POS taggers are discussed in detail. The comparison of their features are presented in Table 3 at the end of this section.

## 3.1 Stanford POS Tagger

The *Stanford POS tagger* was developed by Kristina Toutanova at the Stanford NLP research group (Toutanova et al., 2003). The tagger has been implemented in *Java* programming language (Sun Microsystems, 1995) and is available for POS tagging in English, Chinese, Arabic and French language mainly. Stanford tagger shows good POS tagging accuracies in all these languages due to which many researches consider it as a gold standard for the evaluation purposes in their researches. For example, Gimpel et al. (2011) and Derczynski et. al (2013) have used the Stanford POS tagger for English language; Yu and Chen (2012) have used it for Chinese; Abdallah, Shaalan and Shoaib (2012) used it for POS tagging evaluation of Arabic; Bernhard and Ligozat (2013) used it for French. Many researchers also trained this tagger for POS tagging in other languges like Italian, Filipino and Twitter related English.

### 3.1.1 Maximum Entropy Model

The Stanford tagger uses a maximum entropy model to learn the language structure in a given text and assign POS tags to all the words according to the learned structure. In the maximum entropy algorithm, such a POS tag is selected for a word which has the highest probability in a list of POS tags and the corresponding words. The words and POS tags in a text are considered as sequences of words $\{w_1 \ldots w_n\}$ and tags $\{t_1 \ldots t_n\}$ to keep track of the context of words and tags with other words in a sentence. A *training data* is used for the learning of model, this data contains manually POS tagged words. Using the training data, POS tags are assigned to the words in *test data* which contains untagged words. Given a scenerio, in which a POS

tag has to be selected for a word, the algorithm will search for all the similar words and their POS tags in the training data. Then the most frequent POS tag found with a similar word in the training data will be selected and assigned to that word present in the test data (Toutanova and Manning, 2000). In Stanford tagger, the model is trained on the Penn Treebank tag set shown in Table 2 in Section 1.

### 3.1.2 Feature Set

The Stanford tagger contains following significant features. (Toutanova et al., 2003)

- The POS tag can be predicted for a word by looking at its context with either previous words or next words in a sentence, represented by dependency networks in Figure 9.
- A wide-range language feature set is supported by the tagger to identify correct POS tags for unique words like *EARTH* (all capital letters), *F/F-18 or CAT-12* type of words, company names like *i2c* and also prefixes/suffixes like *unequal/unhappy* or *strongly/easily*.
- Unknown words are handled with more care and intensive error analysis to achieve high POS tagging accuracy and this resulted in obtaining 56.34% correctly tagged sentences as reported by Toutanova et al., 2003.
- *Smoothing* techniques are used to preprocess the training data for accurate prediction of POS tags related to the new words that occur in the test data. Smoothing is explained in section 3.1.4.

### 3.1.3 Bidirectional Dependency Networks

A *dependency network* is the graphical representation of relationships between variables or tags and words (in our case). A simple dependency network is like the cyclic graph shown in Figure 11(c) while those in Figure 11(a) and 11(b) are unidirectional acyclic graphs called the *Bayesian networks*, in which the dependency of variables is one way. These graphs are used to determine the probability of the value or occurrence of a variable on the basis of its neighbour variables.

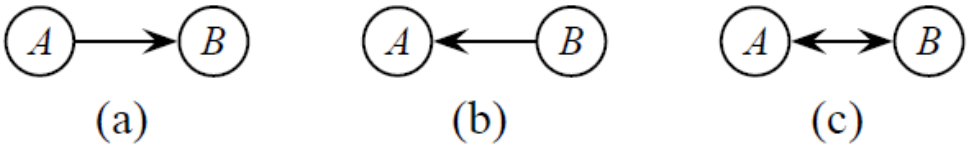



**Figure 11:** Simple dependency networks: (a) left-to-right, (b) right-to-left, (c) bidirectional network. (Toutanova et al., 2003)

Figure 12 is a representation of dependency networks used in *conditional Markov model* (CMM) by Ratnaparkhi (1996) to visualize the relationship between words and their tags in a sequence. The Stanford tagger model uses a form of such dependency networks by CMM called the *bidirectional dependency network* (Figure 12 (c)). According to the bidirectional dependency network, POS tag of a word can be predicted by looking at the context of its neighbouring words and tags in both the backward and forward directions. The bidirectional approach is more likely to predict a correct POS tag as compared to the one-directional dependency networks approach because the context and prediction features from both sides of the target word can be analyzed (Toutanova et al., 2003). In the left-to-right CMM shown in Figure 12 (a), the POS tag for a word is predicted based on the context of previous tags only while in the right-to-left CMM in Figure 12 (b), the POS tag is predicted only by the tags at next positions.

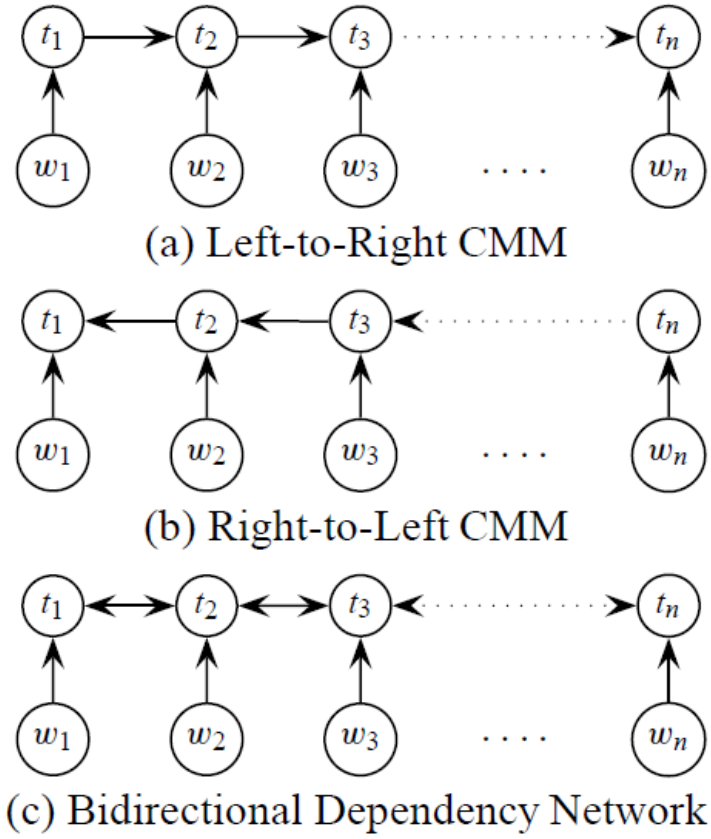
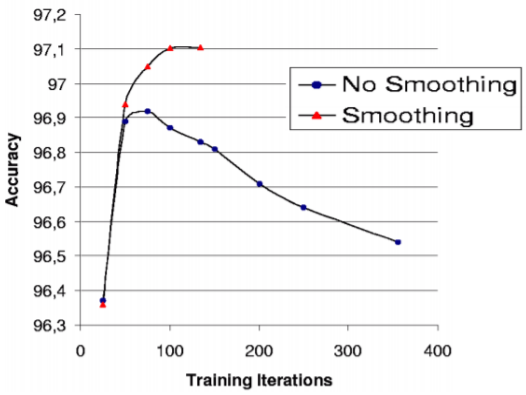



**Figure 12:** Dependency networks: (a) left-to-right (standard), (b) right-to-left (reversed), (c) bidirectional. (Toutanova et al., 2003)

### 3.1.4 Smoothing

*Smoothing* is a technique used in POS tagging to handle new words in the test data, that did not occur in the training data before. The POS tags for such unseen words are unknown but can be efficiently predicted with this technique. With the help of smoothing, taggers ignore unnecessary or repetetive data in the text and put restrictions on the training model to avoid the features which decrease the tagging accuracy. The smoothing technique used in Stanford POS tagger is called the *Gaussian prior smoothing*, in which a probability value is assigned to each tag in the training data and then these probabilities are used to estimate the tag for the new word. Toutanova et al. (2003) have proved that a smoothed training model in the Stanford tagger provides higher POS tagging accuracy as compared to an unsmoothed training model (Figure 13).




| | Overall Accuracy | Unknown Word Acc |
|---|---|---|
| Without Smoothing | 96.54 | 85.20 |
| With Smoothing | 97.10 | 88.20 |

**Figure 13:** Accuracy of Stanford Tagger with and without smoothing (Toutanova et al., 2003)

## 3.2 Trigrams'n'Tags POS Tagger

*Trigrams'n'Tags* (TnT) is a statistical/rule-based POS tagger, developed by Thorsten Brants in 1998. In contrast to the modern POS taggers which were programmed in Java, TnT was developed in ANSI C programming language. This POS tagger is not developed for any specific language so it can be adapted to any language efficiently. However, English and German models are provided with the tagger for training. Its architecture is designed to be integrated in an input/output pipeline of the NLP tasks (discussed in Section 1) such as, the tagger takes text as an input, assigns tag to each word in the text and transfers the output to the next processing level in an NLP task pipeline.

### 3.2.1 Second Order Markov Models

TnT POS tagger is based on the second order Markov models and can be easily trained on a large variety of text corpuses. A second order Markov model looks at the previous two tags in the sentence to predict a tag for the word in question. In this model, a text corpus is considered as a sequence of words and a corresponding sequence of word tags. Brants (2000) gives the formula in Equation 1 to calculate probabilities of tags for a given word. Equation 1 shows that, probability of a tag $t_i$

for the current word $w_i$ (represented by $P(w_i|t_i)$) is estimated by looking at the probabilities of the tags $t_{i-1}$ and $t_{i-2}$ of previous two positions in the sequence.

$$\underset{t_1 \ldots t_T}{\operatorname{argmax}}[\prod_{i=1}^{T} P(t_i|t_{i-1}, t_{i-2})P(w_i|t_i)]P(t_{T+1}|t_T) \qquad (1)$$

Where $w_1 \ldots w_T$ is the sequence of words in a text corpus, $T$ is the total length of the text and $t_1 \ldots t_T$ represent tags for the corresponding words.

TnT model architecture is divided into two steps as shown in Figure 14. In the first step, probabilities of one (unigram), two (bigram) or three (trigram) previous tags are measured to estimate the probability of next tag and also the lexical or context information of the words is used. In the second step, smoothing (explained in Section 3.1.4) is performed and unknown or new words are handled.

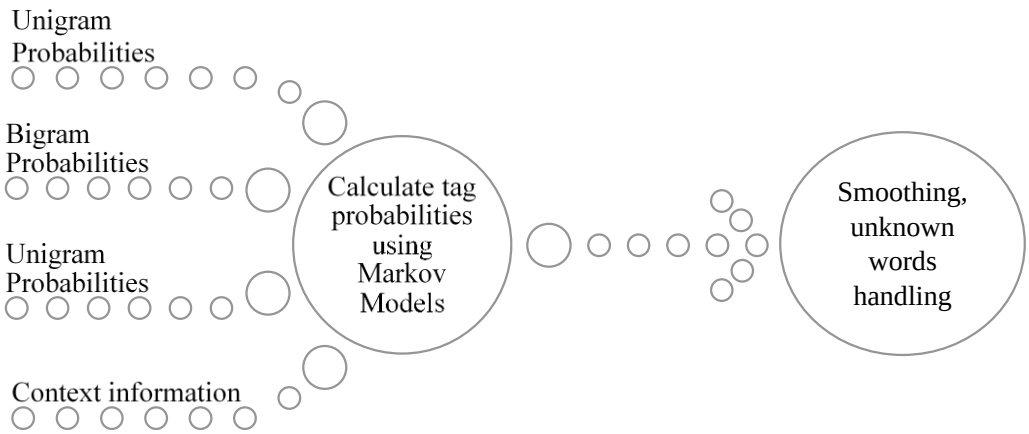



**Figure 14:** TnT system architecture as explained by Brants (2000).

To handle the unknown words, suffix analysis method has been used in which the ending of each word is analyzed up to a defined length. For example, the words *commitment*, *enjoyment*, *disappointment* have a same suffix *-ment*. By this method, such type of words having a same ending are assigned similar probabilities so that a correct tag can be predicted for a new word with matching suffixes.

All the above discussed methods and features of TnT have achieved more than 95% average POS tagging accuracy on English and German language text as evaluated and recorded by Brants (2000).

## 3.3 Support Vector Machine POS Tagger

*Support vector machines* (SVM) is a machine learning algorithm used for classification purpose. For example, in Figure 15 SVM classifies the red and blue circles by creating a boundary line between them. SVM model was first introduced by Vladimir Vapnik in 1995 and the basic concept was to find such a line in the given data which can perfectly divide the elements of that data into two classes. Later, the concept was extended to classify multiple data elements into as many classes as required.

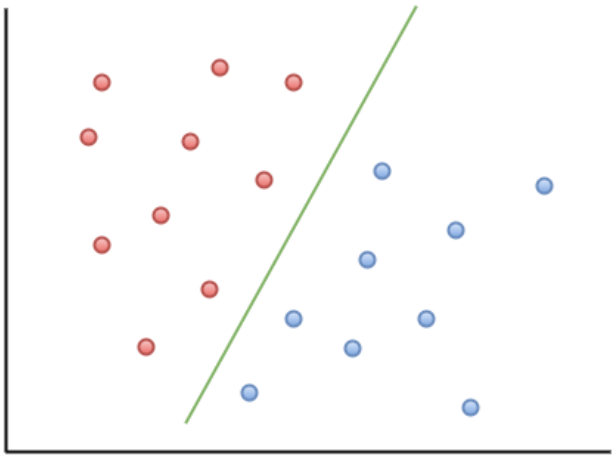


**Figure 15:** An example of SVM classification (geohackweek, 2016).

SVM POS tagger is provided as a component of SVMTool software package developed by Giménez and Márquez in 2004. SVMTool is based on SVM framework and was developed in C++ and Perl[4] programming languages. The

[4] https://www.perl.org/

package includes a data learning tool called *SVMTlearn*, a POS tagger called *SVMTagger* and an evaluating tool called *SVMTeval*. First of all, SVMTlearn is used to learn the language structure of the training corpus. Secondly, SVMTagger is trained by SVMTlearn and POS tagging is performed according to the training data. SVMTeval is used at the end to evaluate the POS tagging results of SVMTagger.

SVMTlearn uses a software called SVM-light[5] (Joachims, 1999) to train the parameters of the SVM model on a given training corpus. The grammar rules and language structure is analyzed to extract the common features of the training data. Then the SVM model parameters are adjusted according to these features and applied to the tagger.

SVMTagger performs the POS tagging on a test corpus, with the help of the features and learned model provided by SVMTlearn. The tagger is flexible to be customized with different options according to the user requirements such as

- Two tagging schemes are available, *greedy* and *sentence-level*. Greedy tagging is performed based on the context of few words while the sentence-level tagging is performed in the context of whole sentence.
- The tagging direction can be chosen as left-to-right, right-to-left or a combination of both.
- Tagging can be done in two passes, by which POS tags are analyzed in the second pass and error can be reduced.
- A backup dictionary can also be provided to handle the new words that were not present in the training data.

---

[5]The SVM[light] software is freely available (for scientific use) at the following link http://svmlight.joachims.org

- SVMTagger also has the option to ignore or eliminate those model parameters which create unnecessary tags in the data. In this way, tagging accuracy can be improved by removing such noise from the data.

- The tagger is made language independent and can be trained on any language. However, it comes with the pre-trained language models of English, Spanish and Catalan.

SVMTeval checks the tagging accuracy of SVMTagger by calculating the number of correctly tagged words and compares it with the *ground truth*. Ground truth is a collection of correctly tagged words used as a standard for evaluation of tagger. SVMTeval helps in tuning the tagger and enhance its performance.

Giménez and Màrquez (2004) have described the SVMTagger as a simple, customizable and easy-to-use POS tagger giving over 97% average tagging accuracy on the standard English and Spanish text corpora.

## 3.4 Natural Language Processing POS Tagger

Natural language processing (NLP) for Java virtual machine (JVM) languages (NLP4J) is a project run by Emory NLP research group. The project provides an efficient POS tagger developed by Jinho D. Choi in 2016. The tagger is developed in Java programming language and is trained for various categories of English models such as newswire, web texts, telephone conversations and emails (Choi, 2016).

This POS tagger is designed on a novel technique called the generalized model selection in which the tagger automatically learns the language patterns and common features in the training data. By learning through the given training data, the tagger gets able to detect common patterns in a test data and create combinations of these patterns related to the words. For example, if two words are usually seen written together in the training data such as *Los Angeles*, then the tagger detects this feature, applies it on similar words and assign same type of POS tags to such words. There is another feature in the tagger called *ambiguity class* in which the words having an

option of getting more than one POS tag are assigned to the ambiguity class [Moore, 2015]. In the ambiguity class of a word, all possible POS tag combinations of that word are added, for example, if the word *book* can be tagged as *NN* (common noun) and also as *VB* (verb), then an ambiguity class of this word will be *NN_VB*.

This POS tagger is unique in terms of speed as it takes less time for POS tagging of very large corpora. Choi, J. (2016) claims that it has the ability of tagging eighty two thousand words in one second which is very fast and efficient. The tagger is designed to perform on several types of text collections and it can perform with more than 97% tagging accuracy on a mixed corpus collected from different source.

**Table 3:** Comparison of the features of POS taggers.

| | **Stanford Tagger** | **TnT Tagger** | **SVM Tagger** | **NLP4J Tagger** |
|---|---|---|---|---|
| Availability | Freely available to download | Not free to download | Available but needs to install non free software | Free to install |
| Training Features | Flexible to be retrained on any language | Can be trained on any language | Easy to train on any language | Can be trained on English models only |
| Code | Java | ANSI C | Perl/C++ | Java |
| Architecture | Flexible to be installed on any hardware and software platform | Supports UNIX/LINUX platforms | Can be easily installed using Perl library | Needs to be installed with another software called Maven |
| Usability | Good documentation, and tool maintainence with active user support community. | Good tutorials/ documentation but not very active support and maintenance of the tool | Very good documentation and support groups for public use | Good documentation and active discussion groups available for support |
| Trained Models | English, Chinese, Arabic, French, German and Spanish | English and German | English, Spanish and Catalan | English |

# 4 Experimental Results

We have performed experiments on Stanford tagger, TnT tagger and NLTK POS tagger. The POS tagging accuracies are compared on two different datasets provided by *natural language toolkit* (NLTK), shown in Table 4.

**Table 4**: Our experimental dataset statistics.

| **Datasets** | **Number of words** |
|---|---|
| News Text (Brown corpus) | 38,087 |
| NPS Chat | 148,576 |

*NLTK* library (Bird et al., 2009) consists of a huge collection of text data and *Python* functions to implement on that data. *Python*[6] is a programming language that is used to develop many NLP applications and tools such as a sentiment analysis tool. NLTK also has a variety of text corpuses such as Brown corpus, web text corpus and NPS Chat corpus for training the taggers according to the required test data. We have explored and used Brown corpus and NPS Chat corpus for our experiments. Following methods are used to import these corpora.

```
>>> from nltk.corpus import brown

>>> from nltk.corpus import nps_chat
```

We used tagged_words() method to find out that the Brown corpus and NPS Chat corpus contain tagged sentences and can be used to evaluate POS tagging in our experiments. So, we apply Stanford tagger, NLTK tagger and TnT tagger on these two

[6] https://www.python.org/doc/essays/blurb/

corpuses. Then we compared the tagged sentences with our results to calculate the number of correctly tagged sentences by each tagger.

Brown corpus is the first largest English corpus created in 1960s at the Brown university. The corpus contains text from 500 different sources, each source contains more than 2000 words. This sums up the number of words in the corpus to be around 1 million and the number of sentences more than fifty thousand (Francis and Kučera, 1979). The NLTK library has divided the Brown corpus into categories like news text, chat text, reviews text, et cetera. Experiments in this thesis have been performed on the newswire text category which consists of more than thirty thousand words.

The tag set used in Brown corpus is based on Penn TreeBank tag set (Table 2) having total 90 tags. The graph in Figure 16 shows top 20 frequently occurring tags in our newswire data set. These tags are summarized in Table 5 with examples. Most frequent tag is *NN*, which represents *nouns*. It exists more than 3000 times in our dataset. The second most frequent tag is *IN*, which represents *prepositions*. It is found more than 2000 times. Third frequent tag is *AT* which represents *articles* like *a, the*; next is *NP* for *proper nouns* like *Alex, John, Finland*; then *NNS* for *plural nouns* like *dogs, books*; *JJ* for *adjectives* like *cold, fast, blue*; *VB* for *verbs* like *running, laughed*; *VBD* for *past verbs* like *ran, thought*; *VBN* for *past participle verbs* like *been, gone*; *CC* for *coordinate conjunctions* like *and, or*; *CD* for *cardinal numerals* like *one, two, 3, 4*; *RB* for *adverbs* like *very, too*; *CS* for *subordinate conjunctions* like *if, though*; *TO* for *to*; *VBG* for *present participle verbs* like *being*; *MD* for *modal auxiliary words* like *should, can, shall*; *PPS* for *third singular nominative pronouns* like *he, she, it*. We note that frequent symbols like commas and full stops have separate tag categories occurring more than 1000 times in our dataset. Another important feature in Brown corpus is *NN-TL* tag that represents *nouns in titles* such as *State, University, President*.

**Table 5**: Examples and definitions of most frequent tags in Brown Corpus

| **Tags** | **Definition** | **Examples** |
|---|---|---|
| NN | Noun | Chair, tree, cat |
| IN | Preposition | at, on, in, for |
| AT | Article | a, the |
| NP | Proper noun | Alex, John, Finland |
| NNS | Plural noun | Dogs, books |
| JJ | Adjective | Cold, fast, blue |
| VB | Verb | Running, laughed |
| NN-TL | Noun in Title | University, President |
| VBD | Past verb | Ran, thought |
| VBN | Past participle verb | Been, gone |
| CC | Coordinate conjunction | And, or |
| CD | Cardinal numeral | One, two, 3, 4 |
| RB | Adverbs | Very, too |
| CS | Subordinate conjunction | If, though |
| TO | To | To |
| VBG | Present participle verb | Being |
| MD | Modal auxiliary words | Should, can, shall |
| PPS | 3rd singular nominative pronoun | He, she, it |

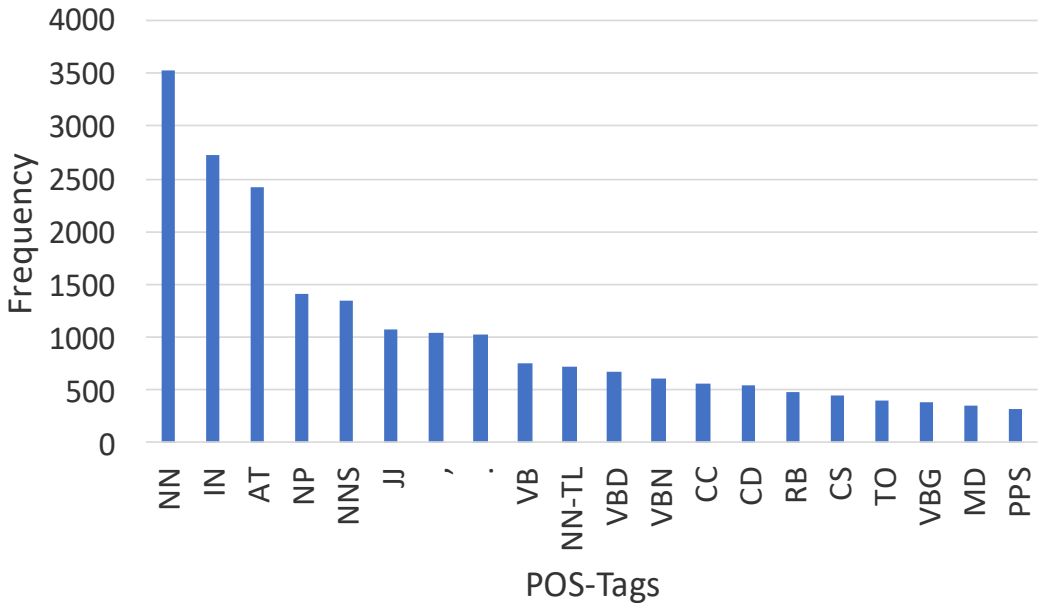



**Figure 16**: Twenty most frequent POS tags in Brown corpus.

The NPS Chat corpus was developed by the Naval Postgraduate School (NPS) in 2010 and is provided by the NLTK library (Forsyth et al., 2007). It is a collection of more than 10,000 conversational messages collected from different internet chatroom services. The text includes informal use of language used during chats like *tweet* text. Therefore, we have chosen this dataset to evaluate the performance of taggers on conversational data in addition to the news text data. This corpus is also based on Penn TreeBank tag set. Figure 17 shows top 20 tags found in our data set of more than 3000 chat messages. The most frequent tag is *UH* which represents abbreviated words like LOL (*lots of laughter*), BRB (*be right back*) and ILY (*I love you*). Second most frequent tag is *NNP* which represents *proper nouns* like *NP* (*Alex, John, Finland*) in Brown corpus. Most tags are similar in both corpuses like *NN, VB, RB, IN, JJ, VBD, VBG, NNS, CC, TO* and symbols like commas and full stops. However, we found few different frequent tags in the NPS chat corpus. For example, *VBP* which is identical to *base verbs* (*VB*) like *stand, ride*; *DT* represents *determiners* like *the*; *PRP* represents *personal pronouns* like *I, you, she, they* and *PRP$* represents *possessive pronouns* like *mine, our, theirs*.

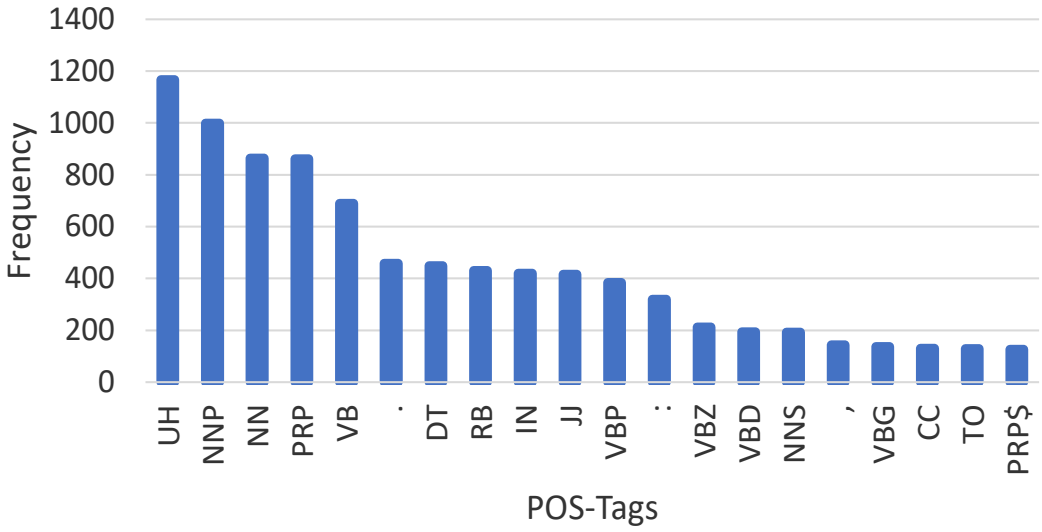



**Figure 17**: Twenty most frequent POS tags in NPS Chat corpus

The POS tagging accuracies are listed in Table 6. We have used pre-trained taggers for both datasets. All three taggers are trained on standard English newswire text. The taggers have performed better on the news text as compared to the chat text, which implies that the pre-trained (on formal English text) taggers struggle to identify tags for unknown or rare words found in the text. In this experimental setting, TnT tagger achieves lowest accuracy on the chat text dataset however, it outperformed the two taggers on news text dataset. This performance of the TnT tagger can be explained as; it successfully manages to tag the words already seen in training data (Brown corpus) better than the other two taggers but performs poorly on the unseen words found in NPS chat corpus. On the other hand, Stanford tagger shows the highest tagging accuracy on the unseen words in NPS chat corpus and a better accuracy then NLTK tagger on known/seen words in Brown Corpus. NLTK tagger shows the lowest tagging accuracy on the seen words in the Brown training corpus but the tagging accuracy on unseen words is slightly higher than the TnT tagger. The graphical representations of POS tagger performances are also shown in Figure 18.

**Table 6**: POS tagging accuracies

| POS Taggers | Datasets | |
|---|---|---|
| | Brown Corpus | NPS Chat Corpus |
| Stanford Tagger | 59.57% | 56.70% |
| NLTK Tagger | 57.43% | 51.60% |
| TnT | 76.52% | 51.55% |

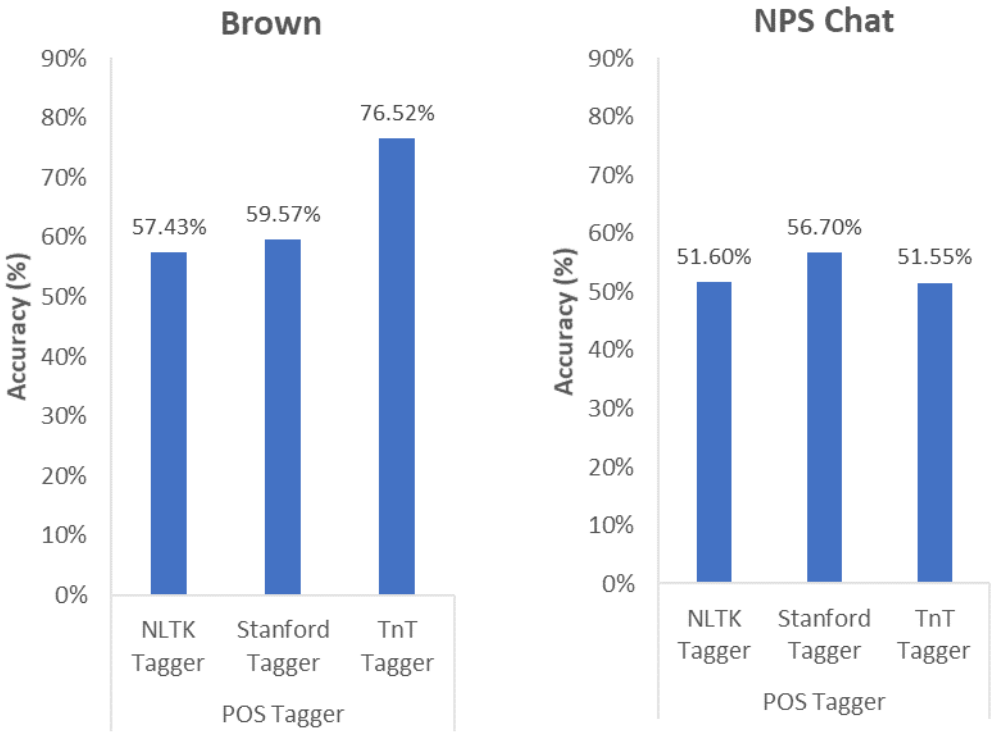



**Figure 18:** Accuracies for POS taggers on both datasets.

When we look at the results in terms of corpuses, the Brown corpus is tagged by all three taggers with clear differences in the tagging accuracies. This shows the level of tagging performance by each tagger on a grammatically correct, error-free and formal language text. However, for the NPS chat corpus the taggers show slight differences in their tagging accuracies. This shows that an informal language text with grammatical errors and spelling mistakes is handled by all three taggers on almost the same or with a small difference in tagging level.

We also implement the taggers on a small sample of tweets to analyze the difference in assigned tags by each tagger. Our sample of Twitter database consists of twenty thousand tweets collected by Laitinen et al. (2018). In Figure 19, we show an example of a single tweet from our sample database with the tagged output given by each tagger.

Input: [ @Eliaswetter @AJ3 He is cheapest 93 rated by about 200k. Seams fair.]

Output:

```
Stanford Tagger
[('@', 'SYM'), ('Eliaswetter', 'NNP'), ('@', 'SYM'),
('AJ3', 'NN'), ('He', 'PRP'), ('is', 'VBZ'), ('cheapest',
'JJS'), ('93', 'CD'), ('rated', 'VBN'), ('by', 'IN'),
('about', 'IN'), ('200k', 'NN'), ('.', '.'), ('Seams',
'NNPS'), ('fair', 'JJ'), ('.', '.')]
-----------------------------------------------------------
-----------------------------------------------------------
NLTK Tagger
[('@', 'JJ'), ('Eliaswetter', 'NNP'), ('@', 'NNP'),
('AJ3', 'NNP'), ('He', 'PRP'), ('is', 'VBZ'), ('cheapest',
'JJS'), ('93', 'CD'), ('rated', 'VBN'), ('by', 'IN'),
('about', 'IN'), ('200k', 'CD'), ('.', '.'), ('Seams',
'NNP'), ('fair', 'NN'), ('.', '.')]
-----------------------------------------------------------
-----------------------------------------------------------
TnT Tagger
[('@', 'IN'), ('Eliaswetter', 'NN'), ('@', 'IN'), ('AJ3',
'NN'), ('He', 'PRP'), ('is', 'VBZ'), ('cheapest', 'JJS'),
('93', 'CD'), ('rated', 'VBD'), ('by', 'IN'), ('about',
'IN'), ('200k', 'NN'), ('.', '.'), ('Seams', 'NN'),
('fair', 'JJ'), ('.', '.')]
```

**Figure 19**: Single tweet example with taggers output.

Table 7 shows the tags assigned to each word of the tweet sentence in Figure 19 by NLTK, Stanford and TnT tagger in comparison with the ground truth. In a single tweet, we analyzed the case in which all three taggers succeed to tag a word, a case where all taggers fail and a case where one tagger fails and other two pass and vice versa. We see that the symbol @ is correctly tagged by the Stanford tagger only. Other two taggers do not tag the @ symbol correctly because most likely the taggers have not seen this symbol in their training data. So, the Stanford tagger successfully detects and tags the symbols in tweets while other two taggers fail for such unseen symbols in this experiment. The usernames (*Eliaswetter* and *AJ3*) are correctly tagged by the NLTK tagger as *proper noun* (*NNP*) while the other two taggers tag them as *noun* (*NN*) which do not match our ground truth. Some words like *He, is, cheapest, 93, rated, by, about* are correctly tagged by all three taggers (tags explained in Table 2). The word *rated* is tagged as *VBN* (*past participle verb*) by both Stanford and NLTK taggers and as *VBD* (*past verb*) by TnT tagger. Both of these tags (*VBN*/*VBD*) are forms of *past verb* and are considered correct. The numeric word *200k* is tagged by the NLTK tagger correctly as *cardinal numeral* (*CD*) and other two taggers fail to detect the correct tag of this word because it is a rare combination of numbers and letter. *Full stop* (.) is considered as a separate tag category (as shown in Figure 16 and 17 too) and has been tagged correctly by all taggers. Furthermore, all three taggers fail to correctly tag the word *Seams*, because the word is spelled as *Seams* instead of *seems* in this tweet example. All three taggers have detected the word *Seams* as a *noun* (*NNPS, NNP, NN*) because the first letter of the word is capital. However, the correct tag should be *verb* (*VB*) or *present participle verb* (*VBG*). The word *fair* is correctly tagged by the Stanford and TnT taggers but NLTK tagger fails in this case. All the tags shown in Table 7 have been explained with examples in Table 5.

We observed that TnT tagger successfully tags all the correct English words such as *He, is, cheapest, 93, rated, by, about* but fail to detect all the misspelled and rare words. This shows that the performance of TnT tagger highly depends on its training data and this is why it gives wrong tags to the rare or unseen words. NLTK tagger is weak in predicting symbols but good in predicting numeric terms. Stanford tagger

detects the symbols correctly but fails in predicting the tags for unique numeric words.

**Table 7**: Comparison of assigned tags by each tagger

| Words | Stanford | NLTK | TnT | Ground Truth |
|---|---|---|---|---|
| @ | SYM | JJ | IN | SYM |
| Eliaswetter | NNP | NNP | NN | NNP |
| @ | SYM | NNP | IN | SYM |
| AJ3 | NN | NNP | NN | NNP |
| He | PRP | PRP | PRP | PRP |
| is | VBZ | VBZ | VBZ | VBZ |
| cheapest | JJS | JJS | JJS | JJS |
| 93 | CD | CD | CD | CD |
| rated | VBN | VBN | VBD | VBD/VBN |
| by | IN | IN | IN | IN |
| about | IN | IN | IN | IN |
| 200k | NN | CD | NN | CD |
| . | . | . | . | . |
| Seams | NNPS | NNP | NN | VB/VBG |
| fair | JJ | NN | JJ | JJ |

# 5 Conclusions

In this thesis, a literature review of state-of-the-art English POS taggers has been presented. The recent research on the POS tagging techniques and the advancements in different tagging approaches has been compared. This comprehensive study shows that new tagging techniques are following new learning methods and have introduced revised training datasets to handle challenges of informal terms being used in the conversational data such as in Twitter data.

Our study on existing POS tagging methods concludes that, for a robust tagging performance, there is a need of reliable models with strong feature set and a variety of linguistic resources to train the taggers. The training dataset combined from different sources ensures an improved tagging consistency of a tagger. For this reason, the addition of unknown words and unique phrases in the training data has shown to be a good performance indicator in different researches.

The experiments carried out in this thesis have shown POS tagging accuracies of three POS taggers on two different corpora. We have analyzed the differences in each tagger according to their tagging accuracies. All three taggers were taken as pre-trained on standard English text. The Stanford and NLTK taggers both show small difference in the tagging accuracies on the standard English text of Brown corpus, but TnT tagger differs significantly by almost 19%. On the conversational text of NPS chat corpus, both NLTK and TnT taggers show similar tagging accuracies while Stanford tagger outperforms them with a significant difference of over 5% in the accuracy value. However, all three taggers give more than 50% correctly tagged words on both data sets. In this way we have also analyzed the differences in the standard and non standard corpuses.

Our results reveal that the conventional taggers, which are trained on a grammatically perfect formal text, do not perform as accurate on the conversational text as on a news text. The Stanford tagger shows a difference of ~2.5% in the accuracies of standard and non standard text. NLTK tagger shows a difference of

~5.8% and TnT tagger has shown the difference of ~24.9% in the tagging accuracies of both datasets. The reason is that these POS taggers do not deal very well with the unknown words and specially the abbrevated words that frequently occur in the conversational or chat messages.

In this thesis, we have provided a comprehensive overview of POS tagging concepts. The study highlights the difference in POS tagging performances based on different types of text categories. The main challenges discussed include the tagging of unknown words, grammatically incorrect sentences, words with spelling errors, abbreviated words and special characters in the text.

These challenges can be handled in several ways. One way is to improve the quantity and quality of training sets. Many researches show that including varieties of text from different domains other than standard news category in the training data can increase the tagging accuracy of taggers.

One other way is to use a combination of different tagging models which can significantly improve the tagging accuracy as shown by some researches. Another way is to perform rigorous error analysis techniques and minimize the tagging errors. The study can be a great addition in the linguistics research category to carry out further improvements in the fields of POS tagging and NLP.

# REFERENCES


Jatav, V., Teja, R. and Bharadwaj, S. 2017. Improving Part-of-Speech Tagging for NLP Pipelines. *The Computing Research Repository (CoRR) abs/1708.00241*.

Derczynski, L., Ritter, A., Clark, S. and Bontcheva, K. 2013. Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. Proceedings of the *International Conference Recent Advances in Natural Language Processing RANLP 2013*.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. and Smith, N.A. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. Proceedings of the 49th Annual Meeting of the *Association for Computational Linguistics: Human Language Technologies*.

Gui, T., Zhang, Q., Huang, H., Peng, M. and Huang, X. 2017. Part-of-Speech Tagging for Twitter with Adversarial Neural Networks. Proceedings of the 2017 *Conference on Empirical Methods in Natural Language Processing*.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N. and Smith, N.A. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. Proceedings of the 2013 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

Weerasooriya, T., Perera, N. and Liyanage, S.R. 2017. KeyXtract Twitter Model - An Essential Keywords Extraction Model for Twitter Designed using NLP Tools. Proceedings of the *10th KDU International Research Conference*.

Levy, R. and Andrew, G. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures, *5th International Conference on Language Resources and Evaluation (LREC 2006),* pages 2231 - 2234.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K. and Schasberger, B. 1994. The Penn Treebank: Annotation Predicate Argument Structure, Proceedings of the *workshop on Human Language Technology - HLT '94*, pages 114 - 119.

Toutanova, K., Klein, D., Manning, C. and Singer. Y. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of *HLT-NAACL 2003*, pages 252 - 259.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R. and Kadie, C. 2000. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, pages 49 - 75.

Brants, T. 2000. TnT – a statistical part-of-speech tagger. *Applied Natural Language Processing Conference 6*, pages 224 - 231.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. *Conference on Empirical Methods in Natural Language* Processing 1, pages 133 - 142.

Asmussen J. 2015. Survey of POS taggers. Technical Report, *DK-CLARIN WP 2.1*.

Hasan, M., F., UzZaman, N. and Khan, M. 2007. Comparison of Unigram, Bigram, HMM and Brill's POS Tagging Approaches for some South Asian Languages.

Brill, E. 1994. Some Advances in Transformation-Based Part of Speech Tagging. *AAAI '94 Proceedings of the Twelfth National Conference on Artificial Intelligence (vol. 1)*, pages 722-727.

Giménez, J. and Màrquez, L. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In Proceedings of the *4th International Conference on Language Resources and Evaluation (LREC'04)*, Volume 1, pages 43 - 46.

Choi, D., J. and Palmer, M. 2012. Fast and Robust Part-of-Speech Tagging Using Dynamic Model Selection. *ACL '12 Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*: Short Papers - Volume 2, pages 363-367.

Choi, D., J. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. Proceedings of the *54th Annual Meeting of the Association for Computational Linguistics (NAACL'16).*

Robert M. 2015. An Improved Tag Dictionary for Faster Part-of-Speech Tagging. In Proceedings of the *Conference on Empirical Methods in Natural Language Processing, EMNLP'15*, pages 1303–1308.

Jørgensen, A. and Søgaard, A. 2016. A Test Suite for Evaluating POS Taggers across Varieties of English. *WWW '16 Companion Proceedings of the 25th International Conference Companion on World Wide Web*, pages 615-618.

Moreno-Monteagudo, L., Izquierdo-Beviá, R., Martínez-Barco, P., and Suárez, A. 2006. A Study of the Influence of PoS Tagging on WSD. *Lecture Notes in Computer Science*, pages 173-179.

Watson, R. 2006. Part-of-speech tagging models for parsing. *Proceedings of the 9th Annual CLUK Colloquium*, Open University, Milton Keynes, UK.

Buitelaar, P., Ramaka, S. 2005. Unsupervised Ontology-based Semantic Tagging for Knowledge Markup. *International Workshop on Learning in Web Search at ICML*, pages 26-32.

Ninomiya, D. and Mozgovoy, M. 2012. Improving POS tagging for ungrammatical phrases. *HCCE '12 Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments*, pages 28-31.

Yu, C. and Chen, H. 2012. Detecting Word Ordering Errors in Chinese Sentences for Learning Chinese as a Foreign Language. *Proceedings of COLING 2012*. Technical Papers, pages 3003–3018.

Abdallah, S., Shaalan, K. and Shoaib, M. 2012. Integrating Rule-Based System with Classification for Arabic Named Entity Recognition. *CICLing 2012*, pages 311–322.

Bernhard, D. and Ligozat, A. 2013. Hassle-free POS-Tagging for the Alsatian Dialects. *Non-Standard Data Sources in Corpus Based-Research*, pages 85- 92.

Guilder, L. 1995. Automated Part of Speech Tagging: A Brief Overview. *Handout for LI5G361*.

Bird, S., Klein, E. and Loper, E. 2009. Natural Language Processing with Python. *O'Reilly Media*.

Forsyth, E., Lin, J. and Martell, C. 2007. Lexical and Discourse Analysis of Online Chat Dialog. *Proceedings of the First IEEE International Conference on Semantic Computing*, pages 19-26.

Laitinen, M., Lundberg, J., Levin, M. and Martins, R. 2018. The Nordic Tweet Stream: A dynamic real-time monitor corpus of big and rich language data. *DHN*.