Valtteri Korolainen

PART-OF-SPEECH TAGGING IN WRITTEN SLANG

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2014

TIIVISTELMÄ

Korolainen, Valtteri
Part-of-Speech Tagging in Written Slang
Jyväskylä: Jyväskylän yliopisto, 2014, 76 s.
Tietojärjestelmätiede, pro gradu-tutkielma
Ohjaaja: Puuronen, Seppo

Erilaiset kieliteknologiasovellukset ovat olleet jo vuosikymmeniä arkipäiväisessä käytössä. Esimerkiksi ennustava tekstinsyöttö ja automaattinen korjaus ovat olleet käytössä jo vuosikymmeniä. Puheen tunnistus ja kielen automaattinen kääntäminen ovat puolestaan hieman uudempia sovelluksia. Tieteenalana kieliteknologia on vuosikymmeniä vanha, mutta silti koneilla on vielä monesti vaikeuksia ymmärtää luonnollisia kieliä. Tämän tutkimuksen tavoite on kartoittaa koneiden kykyä annotoida tekstiä automaattisesti kun käsiteltävä aineisto sisältää slangia. Tutkimus sisältää empiirisen kokeen automaattisten annotointialgorimien toiminnasta. Kielen prosessointi on myös nykyään käytössä olevilla algoritmeilla verrattain raskasta. Osa sovelluksista voidaan kuitenkin suorittaa pilvipalveluissa. Eurooppalaisten kielien prosessointi nykyalgoritmeilla on kohtuullisen hyvällä tasolla verrattuna moniin muihin kieliin. Tähän syynä on huomattavasti laajempi taustatyö. Vaikka monet sovellukset onnistuisivat usein ymmärtämään luonnollista yleiskieltä, niin slangin prosessointi on huomattavasti hankalampaa. Pääsyyt slangin prosessoinnin haasteellisuudelle ovat slangitutkimuksen vähäisyys kieliteknologioihin liittyen sekä slangin monesti kompleksisempi luonne. Automaattinen simultaanitulkkaus on jo jossain määrin mahdollista nykyaikaisilla kieliteknologiasovelluksilla. Yksi tapa arvioida tiettyä kieliteknologiaa on analysoida taustalla olevaa sanaluokkajäsentäjää, jonka tehtävä on annotoida tekstifragmentteja. Tämän tutkimuksen tutkimusongelmana on selvittää n-gram algoritmin suorityskyky muihin käytössä oleviin algoritmeihin nähden slangia annotoitaessa. Tilastollisia lähestymistapoja käytettäessä myös taustalla oleva manuaalisen jäsentämisen laajuus vaikuttaa merkittävästi sanaluokkajäsentäjän toimintaan. Eurooppalaiset kielet voidaan prosessoida monesti luotettavammin tilastollisilla menetelmillä, kun taas esimerkiksi Etelä-Intian kielet, kuten Hindi, ovat monesti luotettavampia prosessoida sääntöihin perustuvilla menetelmillä. Englanninkieli voidaan luonnollisessa muodossaan annotoida automaattisesti 97% tarkkudella; englanninkielisen slangin automaattinen annotointi saavuttaa puolestaan vain 93% tarkkustason. Tutkimustuloksista voidaan todeta, että vaikka algoritmin valinta vaikuttaa osaltaan annotoinnin tarkkuuteen, niin sääntöihin perustuvat menetelmät ovat tärkeä lisä slangin annotoinnissa. Tärkein sääntöihin perustuva lisämenetelmä on sanojen klusterointi.

Asiasanat: Automaattinen sanaluokkajäsennys, Markovin piilomalli, Luonnollisten kielten käsittely, Algoritmit, Koneoppiminen, Kieliteknologiat

ABSTRACT

Korolainen, Valtteri
Part-of-Speech Tagging in Written Slang
Jyväskylä: University of Jyväskylä, 2014, 76 p.
Information Systems, Master's Thesis
Supervisor: Puuronen, Seppo

Contemporary computers have different capabilities to process natural languages. For example speech recognition and machine translation are both due to study of natural language processing (NLP). Still, machines have some problems of understanding a natural language since words can be ambiguous. Most of the time machines are able to understand the single words. Complete sentences are causing more problems. As well, a part of the actual language processing is moved to cloud from local machines due to heavy algorithms that have a high time or space compelexity. English and other European languages have better success rate in NLP solutions than other languages. Mainly this is because of the amount of work and prior analysis done around the language. Even though variety of different NLP solutions exists, they are mainly focused on standard language. Our research contains empirical study which goal is to describe n-gram algorithm suitability in automatic slang annotation context. Slang processing is more problematic than processing standard language, which can be seen in lower accuracy rates. Some of the problems are caused lack of extensive slang analysis when on the other hand some problems are due to complexity of slang. Simultaneous interpreter is one possible solution of upcoming NLP innovations but it has limitations since slang processing is still partly under a development. One way to analyze lingual capabilities of a machine is to evaluate the success rate of Part-of-Speech (POS) tagging. The research problem is how n-gram algorithms are performing in slang tagging compared to previously experimented algorithms. As a result of this study it is been found that tagging algorithm selection is in major part of tagger accuracy. In statistical approaches corpus size is remarkably affecting the accuracy as well. Languages are performing differently with different algorithms. For instance, statistical tagging algorithms are mostly having better accuracies in European languages while rule based tagging algorithms are outperforming statistical taggers in South Indian languages. From the POS tagging point of view English slang can be considered as different language from Standard English. While Standard English text can be automatically tagged with success rate of 97% the slang taggers are only fairly reaching 93% success rate. As a conclusion for research findings, rule-based approaches are important addition for slang POS taggers. Most important of these kinds of tools is word clustering.

Keywords: Part-of-Speech tagging, Hidden-Markov Model, Natural Language Processing, Algorithms, Machine Learning, Language Technologies

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AI	Artificial Intelligence
API	Application Programming Interface
CRF	Conditional Random Fields
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
MEHMM	Maximum-Entropy Hidden Markov Model
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS	Part-of-Speech
SVM	Support Vector Machines

CONTENTS

1 INTRODUCTION

This chapter describes the scope of this study, why the study is made and what are the focus points. The chapter describes the background of the domain. As well, this chapter introduces motivational use-cases and new technological possibilities and upcoming technologies strictly related to the topic.

1.1 Motivation

Machine learning has got interesting solutions and it has been applied to many different kinds of uses during recent years. For example phones and gaming devices have voice controlling abilities and computers can be used by neural impulses with an electroencephalography headset. As well, these inventions enable a lot of possibilities for technical variations. Related inventions can be used in gaming industry or as important feature in assistant tools. For instance one of these kinds of inventions already made is a mind control for an electric wheelchair, which is one example of neuroheadset solution developed by Emotiv Systems (Emotiv.com). As well, Google has already developed and published an eyeglass extension, which allows user to have small transparent screen on one's sight and it has cell phone-like abilities (Washingtonpost.com). For example, the glasses are able to show on transparent screen incoming text messages to your headset and they enables different possibilities for augmented reality solutions. Further more, studies of similar technology for contact lenses have already started, but they are not as close to be published yet (Cnet.com). As Kaku (2011, p. 6) has visioned, that in the future if you meet individuals speaking foreign language unfamiliar to you, subtitles would appear straight into your contact lenses as they speak. As well, one lower step in implementing this kind of technology could be automatic subtitling tool for movie industry. This kind of technology is compilation of many different technologies on hardware and software level. Still, at least the hardware level is already relatively near to this kind of solutions.

Even though the bionic contact lenses are still under development, similar simultaneous translator solutions might already be possible for example with Google glasses if some voice recognition tool would be integrated with Google translate for instance (Google.com). This study is related to one technology on software level, which would make automated interpretation possible in such solution. Application development and research in this field are categorized under a study of Natural Language Processing (NLP), which is one subcategory under a study of Artificial Intelligence (AI). Even though AI includes many different fields of studies, they still have a lot of common principles as well and the domains might have other overlapping parts. Many mathematical models and computational algorithms are applied in various differing domains. For instance, one key algorithm used in this study is called Hidden Markov Model (HMM), which is widely used in different areas of language processing. As well, since HMM is not originally related to language technologies, it is used in many other domains such as sequencing human DNA (Churbanov & Winters-Hilt, 2008) and decoding GSM signal (Xie, Wen & Li, 2013).

1.2 Background

NLP is a research where natural language like English is taught for the machines (Fromm, 1998). This includes different kind of tasks like machine translation, question answering and speech recognition. One application using some of these NLP techniques is Apple Inc.'s mobile phone solution called Siri, which is voice-controlled personal assistant (Ibtimes.com, 2011). This speech recognition solution for instance, allows user to execute commands by speech. For example, the commands can be related asking the software to do calendar marks and to write and send emails. As well, it contains question answering ability, which is using question answering machine Wolfram | Alpha as a knowledge base (Readwrite.com).

Even though NLP is creating possibilities for more practical machine usage, it is creating other financial opportunities as well. For example sentiment analysis is an opinion mining study under NLP. The key goal in sentiment analysis is to analyze different trends from current conversations on Internet (Pak & Paroubek, 2010). This can be a handy guidance tool for the research and development departments in different corporations or it can be used as a prediction tool for a stock market trends (Bollen, Mao & Zeng, 2012). NLP could also be used to change some labor-intensive work fields towards capital-intensive production (Friedman, Shagina, Lussier & Hripcsak, 2004). Automated customer services are good example of this kind of transition, which is already in use on some companies. On the bottom line, the key goal behind all of the solutions is machines' capabilities to understand natural language on different kinds of contexts such as text or speech.

Since technology has created different possibilities for individuals to send messages and publish their thoughts via Internet, different new usages of standard languages have developed. These new writing patterns can be considered

as a totally new language since they obey different kinds rules than standard language. (Gimpel et al. 2011)

1.3 Focus Point of the Study

The main goals in NLP are to find a way for humans to interact with machines with ambiguous natural language. NLP is a key technology in many advanced systems. For example, it is one of the key goals in human-robot interaction. (Bai, 2009) As well, this implies to Human-Computer Interaction (HCI), which is a parent category of human-robot interaction.

The intelligence behind machines is based on models and algorithms, which are controlling the execution process. In order to make machines and humans to communicate with natural language, the machines have to be taught the rules of a natural language. NLP is a joint study of computer science and linguistics. (Yang & Liang, 2010) NLP development is highly important part to improve HCI and machine usability. Some NLP applications requires large amount of processing power, which can be overwhelming for portable machines. (Chien, Chen & Lee, 1993) This is because of the wide knowledge base and complexity of natural language (Langanke, 2008). As our empirical study shows on chapter 5.4.1, this is still true in HMM based n-gram annotation used with large background data. Main challenges of NLP systems are related to usability, speed and accuracy (Chien et al., 1993). Even though there are working NLP solutions on the market, for an end user the possibilities might seem a bit limited if the NLP scope is not familiar.

In most cases, solutions need Internet access if they are used from portable machines. This of course is depending on the final NLP application. For instance, spell checkers and auto corrections are good examples of NLP applications, which are in every day use on portable machines. (Swets.com) Still, solutions such as speech recognition in voice control solutions and machine translation are currently at least partly executed in cloud with computer clusters because they require more processing power. In some use cases the Internet connectivity might still be an issue, but in most industrialized countries internet is almost everywhere e.g. in coffee shops, in public transportation and mostly in people's cell phones. The local machine execution is moving towards cloud processing which means that heavier processing would not be an issue for a light weight devices (Kaku, 2011). Still, if the algorithms are too time consuming, it might have a huge impact on a processing cloud while multiple execution queries are lining up.

Machine knowledge of a natural language or the simulation of knowledge is based on a prior language analysis and how the analysis is stored in different corpora. Still, many of the studies and language corpora are based on standard language. This is mainly because the most of the analysis is based on news, science publications or books where the language is quite formal (Taylor et al., 2003, p. 5). Still, language is used informally in many occasions. Different dialects and slang are used in everyday conversations and in Internet as well. (Rit-

ter et al., 2011) For example automated simultaneous interpreter would need to have knowledge of slang in order to be helpful in average conversation. As well, Internet is full of informal text. Internet slang is widely used in different blogs, forums and instant message conversations. Since text publishing in Internet is possible with only light computer skills, misspellings are not rare either since people have different written capabilities. (Owoputi et al., 2013) In order to analyze this data, NLP studies need to have some focus on slang as well. The focus point of this study is to describe how machines understand written slang.

One way to describe machine interpretation of a natural language like English is to analyze how machines are automatically annotating text. The automated text annotation is done by dividing text tokens into different lexical groups. Lexical group labels are called Part-of-Speech (POS) tags. (Lv, 2010) Algorithm selection and corpus selection are important parts for POS tagger success (Dandapat et al., 2004). Different algorithms are proposed in certain use-cases. One of the most widely used algorithms for POS tagging process is HMM. HMM is continuously reported to have great performance on English language, but that does not necessarily apply well to other languages like Hindi for instance. (Hasan, 2007) HMM and different language model n-grams are widely used and compared algorithms for POS tagging because of their language independent behavior (Morwal & Jahan, 2013). Still, most of the studies related to slang tagging are focused on corpus development or improving domain specific methods (e.g. Gimpel et al., 2011). The goal of this study is to describe the most suitable algorithm for POS tagger in slang context. The research problem is how n-gram algorithms are performing in slang tagging compared to previously experimented algorithms.

1.4 Structure of the Thesis

Chapter 1 is a motivational introduction. It describes future possibilities and current uses of NLP in general. Chapter 2 describes the background information of NLP scope and basic evaluation methods used to describe efficiency of NLP solutions. Chapter 3 represents the computational process of POS tagging and how it is used to automatically analyze text. The chapter also covers previous experiments and the experiment results with different kind of use-cases. The chapter 3 describes background information of our experiment. POS tagging is executed with different algorithms. Chapter 4 describes HMM algorithm since HMM solution N-grams are tested in the empirical study. Chapter 5 describes our empirical study. The empirical study of POS tagging is done by analyzing execution and processing results of Natural Language Toolkit (NLTK) for Python in different use-cases and with different size n-gram algorithms. The study compares algorithm performance by analyzing online conversational messages in micro-blogging service Twitter (twitter.com) written in English. Since messages are written with informal and noisy language, it is considered as Internet slang. Because of the special characteristics of the slang, additional Python libraries and tools are used as well, which are represented in various

articles. Chapter 5 also includes an analysis of the findings of the empirical study. Conclusions of the whole study and suggestion of future work are represented in the chapter 7. Chapter 7 also covers how the findings of the study are affecting the NLP scope.

In conclusion, this study is focusing on POS tagging for written English Internet slang. Even though English language is used as key base of a natural language in this study, the focus point is still in multilingual algorithms and special characteristics of slang, which gives an ability to use findings of this study in different environments.

2 MACHINE INTERPRETATION OF A NATURAL LANGUAGES

This chapter describes the Natural Language Processing (NLP) domain and where it is applied in general. As well, this chapter describes basic evaluation techniques used in NLP systems and how the evaluations are usually counted. This chapter also represents an example of architecture for a commonly used NLP system.

2.1 Natural Language Processing

The key goal for NLP systems is to get machines to understand the natural language. NLP domain contains various kinds of tasks such as information extraction, speech recognition, optical character recognition, syntactic parsing or spam filtering. (e.g. Momtazi & Klakow, 2010) Even tools for spelling correction are NLP applications.

NLP is a layered process, which means it has various different computational subtasks depending on the actual NLP domain (Zin, 2009). Some NLP subtasks are general and they can be applied to solve different goals (e.g. HMM, or CRF), when some subtasks on the other hand are strictly related to certain purpose. For example lexical storages are commonly related to certain use (text or recordings) or to certain language[1]. (Derczynski et al., 2013) The basic principles for processing Internet slang are the same as processing standard language but there are some extensions. These differences are further discussed in chapter 3.6.

The computational language description is collected by gathering information of language use-cases and extending it with fine-grained meta-data based on prior language analysis (Francis et al., 1979). These kinds of meta-data sets are called corpus, which are more extensively described in the chapter 3.3. Still,

[1] Some language independent storages or storing methods exists. They are mostly used for languages, which do not have much data to compare for analyzing process.

usage of meta-data extension is not always necessary (Hasan, 2007). Contemporary NLP system architecture includes cloud based data storaging and processing as well. For example a cloud-based Application Programming Interface (API) service Mashape lists over 30 different types of NLP solution APIs, which it is hosting or providing. (Mashape.com, 2013)

Using a cloud services to support NLP applications can be a handy solution if some NLP applications require a lot of processing power. The processing power requirements vary between different NLP systems. Still for example speech recognition system are relatively slow to execute. (Chien et al. 1993) In this case, mobile usage of the application might be challenging without cloud support.

For example, Bird et al. (2009) describes different subtasks of a Spoken Dialogue System in architecture pipeline, which is represented in FIGURE 1. Simultaneous interpreter system would have quite similar pipeline and subtasks, which are represented in FIGURE 1. Still, that does not apply to all NLP applications. The key part is still that there are many different subtasks in many layers ehich all have different time complexity. FIGURE 1 includes five layers such as phonology, morphology, syntax, semantics and reasoning. Even when the time consumption of each layer is relatively small, the total time execution time might be noticeable because of the sequential execution process. FIGURE 1 shows that a spoken dialogue system have nine different stages. When each of the stages have multiple subtasks, it is not surprising that the total process might take time.

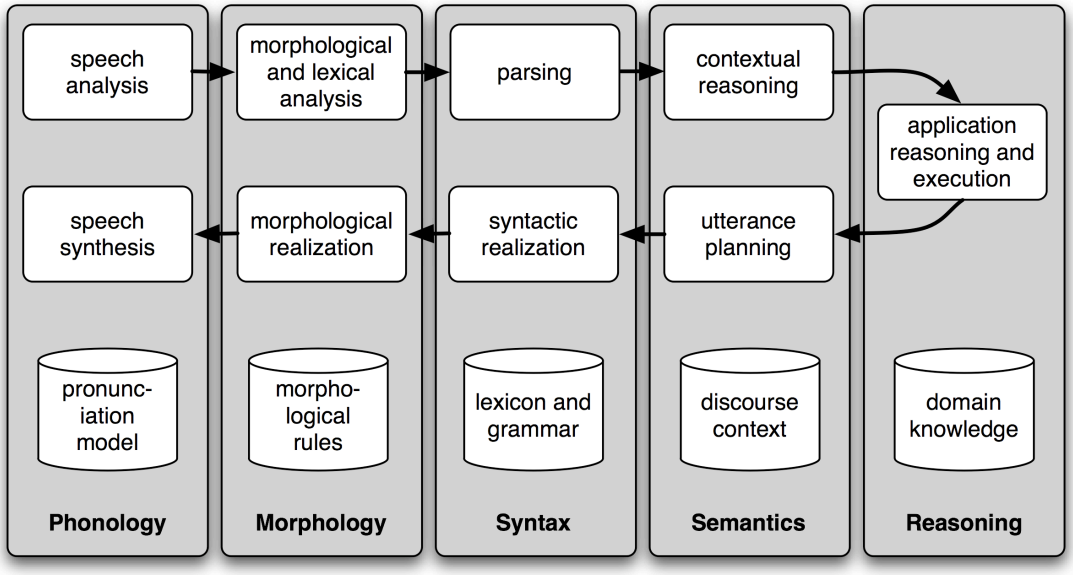


FIGURE 1: Architecture pipeline for a Spoken Dialogue System (Bird et al., 2009, 32)

2.2 Evaluation

This chapter covers the goal of evaluation of NLP systems and how evaluations are generally made. There are some differences in evaluation methods depend-

ing on the NLP scope. In general, accuracy is the largest issue in NLP system evaluation where accuracy represent NLP system success rate. The special characteristics of POS tagging system evaluation is described in chapter 3.4.

2.2.1 Evaluating NLP Systems

Evaluations of NLP tasks are necessary since there are accuracy problems in interpreting language representation whether it is speech or text. Further more, depending on subdomain of NLP system, there could be various numbers of application layers to do the processing. When the lower level processor in NLP pipeline is having accuracy problems it might multiply the problems to the final outcome in some case.

Efficiency is a key goal for a tagger. Efficiency can be measured with time consumption and accuracy. The time consumption of the whole system execution can be measured and represented in seconds. Even though time consumption of a NLP system is one part of the total efficiency, still the accuracy is mostly considered as a primary issue. NLP system accuracy is traditionally presented with two measures, which are precision and recall which are described in chapter 2.2.2. Evaluation methods have still occasionally some differences. Mainly the differences are caused by different NLP scope. (Mansouri et al., 2008)

2.2.2 Counting the Evaluation

Generally NLP system accuracy is described in three different values, where one of the values is combination of the other two. These values are called precision, recall and f-measure where the f-measure is the combinational value. The basic idea in counting these values is to evaluate the task outcome by categorize queried response to correct and incorrect matches and whether they are returned from the total resultset or not. This evaluation is done by first categorizing the responses in four groups which are true positive (TP), true negative (TN), false positive (FP) and false negative (FN). (e.g. Mansouri et al., 2008) In practice TP means number of correct responses, FP means number of incorrect responses, TN means number of correct responses left out from responses and FN means number of incorrect responses left out from responses. Still, FN is irrelevant value for counting precision or recall. These evaluation categories are used to calculate precision and recall. Precision (P) can be counted as

$$P = \frac{TP}{TP + FP}. \tag{7}$$

Recall (R) on the other hand, can be counted as

$$R = \frac{TP}{TP + TN}. \tag{8}$$

Commonly precision and recall are represented in one value called F-measure. F-measure gives good overall estimate of the accuracy. F-measure can be counted as

$$F = \frac{2RP}{R+P}. \tag{9}$$

F-measure is mostly used in cross-system evaluation as one weighted value. The measures are mainly adopted from information retrieval scope, but they are widely used in other tasks such as in Information Retrieval (IR) and Named-entity Recognition (NER). (Mansouri et al., 2008)

3 PART-OF-SPEECH TAGGING

This chapter describes POS tagging system meaning in a part of NLP system. The chapter describes general challenges and success factors of POS tagging and how POS taggers are using previous information as background knowledge for tagging decision-making.

3.1 Significance of a POS Tagger

POS tagging is one widely used computational approach to analyze text. It is an important part of many natural language processing tasks. For example POS tagging is used in speech recognition, speech synthesis, machine translation and information retrieval. (Lv, 2010) POS tagging is used as one processing phase if tagging is required in the NLP architecture. POS tags can be considered as word meta-data. In practice, POS tags are word class labels attached to word such as nouns, verbs and prepositions. The key goal in POS tagging is to categorize the words in a text sequence. (Zin, 2009)

Study of POS tagging is important part of AI as well. POS tagging is used in many NLP solutions, which are important part of AI field. Because POS tagging is a challenging task, it has been described as bottleneck of AI. (Schubert et al., 2003)

3.2 POS Tagger Pipeline

POS tagging is done by categorizing words according to their lexical usage in different groups. This procedure is giving a semantical meaning for processed words. In this sense, POS tagging is giving machines a way to understand text. The actual tagging is not tied to any algorithm but still some methods are much more popular than the others. (Denis & Sagot, 2012) Tagging methods can be divided into statistical and rule-based approaches. Both approach types are us-

ing probabilities on some level. Statistical approaches are tagging the words based on their lexical use while rule-based taggers are more focusing on the contextual use of words. (Lv, 2010) Statistical and rule-based methods perform really differently depending on the processed natural language. For example English language processing has its best performance with certain statistical approaches while Portuguese is having better results with rule-based approaches. (De Holanda Maia & Xexéo, 2011) Still the most effecting part for tagger success is the size of the corpus (Banko & Moore, 2004).

FIGURE 2 is a reconstructed image of POS pipeline as Zin (2009) describes it. The goal for POS tagger is to divide words in different lexical groups from raw text. As seen in the FIGURE 2 the input text is segmented first which means that different words are separated from each other. In English language this procedure is mostly just separating words on each white space. (Zin, 2009)

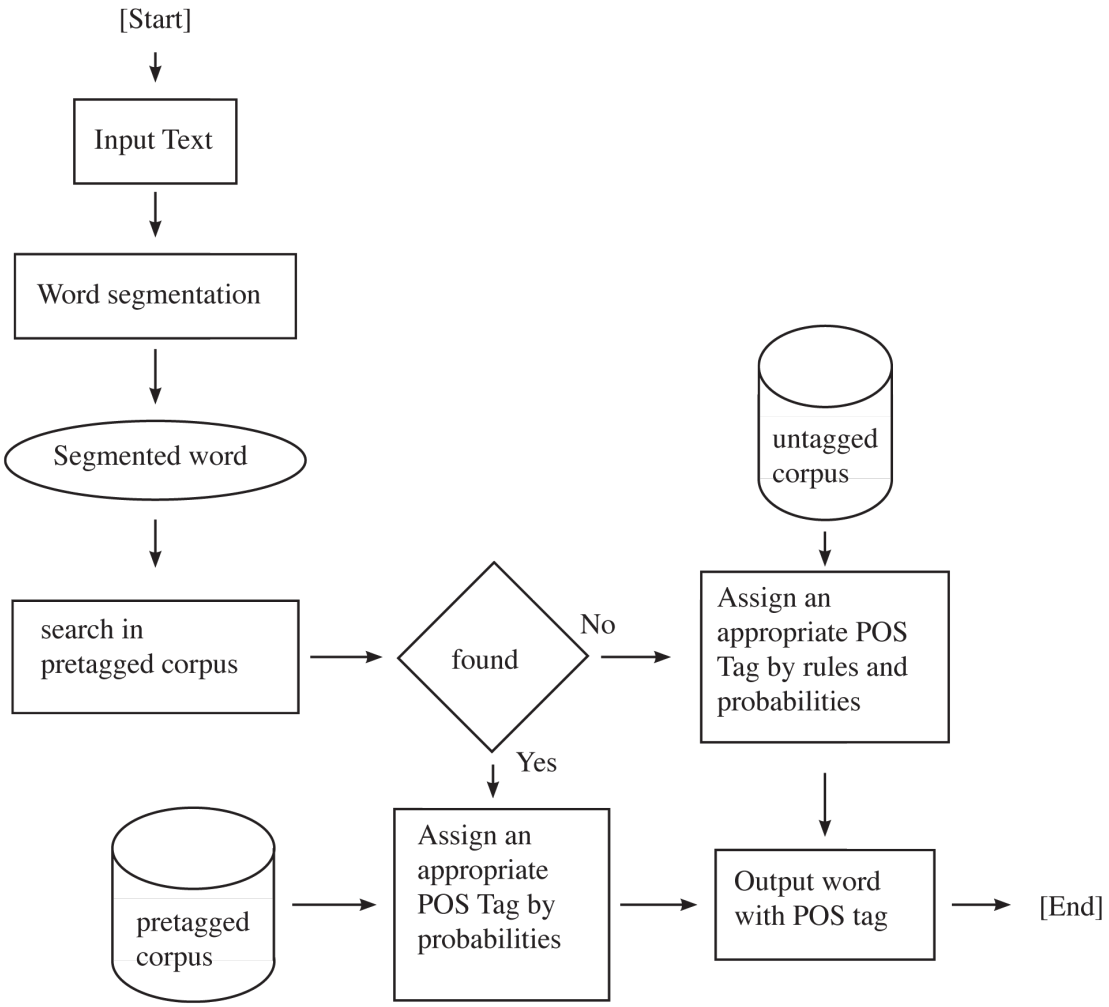


FIGURE 2: Part-of-Speech tagger pipeline (based on Zin, 2009)

For some languages like for Myanmar for instance, the segmentation is a bit harder process since the words are written together without white space. After this, the word annotation is looked up from pre-tagged corpus for each word. The POS tag is given for each word no matter if the word is found from

the corpus or not. Only different algorithm is used on either case. As well, some of the text token may be left without a tag if text tokens are noisy or the tags cannot be reasoned for some other reason. (Zin, 2009)

3.3 Corpus

This chapter describes meaning of text corpus and how it is used in POS tagging applications. As well, this chapter describes differences between different corpora and how they are describing ways to use a natural language to computers. The chapter also represents challenges and limitations related to corpus usage.

3.3.1 Importance of a Corpus

Tagging is always based on previous data on some level. Sometimes rule-based approaches do not use prior training data. In such cases language rules are taught to the tagger. (Brill, 1995) Different methods could still use the same data really differently or the stored data could have really different kind of structure comparing to other methods. One approach is to use lexical data sets called corpus, which are specifically made for POS tagging. A corpus can contain just untagged sentences or it can be pre-tagged as well. Corpus is a sort of data store of a language, which purpose is to describe a natural language.

Corpus can be used in multiple occasions and it can be constructed for a specific purpose. (Zin, 2009) As well, there are different corpora or different corpus parts for speech and text. Different usages can be related to specific topic or specific domain such as speech recognition or informal text messages like Twitter messages (e.g. Gimpel et al. 2011). Since this study is related to written text, the focus point in corpus would be in text related tagging as well. Corpus can be pre-tagged or untagged regarding the POS tags where pre-tagged corpus has the linguistic definitions of each word to extend the plain text. In general, a corpus is wide collection of word sequences, which have been previously used in actual context. For example Corpus of Contemporary American English contains over 450 million words from various sources, which are either spoken (Radio or TV), fiction, popular magazines, newspapers or academic journals. The larger corpus the better it describes the word usage for the NLP machine. (Zin, 2009) Still this applies only to a certain point since the size does not always guarantee higher recall rate. As the corpus size increases enough, at a certain point success rate of NLP system might start to decrease. Decreasing is due to noisy data or atypical language for the context. (Liu, 2003)

Corpus is only a way to store previously used language to structured training data. In this sense corpus can be referred as predetermined training set. There are different models for the usage of corpus data. HMM, Support Vector Machines (SVM) and Conditional Random Fields (CRF) are commonly used models in tagging process. Chapter 4 describes the HMM in more accurate level

since its really commonly used for POS tagging and it provides simple solutions to common tagging problems and it has widely used variation as well. (Ekbal et al., 2007)

3.3.2 Corpus Usage in POS tagging

Word labeling in POS tagging is done by categorizing each word in a sentence to different lexical groups such as nouns or verbs. In computer processing there are different abbreviations for each word group. For example widely referred Brown Corpus that was originally composed in 1964 is marking singular or mass noun as NN and verb in base form as VB. (Francis et al., 1979) The example (1) represents POS tagged sentence as Natural Language Toolkit (NLTK) toolkit is tagging the sentence.

(1)	DT	NN	VBD	PRP	DT	NN
	A	police	gave	me	a	fine.

The example sentence contains for example a singular or mass noun (NN), a verb in past tense (VBD) and a determiner (DT). The example is processed with NLTK tool for Python using basic configuration, which is using Penn TreeBank tag set as corpus. Possible Penn TreeBank tags are presented in attachment 1. (Taylor et al., 2003) Form of the word class tags and appearance amount in a single corpus varies between different corpora. For example, while Penn TreeBank tag set has 35 different tags, Brown Corpus uses 226 different kinds of tags (Francis et al. 1979). On contrary, slang corpus only uses 24 tags (Gimpel et al., 2011). Fine-grained corpus might be a smarter choice when POS tagger is used as a part of translation application for instance, since coarser tag set could affect the tagging accuracy (Derczynski et al., 2013).

Pre-tagged corpora have differences in marking word classes. Corpus can either contain tagged sentences or they can be Treebank, which are describing the sentence structure on more concrete level. Penn TreeBank is one common example of Treebank used in POS tagging systems. FIGURE 3 shows an example of a sentence constructed as tagged tree. The original example sentence stored to corpus as represented in Schubert and Tong (2003) is:

> "Rilly or Glendora had entered her room while she slept, bringing back her washed clothes."

As it the sentence is formatted to a Treebank form it contains information about the sentence structure. The basic idea is that the basic sentence (S) contains smaller sentence fragments (POS classes), which again might contain other fragments. The lowest level of the tree contains tag connections to actual words, as can be seen on FIGURE 3. When there is enough information about sentence structures, statistical tagging algorithms are able to reason the most possible tags for given text based on previous word and tag connections. (Schubert et al., 2003)

```
((S
  (NP
    (NP (NNP Rilly) )
    (CC or)
    (NP (NNP Glendora) ))
  (AUX (VBD had) )
  (VP (VBN entered)
    (NP (PRP\$ her) (NN room) ))
  (SBAR (IN while)
    (S
      (NP (PRP she) )
      (VP (VBD slept) )))
  (\, \,)
  (S
    (NP (\-NONE\- \*) )
    (VP (VBG bringing)
      (PRT (RB back) )
      (NP (PRP\$ her) (JJ washed) (NNS clothes) ))))
(\. \.) )
```

FIGURE 3: An example of a sentence in a Treebank (Schubert et al., 2003)

3.3.3 Challenges in Corpus Usage

The relation of a corpus to a specific domain implies that corpora are mostly language dependent. This includes usage of slang words and different dialects, which might cause erroneous tagging or some words might be left without a tag. This is the reason why corpus should have similar writing style than the processed text. As well, for example Brown Corpus is categorized in different text styles, which increases the tagging accuracy while used with similar annotation problems. Another problem is correctness of the processed text. Since the processed texts are mostly written by humans it creates the possibility for source based human error. (Gupta et al., 2013) These kinds of errors are commonly considered as noise. The term noise is used since it causes similar problems as processing noisy sounds from a voice. There are techniques to avoid noise related problems. One solution is to detect and correct spelling mistakes by using corpus, which contains common errors. (Agarwal et al., 2007) Spelling correction is still its own topic under NLP and it is left out since it is only a minor issue in this study.

In some cases it is possible that there are errors in corpus. After all the annotation is done at least partly by humans, which creates an opportunity to human error. (Loftsson, 2009) For example according to Nguyen (2011) some parts of Penn TreeBank n-grams have incorrect labels. This is effecting to machine representation of language and it might cause problems in later use in NLP system. In slang context, corpora used in annotation are known to be noisy, since the usage scope is much more informal than a standard language. Even though there are methods to mitigate this problem, it decreases the accuracy on a minor level (Owoputi et al. 2013).

3.4 POS Tagging Evaluation

Evaluating success of a POS tagger has some differences comparing to evaluating methods used in other NLP systems. Precision value is considered to be the most useful factor in POS tagger evaluation and typically recall value is ruled out. (Dandapat et al., 2004) Since, the F-measure is not typically counted for POS tagging systems, different sources are referring to precision value with different names or they are simply referring to accuracy rate. Sometimes recall rate is referred to as the precision value (Gimpel et al. 2011). In conclusion, POS tagger accuracy is in general counted as number of correct responses divided by number of all possible responses, which directly refers the precision value (Mansouri et al., 2008). In practice, accuracy testing needs manual work with corpus which can be separate in training data and test data. Testing practices are described in chapter 3.6 and in chapter 5.3.

3.5 Previous Work

This section represents previous studies related to POS tagging. The chapter focuses to describe annotation differences under different domains. As well, this chapter describes current issues in slang annotation.

3.5.1 Standard Natural Language Annotation

Traditionally POS tagging methods are divided in rule-based approaches and statistical approaches which both have their own pros and cons (Lv, 2010). Usually statistical methods can be directly applied to other languages but prior training data is required. (Denis & Sagot, 2012) In most cases in statistical approaches prior manual tagging for training data should be really extensive which requires months of manual labor. For example HMM does not perform well with a small amount of training data. (Ekbal et al., 2007) On the other hand, rule-based approaches might not need any prior annotation but the language rules have to be taught to the tagger. (Brill, 1995)

Tagging formal English language has not gained much accuracy since the mid 1990's. Brill's (1995) rule-based tagger already had near 97% of success rate. As well, statistical approaches got similar results around at the same time. (Jung et al., 1996) English language and most of the other European languages (Ekbal et al., 2007) along with Chinese (Banchs & Codina, 2009) have extensive prior studies in POS tagging. European languages have been mostly reported to have great accuracy with statistical approaches. For example French (Denis & Sagot, 2012) and English (Jung et al., 1996) tagging has been reported to have 96-98% accuracy with statistical approaches.

Still, POS tagging studies during latest decades have extended the knowledge of the domain since studies have focus point in other languages as well.

(e.g. Zin, 2009) Even though English language POS tagging is not solved problem yet (Manning, 2011), the extensive prior work around English language have given a good ground to be used in other domains (Banchs & Codina, 2009). For example Amazigh POS taggers are already getting 99.97% accuracy rate (Outahajala, 2013).

This is a good direction for NLP development since it creates more possibilities for translingual or translation softwares and tools. In general, even though accuracy of POS tagging systems has increased, the execution times have still decreased. (Owoputi et al. 2013)

3.5.2 Slang Annotation Experiments

As a comparison between English and other languages, English slang POS taggers are reaching similar accuracy rates as some formal language taggers. For instance, statistical approaches for Marathi language are reaching accuracies from 77% to 93% depending on the selected algorithm (Singh et al., 2013), whereas contemporary English slang taggers have accuracy rates between 90% and 93% (Owoputi et al., 2013).

HMM has been applied to various languages. For instance, it has been used in European languages like English (Banko & Moore, 2004), French (Denis & Sagot, 2012) and Portuguese (de Holanda Maia & Xexéo, 2011), as well it has been used for Asian languages such as Hindi, Telegu, Bangla (Hasan, 2007), Punjabi (Sharma, 2011), Marathi (Singh et al., 2013) and Myanmar (Zin, 2009). Still, HMM and its n-gram solutions are performing really differently in different languages. For example HMM on the Telegu is reaching only 56.6% accuracy rate (Hasan, 2007), while English (Banko & Moore, 2004) and Myanmar (Zin, 2009) are reaching over 96% accuracy rates. On the other hand, some languages like Portuguese performs better with rule-based approaches (de Holanda Maia et al., 2011).

In conclusion algorithm selection is an important part of POS tagger accuracy. Since different algorithms perform better under different domains, the algorithm suitability should be studied in each case. Even though rule-based approaches outperform statistical approaches in some cases, still the performance differences are quite small. Mostly used statistical approach HMM performs relatively well in most languages comparing to other approaches if the prior manually annotated data collections are large enough. Automatic annotation faces lack of performance in languages with loose word ordering (Hasan, 2007).

3.6 Special Characteristics of Internet Slang Tagging

Recently the usage of written slang has notably grown because of different social media platforms. This creates different possibilities for NLP solutions because services like Twitter (twitter.com) are offering their data for mining pur-

poses (Huberman, Romero, & Wu, 2009). Some NLP solutions are already made using the platform such as sentiment analysis services. (Kouloumpis et al., 2011)

Key differences with formal text and Internet slang are the fact that slang contains noisiness and informal writing style. Internet slang is mostly used in social media status updates, blog posts or instant messages. Considering the whole NLP perspective, slang processing is hard since sometimes messages are quite short, and message context is really hard to figure out. English have notable differences between slang and formal language. As well, English slang has context dependent characteristics. For example some times some words are dropped out and services like Twitter have their own feature markup such as hashtags and emoticons. (Ritter et al., 2011)

When the use of grammar is not strict, it creates lot of new ambiguity problems compared to standard language. One solution to resolve noisy data for tagging purposes is to cluster words in groups and to look correspondence for word and tag in each cluster. Clustering is done by grouping words with the same meaning into a single cluster. FIGURE 4 shows an example of a cluster for a word ”tomorrow”. (Ritter et al., 2011) If we look at the example cluster FIGURE 4, it can be easily seen that written forms of a slang word might cause a lot of new polysemy problems. For instance, the word ”tomorrow” can be written as ”tmw” which as well can be an abbreviation from something else depending on the context. Still the idea of word clustering is only to be used as an additional tool to help tagger in decision-making process (Owoputi et al., 2013).

‘2m’, ‘2ma’, ‘2mar’, ‘2mara’, ‘2maro’, ‘2marrow’, ‘2mor’, ‘2mora’, ‘2moro’, ‘2morow’, ‘2morr’, ‘2morro’, ‘2morrow’, ‘2moz’, ‘2mr’, ‘2mro’, ‘2mrrw’, ‘2mrw’, ‘2mw’, ‘tmmrw’, ‘tmo’, ‘tmoro’, ‘tmorrow’, ‘tmoz’, ‘tmr’, ‘tmro’, ‘tmrow’, ‘tmrrow’, ‘tmrrw’, ‘tmrw’, ‘tmrww’, ‘tmw’, ‘tomaro’, ‘tomarow’, ‘tomarro’, ‘tomarrow’, ‘tomm’, ‘tommarow’, ‘tommarrow’, ‘tommoro’, ‘tommorow’, ‘tommorrow’, ‘tommorw’, ‘tommrow’, ‘tomo’, ‘tomolo’, ‘tomoro’, ‘tomorow’, ‘tomorro’, ‘tomorrw’, ‘tomoz’, ‘tomrw’, ‘tomz’

FIGURE 4: An example of a word cluster (Ritter et al., 2011)

Gimpel et al. (2011) are detecting additional spelling related issues, which are more fine-grained problems than considering all spelling mistakes as noise. Phonetic normalization is one of their methods to detect different writing forms

of a single word. Still, Owoputi et al.'s (2013) updates Gimpel et al.'s (2011) study with a couple of ways related to test data processing and data collection. One solution is to cluster the possible spelling errors from the input data. Another improvement is in data collection method. Practically all the messages with uncommon words are filtered out. These updates seem to cover all the issues related to spelling. Another issue in automatic slang annotation is a proper noun detection. Gimpel et al. (2011) are solving this problem by listing all nouns occurring in the test data and comparing unrecognized word tokens to the list.

One problem for automatically annotating slang is currently how the languages are developing. Languages and more closely slangs might be developing faster than words are being annotated to corpus. Since Internet has made updating and sharing information popular in short messages such as status updates and microblog posts, it gives a fast ground to spread new words or abbreviations which can be based on current news or technologies etc. (Bird et al., 2009)

3.7 Challenges

This chapter describes the most common challenges of POS tagging. The chapter covers challenges related to tag ambiguity, accuracy problems of longer sentences and special characteristics of slang.

3.7.1 Ambiguity

The major problem in POS tagging is ambiguity of the words. This is due to synonyms, homonyms and other polysemy in general. The example (2) contains a word ″fine″ as previous examples but this time the meaning of the word is different and so is the categorization of the word, which in this case is tagged as adjective (JJ).

(2)	PRP	VBP	NN
	Everything	is	fine.

With basic settings NLTK toolkit faces the problem in the word ambiguity in given example sentence (2). Without configuration the word fine is tagged as noun (NN) even though it is obvious that in this context sentence the word ″fine″ should be tagged as adjective. The simple solution to this is to look at the surrounding words. This means that the example (2) is tagged with unigram model. Simple solution is to look surrounding words and use bigram model for tagging instead. Example (3) shows bigram tagging for the same sentence. A bigram tagger considers occurrence probabilities based on two token queues where unigram is considering only the probability of a single word occurrence.

(3)	PN	BEZ	JJ
	Everything	is	fine.

The higher the n-gram level the slower the tagging is to execute on a computer. The same applies for a larger corpus and it is affecting significantly the memory requirement as well. This is why it is encouraged to detect only local relations on a sentence. (Chien et al., 1993) The n-gram principles are more accurately described in chapter 4.2.

Some of the key tasks in NLP do not produce 100% accurate results. In most cases this is caused by ambiguity of the words but as an extension there are different minor issues as well to effect the language processing. Idioms and other language related linguistic phenomenons are affecting the interpretation of given issue. As well, author's vocabulary might differ from target audience. (Kowalski, 2011, p. 3) This is not only a machine related problem since human communication is also based on the idea that message receiver is trying to figure out the best probable interpretation of the message from the sender. One minor issue affecting the final interpretation of a message is noise, which can be for example regular noise on a speech and mispronounced or miswritten word. (Derczynski et al., 2013)

3.7.2 Sentence Accuracy

It have been reported in many studies that current POS taggers are tagging words in English texts with over 97% of accuracy. In most cases, correct tagging is counted only by looking at word tagging. Complete sentences are still a bit problematic since they are mostly reaching accuracies of only 55-57%. This is aligned with accuracy of tagging one word if we assume 20-word sentence. Then the probability for whole sentence to be tagged correctly would be .9720^20 $\approx$ 54% . In some cases this could mean that POS tagger are missing the key meaning of the tagged sentence. Tagging slang is even more difficult since the best taggers are reaching accuracy only 93%. In a 20 word sentence this would mean that the probability for to get the whole sentence with correct tags is only .9320 $\approx$ 23% . The difference is partly caused by the way to count the correctness since punctuation marks and special characters are getting tagged as well, which are quite unambiguous. (Manning, 2011)

Related to formal text written in English, Manning (2011) notes that human annotators disagrees more often about tagging decisions made by other human annotators than tagging decisions made by automatic taggers. This of course gives a perspective of the possible performance of a tagger. Even though the success rate might sound good, the problems might sum up later on in the NLP system. For example, when POS tagging is used in machine translation, the incorrect tagging is directly affecting the translated outcome and it might cause unexpected results.

3.7.3 Informal Data

Even if formal text tagging is quite insufficient for complete sentences, the tagging of informal conversation could be even more insufficient than actual reported results. Currently highest slang tagging accuracy has been reported to be near 93%. On the upside slang taggers are using additional methods, which are improving the accuracy for the particular use-case (Derczynski et al., 2013). As well, slang sentences in conversations tend to be shorter than in informal texts, which might increase the success rate. The basic idea of a tagger is to annotate words based on the probability, how likely it is for a word to have certain tag with given surrounding words having counted or given tagging. This is one reason why the probabilities that taggers are counting are only fractions of a percentage. In a way, if we consider the whole sentence to be context for a certain tag, the accuracy of tagging sentence with 55% is relatively good.

Even though POS tagging is problematic, the success level could be good enough depending on the final application. Even though some POS tags might be incorrect, it can be still helpful in decision making for other applications. For example Named Entity Recognition (NER), which is part of information extraction, is trying to find useful information from unstructured text by classifying nouns in a document as person name, location name, organization name etc. (Biswas, 2009). NER is trying to resolve text content by trying to answer questions such as “who”, “where” and “when”. Under this domain, NER has its own methods but still POS tagging can be used as a lower level tagger to annotate text and help NER in decision-making (Biswas, 2009). Considering slang especially in microblogs or in instant messages the annotation is even more problematic than on formal text. The messages are usually short, and comparing to formal English some words are missing. As well the rest of the words commonly have a lot of spelling mistakes or they are intentionally written in different form than they should in a formal context. Despite the problems, different methods are proposed to fix these problems (Ritter et al., 2011).

Accuracy rates of slang annotation in different studies are notably lower than in formal language. Studies during recent years have took huge steps in English slang tagging. In some cases English slang tagging reaches better accuracies than some formal languages. Still slang tagging has its own challenges as well. Even though word ambiguity causes some accuracy problems, another issue is to detect common and proper nouns. As well, since Internet slang is used differently in different places, there are domain related issues to be solved as well, such as specific mark-up. (Owoputi et al., 2013)

3.8 Conclusion

In conclusion, POS tagging is one important application layer in NLP system (Bird et al., 2009). The goal of POS tagging is to automatically annotate words (Zin, 2009). Different algorithms are used for tagging process and they can be divided into statistical and rule-based approaches (Lv, 2010). POS tagging algo-

rithms are not 100% accurate, but with a right algorithm selection taggers can reach over 95% accuracy rate (Manning, 2011).

Written conversational slang has own characteristics and different additional methods have been applied to slang tagging to gain more tagging accuracy. One successful tool for slang POS tagging is to use word clusters, which are libraries of words with many different writing styles. (Gimpel et al., 2011)

4 HIDDEN MARKOV MODEL

This chapter describes HMM algorithm's working principle and how the algorithm is counting tagging probabilities. As an addition, this chapter describes popular HMM implementation called Viterbi algorithm and how it is used to count probabilities based on HMM. As well, this chapter describes more closely how and why HMM is used as POS tagging algorithm. This is due to experimental part in chapter 5, where different size HMM-based N-grams experiments are described.

4.1 HMM in General

HMM is one widely used statistical annotation algorithm, which have high performance with different natural languages such as Standard English (Banko & Moore, 2004). Viterbi algorithm is one successful implementation of HMM (Forney, 1979). HMM is used under many different domains. It is used partly in sequencing human DNA (Churbanov et al., 2008), decoding GSM signals (Xie et al., 2013) and in different language processing tasks (e.g. Alpaydin, 2010). HMM is a mathematical model to define transition probabilities from unobservable state changes. In other words, HMM is a layered stochastic process having non-observable layer, which can be only defined with another set of stochastic processes, where the second layer is defined by observations. (Rabiner & Juang, 1986) For example, it is possible to count word occurrence probabilities from a certain text, as well it is possible to define how likely two words are appearing together or within a certain group. Further more, words have different likelihoods to appear with certain kinds of words meaning certain kinds of word classes. Even though these kinds of occurrences and connections can be counted, it is impossible to see or count the state transitions behind the whole writing process. This is why most of the compared probabilities are relatively low numbers and connections are usually considered within one sentence. Nevertheless, the key idea for HMM is to observe language usage patterns which in NLP context can be used to define rules in a natural language. These

patterns in HMM context are referred to as states and state transitions. The transitions are listed in lexical libraries called corpus, which are used to count statistical information based on transition between different word classes or based on tag-to-word connections. (Alpaydin, 2010)

4.2 Operating Principle

Even though HMM is its own model, it is also used to refer to a collection of different models since they all have a common operating principle. Likewise, HMM it self is adopted from Markov chains with the extension of usage of hidden states. (Kuo, 2006) Even though HMM is widely used in NLP systems, it has been applied in many other different areas such as bioinformatics (Alpaydin, 2010) and gesture recognition (Takano, 2012). Most commonly used HMM related models are Bayesian networks, CRF and Maximum-Entropy Hidden Markov Model (MEHMM). Overall, shortly to describe related models, HMM algorithms can be represented as Bayesian network while counting transition probabilities between unobservable groups. (Kuo, 2006)

CRF is partly similar model than HMM with many common principles but CRF is more linear model (Awasthi et al., 2006) and performs better on certain scope. (Wen-qiu, 2012) MEHMM, which is as well referred later, combines HMM and maximum-entropy model by using observed states as well when counting probabilities. Where HMM probability depends on the current state, MEHMM depends on previous states as well, which increases the performance on precision and recall. (McCallum et al., 2000) With N states and T transitions counting MEHMM might be time complexity especially when the source is large and the time consumption is counted as TN^T. MEHMM decreases the time consumption of the reasoning algorithm since it rules out many recursive iterations and changes the running time to TN^2. (Allen, 1995)

HMM is widely used in many different NLP applications. For example Liu (2003) lists the following NLP domains where HMM is in important role: part of speech tagging, speech recognition, tag segmentation, topic detection and information extraction. Even though the basic Markov chain has been developed in the early 20th century and the groundwork of NLP is based on the theories and studies created on 1960s, 1970s and 1980s (Rabiner & Juang, 1986). Still, it has been only recently applied to practical use because of limited computational processing power (Kuo, 2006). For instance, in the early 1990s sentence recognition from speech took time from couple of seconds to dozens of seconds even though lexicons and unigram bases in use were relatively small (Chien et al., 1993).

As mentioned, HMM contains Markov chain where the states are hidden and state transitions are estimated from visible observations. FIGURE 5 shows a possible HMM reasoning where X represents state, Y represents the observed outcome, a_{12} and a_{23} are representing possibilities for each transition and b_1,…, b_3 represents unigram probabilities for a state to become the related outcome. Compared to example (1) Y_1, Y_2 and Y_3 could represent words in the sen-

tence ”Everything is fine”. In this example the states and the transitions are hidden as it is assumed in HMM, since the sentence construction process is unknown.

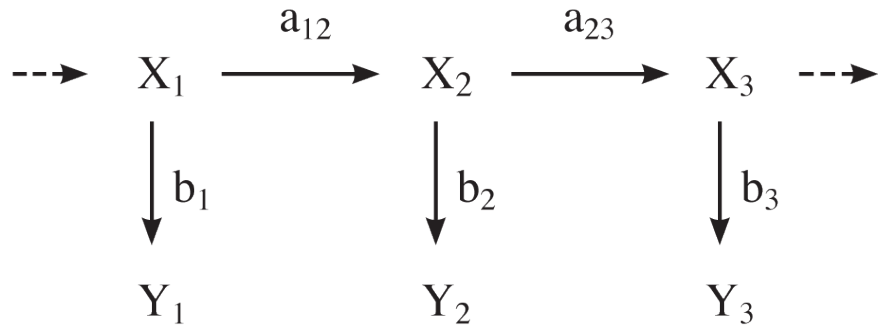


FIGURE 5: Hidden Markov Model Process

Language model N-gram taggers, such as bigram tagger for instance, assumes that word class depends only on the word it is bound to and on a previous tag. Traditionally HMM is tagging words in the same way since N-gram algorithms are HMM implementations in a limited environment. Compared to n-grams, HMM has been extended to consider future tags as well. (Banko & Moore, 2004)

Even though the HMM states are hidden at the start, the state transitions can be counted from observations. It is assumed that the hidden states are actual word classes. In other words, tags are assumed to be hidden states in HMM. FIGURE 6 describes hidden state transitions. As corpus contains manually tagged word to tag connections, it is possible to connect the words to actual states. (Forney, 1973)

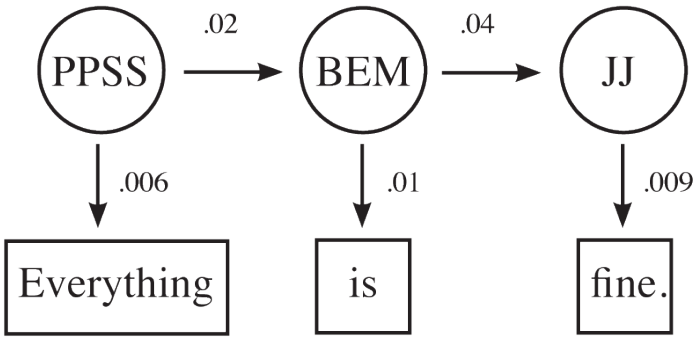


FIGURE 6: POS tags placed in Hidden Markov Model with transition and unigram probabilities with visible transition states

Still some words are ambiguous, since they might have multiple meanings used in different context. Since corpus contains word tagging, it is easy to count likelihood of a different tag for each word. This probability is the same, which is generating the node from connected state. In practice, counting the likelihood

means listing all possible unigram tags for each word. After the possible tags are listed for each word, the state transition probabilities are be fetched from a corpus. This can be done since observable state transitions can be counted, which can be used to estimate the state transitions behind the words, since different word classes have different likelihoods to appear with other classes. (Forney, 1973)

Finally, the most probable route for different state transitions can be counted by using transition and unigram probabilities. FIGURE 7 shows an example listing of possible tags for each word and possible state transitions. The transition probabilities are shown as examples of possible transitions in a possible corpus. The tags are based on NLTK tagger using Brown Corpus. Circles are representing possible states, in this case word class tags, which each are linked to observed together appearing words below. Even though FIGURE 7 is a simplified example with only four observation words and with two possible tag states, still there are already sixteen possible ways to construct the tagging sequence. Possible number of sequences grows exponentially as possibilities of tags grow. As well, with four word class possibilities for each word, the amount of possible routes would be 256. This explains the non-linear time consumption of the algorithm.

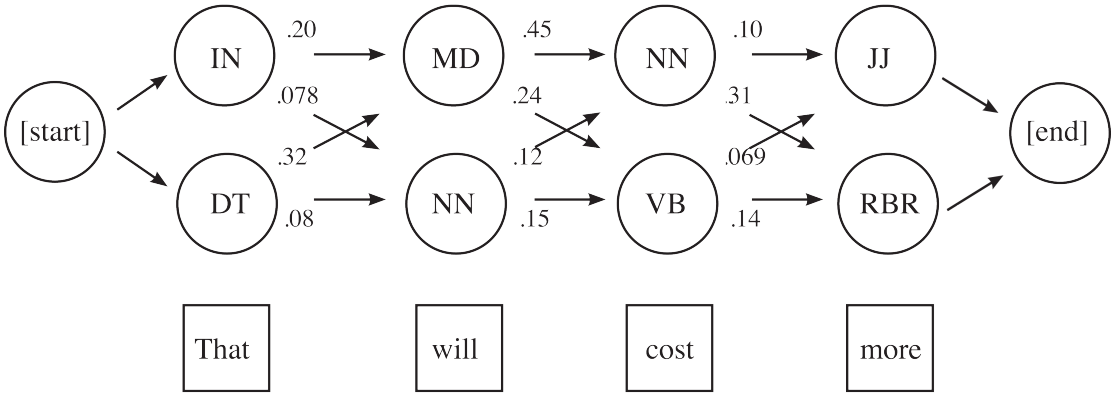


FIGURE 7: A forward trellis of tagging possibilities with example transition probabilities

As a comparison to language modeling, general HMM is continuously reported to have slightly better accuracy than unigram, bigram or trigram taggers even tough the basic principle of n-gram algorithms are based on HMM. (Singh et al., 2013) Even though HMM gives more accurate tagging results with English language, it has inefficiency problem considering exponential time complexity because of the algorithm as described earlier in chapter 4.2. (Kuo, 2006) In this sense, n-gram tagging algorithms might have notably smaller time consumption. As well, it has been studied that in some point growing the maximum n-gram size does not give more accurate results. Even some accuracy decreasing might happen as the n-gram size increases, which is mostly fault of noisy text and word ambiguity. De Holanda Maia and Xexéo (2011) suggest that the accuracy does not grow after the maximum n-gram size has grown to five. The n-gram size impact on the accuracy is described in the attachment 2. In

some use cases, faster algorithm might be preferred over more accurate tagging algorithm especially when the accuracy differences are relatively small. This efficiency difference is tested more closely on the chapter 5. As well, Owoputi et al. (2013) notes that efficient algorithms are highly important in slang POS tagging context since at least in slang context the amount of annotatable data is all the time rapidly increasing and real time result for the final system might be needed.

4.3 Viterbi Algorithm

In extensions for HMM there are different algorithm implementations to calculate the most probable sequence word class transitions. The most common HMM extension is called Viterbi algorithm, which considers only the most possible transitions from each state in given context before continuing to next stages. (Chao, 1993) The Viterbi algorithm is one supervised-learning algorithm, which needs corpus for tagging background data. Based on HMM, Viterbi decoding considers full sentences to be annotatable paths instead of limited amount of transitions like n-grams. As well, there are HMM-based unsupervised tagging algorithms such as Baum-Welch, which can train HMM from without prior knowledge of tagging statistics (Hasan, 2007).

Viterbi algorithm is trying to find the most probable path from trellis graph. Running the HMM with Viterbi is less time consuming since it only considers unbreakable chain structure. FIGURE 8 is Forney's 1973 Viterbi algorithm example with a modification that the algorithm tries to find most probable route from trellis. The first stage (a) is an example description of full reasoning trellis with all possible transition routes. Numbers between transitions describes the cost of a transition, which in this case means probabilities where lower number means a higher probability. Originally the numbers are describing the cost of transition. On the stage (b) the previous stage (a) is divided in five parts where each part describes a transition. On each transition only the most probable transition are taken into account. (Forney, 1973)

As an addition, Viterbi algorithm has been efficiently used with MEHMM as well. MEHMM is optimizing the Viterbi trellis path better than plain HMM. Even though this affects slightly to the accuracy, it is notably less time consuming option, because of the greedy decoding of MEHMM. (Owoputi et al., 2013)

4.4 Counting HMM Likelihood

HMM is based on counting probabilities of hidden states based on linked observations. For to be able to explain HMM and also N-grams, the probability calculations behind word classes and word class transition should be understood as well. As stated earlier, HMM can be interpreted as Bayesian network

which enables to use Bayes' theorem to count the probability of each tag for given word. (Kuo, 2006)

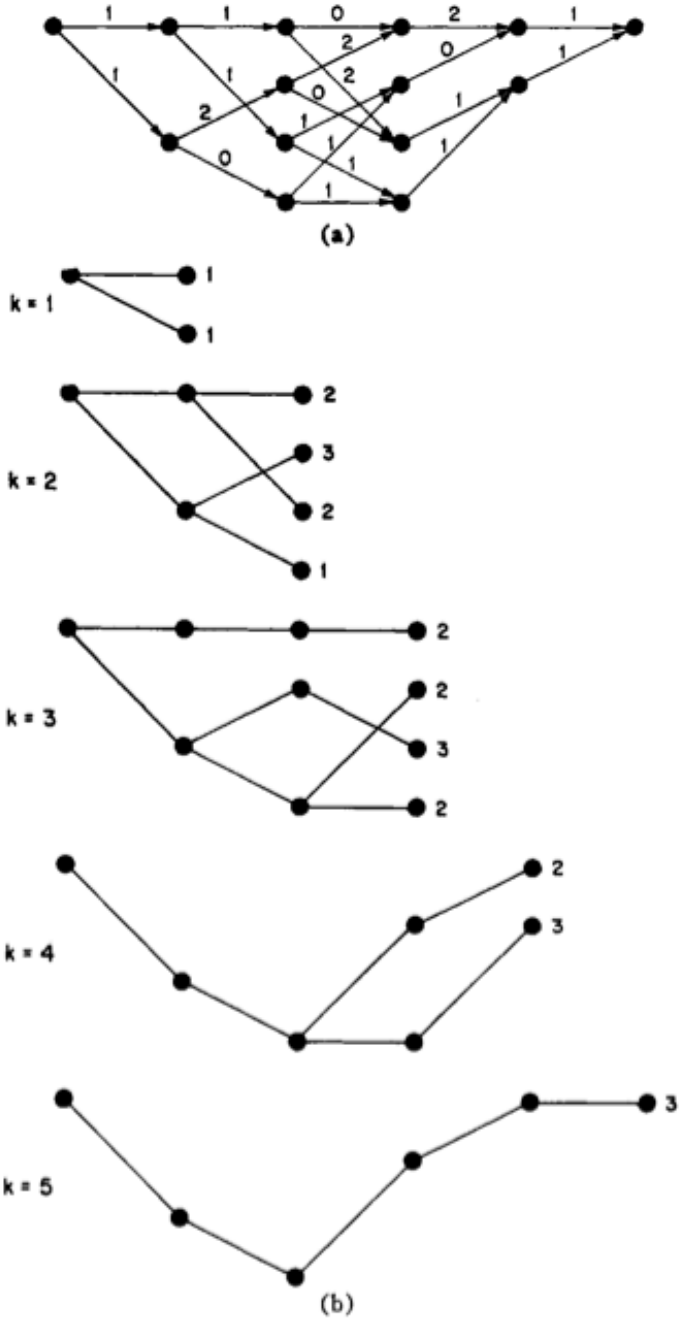


FIGURE 8: Viterbi algorithm reasoning (Forney, 1973)

Let

t_1^n = Sequence of tags ($t_1..t_n$) and

w_1^n = Sequence of observed words (w_1 .. w_n).

A word can have different tag depending on their lexical meaning. Since some words are ambiguous there are many different possible solutions to construct the tag sequence based on the word sequence. The main goal is to find

the most likely tag sequence $\tau(w_1^n)$ with given word sequence w_1^n by using corpus as background knowledge. More formally

$$\tau(w_1^n) = \arg\max_{t_1^n} P(t_1^n|w_1^n). \tag{1}$$

Probability for a tag is depending on a given word, which can be presented with Bayes' theorem. Kuo (2006) represents the problem applied to the Bayes' theorem as

$$P(t_1^n|w_1^n) = \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}. \tag{2}$$

Since the word sequence is already known, the normalizing factor $P(w_1^n)$ in Bayes' theorem can be assumed as constant. In practice, the normalizing factor can be ruled out, which means that

$$P(t_1^n|w_1^n) = P(w_1^n|t_1^n)P(t_1^n). \tag{3}$$

No matter what the constant would be, it would not change the final outcome since the effect of the constant to each compared path is the same. As applied to Bayes' rule, now

$$\tau(w_1^n) = \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n). \tag{4}$$

As Jurafsky and Martin (2006) states Markov assumed that based on equation (4)

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \wedge P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}). \tag{5}$$

Finally as Huang et al. (2009) describes, the Markov assumptions can be modeled into previously simplified Bayes' theorem. This means that the most probable sequence $\tau(w_1^n)$ can be counted when

$$\tau(w_1^n) = \arg\max_{t_1^n} \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i). \tag{6}$$

More practically this means that a probability for a tag depends on the current word and on the previous tags and tag transition probabilities. On the beginning transitions are hidden, but this changes as the counting proceeds. (Singh et al., 2013)

4.5 Conclusion

In conclusion, HMM is a collection of algorithms operating with same principle to count transition probabilities from unobservable states (Kuo, 2006). HMM is widely used in POS tagging. Viterbi algorithm is one popular solution to implement the HMM. The time consumption of basic HMM is exponential, which makes it slow to execute. Still the accuracy of HMM in POS tagging context is high compared to other approaches. (Kuo, 2006)

5 EXPERIMENTAL STUDY

This chapter describes experimental study covering background information, the goal of the study, data gathering, preparation and experiment results. As well, this chapter contains comparison of the results between this study and previous similar studies. Suggestions for future work are more discussed in chapter 7.6.

5.1 Experiment Background

POS tagging in Internet slang is quite a new field of study compared to the fact that standard language is still sometimes tagged with corpuses developed on 1960's (Francis et al., 1979). Since the field is quite new studies in the field are not quite comprehensive yet. Still, slang tagging is helping tremendously in data mining from Twitter message stream. Real time data has already been proven to be beneficial in various situations such as detecting spread of epidemics. (Culotta, 2010)

Gimpel et al. (2011) are describing good ground level work for POS tagging from Twitter blog posts (tweets). The most important part is that the study is taking care of noisy text problem by clustering the words with same meaning into separate groups. Owoputi et al. (2013) have later updated the study with slightly more accurate algorithm. Among word clustering, both of the studies are containing large sets of manually annotated data, which is really important since there are no previous corpora covering different slang expressions. Because of the large background information those two studies are used as a background work for this study as well. The key difference in this study is to analyze POS tagging in slang with different algorithm than previous studies. As early mentioned in chapter 3.5 the success of a tagger depends mostly on the selected tagging algorithm. This difference can already been seen comparing results between Gimpel et al.'s (2011) and Owoputi et al.'s (2013) researches which both are using different tagging algorithms.

HMM has been great choice for algorithm in many different languages. In fact, Owoputi et al.'s (2013) study is based on MEHMM, which is one implementation of HMM algorithm. This study is focused to describe advantages and disadvantages of n-gram tagging algorithm in slang processing context. Derczynski et al.'s (2011) notes that for example bigram algorithm can be used to reduce uncertainty.

5.2 Goal of the Experiment

Previously slang POS tagging has been studied only briefly compared to Standard English. Only couple of tagging algorithms has been experimented with slang. Lack of the prior available data is partly a reason for that the studies have started only recently. Gimpel et al. (2011) are experimenting slang tagging with CRF and Owoputi et al. (2013) are improving the study by using MEHMM coded with Viterbi algorithm. The goal of this study is to describe a suitability of HMM based n-gram algorithms for English slang POS tagging and to detect factors affecting slang tagging accuracy.

Gimpel et al. (2011) and Owoputi et al. (2013) are suggesting different additional tools to gain tagging accuracy. Some of the tools are reimplemented with basic features and they are not trying to be as comprehensive as in earlier studies. The idea is to generally describe possible advantages of additional tools, but still the main focus is to compare suitability of tagging algorithm for Internet Slang.

5.3 Experiment Preparation

This chapter describes necessary preparations of our experiment covering data gathering, our experiment environment and configurations. As well, we present how the experiment is linked with previous studies.

5.3.1 Gathering the Test Data

The idea for this experiment is to annotate English slang words used in social media. In this research we use status updates, conversations and blog posts used in Twitter, since Twitter offers free and extensive API for their ongoing message data. Still, there is no actual data gathering done specifically for this study. For to able to compare the results to previous studies more accurately, we use previously gathered Twitter data from Owoputi et al. (2013). Another upside of the data selected is that the data is manually annotated. Manual annotation for the test data would have been necessary in any case for the accuracy evaluation.

Gimpel et al. (2011) have manually tokenized, annotated and filtered 1,827 tweets with 26,463 tokens. This data is divided in three parts, which are training data, development data and test data. The training data and development data are used together as slang corpus. The test data is used as-is. In practice, corpus and test data are both separate text files. The file contents are described in FIGURE 9 as they are separated in two files. Data gathering were done 27th of October 2013 by filtering English messages from United States' time zone. This data is called as OCT27, which contains 500 tweets (7,124 tokens). As well, we use new version of the data collected with similar method, which is represented in Owoputi et al.'s (2013) study. The only exception is that different tweets were collected every day between 1st of January 2011 and 30th of June 2012. This new data is called as DAILY547 containing 547 tweets (7,707). As an addition, we use Penn TreeBank and Brown Corpus as well, since they are widely used corpus. English language is the base for the slang used our data and it can help to train our tagging model more precisely than just plain slang corpus. After all, Owoputi et al.'s (2013) manually annotated data is relatively small compared to corpora sizes used for Standard English.

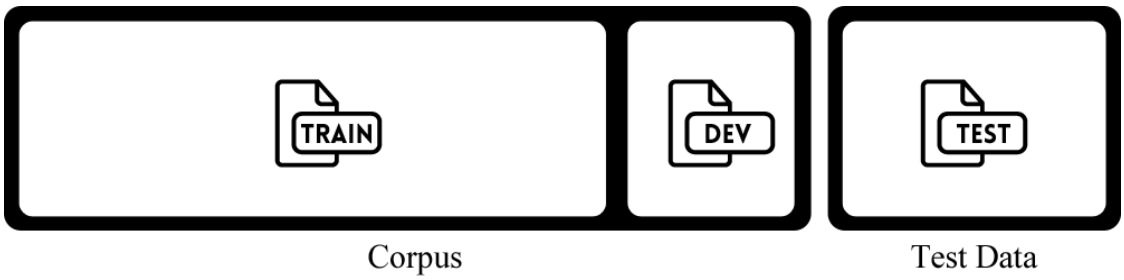


FIGURE 9: Manualy annotated tweets as they are separated in different files.

5.3.2 Experimental Environment

We selected open source tagger called Natural Language Toolkit (NLTK) for Python to be used for tagging process. NLTK offers access to different corpus and it contains different tagging algorithms (Bird et al. 2009). Tests are run on a MacBook laptop having 2,4 GHz Intel Core 2 Duo processor and 3 GB of 1067Mhz DDR3 memory. Test environment is run with built-in Python 2.7.6 on Snow Leopard operating system. Corpora used in test were downloaded via NLTK on 1st of February 2014.

Test environment was made specifically for this research using NLTK for Python. The actual n-gram based tagging process is executed with NLTK. Still, the domain specific execution order for given source data needed to be implemented. As well, Twitter-specific characteristics need special handling. An owerview of our code is shown in attachment 6 containing full program code of actual tagging process. The system diagram of our tagger is described in attachment 7. After clustering for the corpus and the test data is done, our tagger executes the taggin process in following order: Twitter-specific mark-up detector, number checker, bigram core tagger with clustered corpus, unigram tagger with clustered corpus, unigram tagger using Brown Corpus and Penn TreeBank,

name checker, emoticon detector and tokenization error detector. This execution order is repeated for each tweet. Our tagger execution order is described in FIGURE 10.

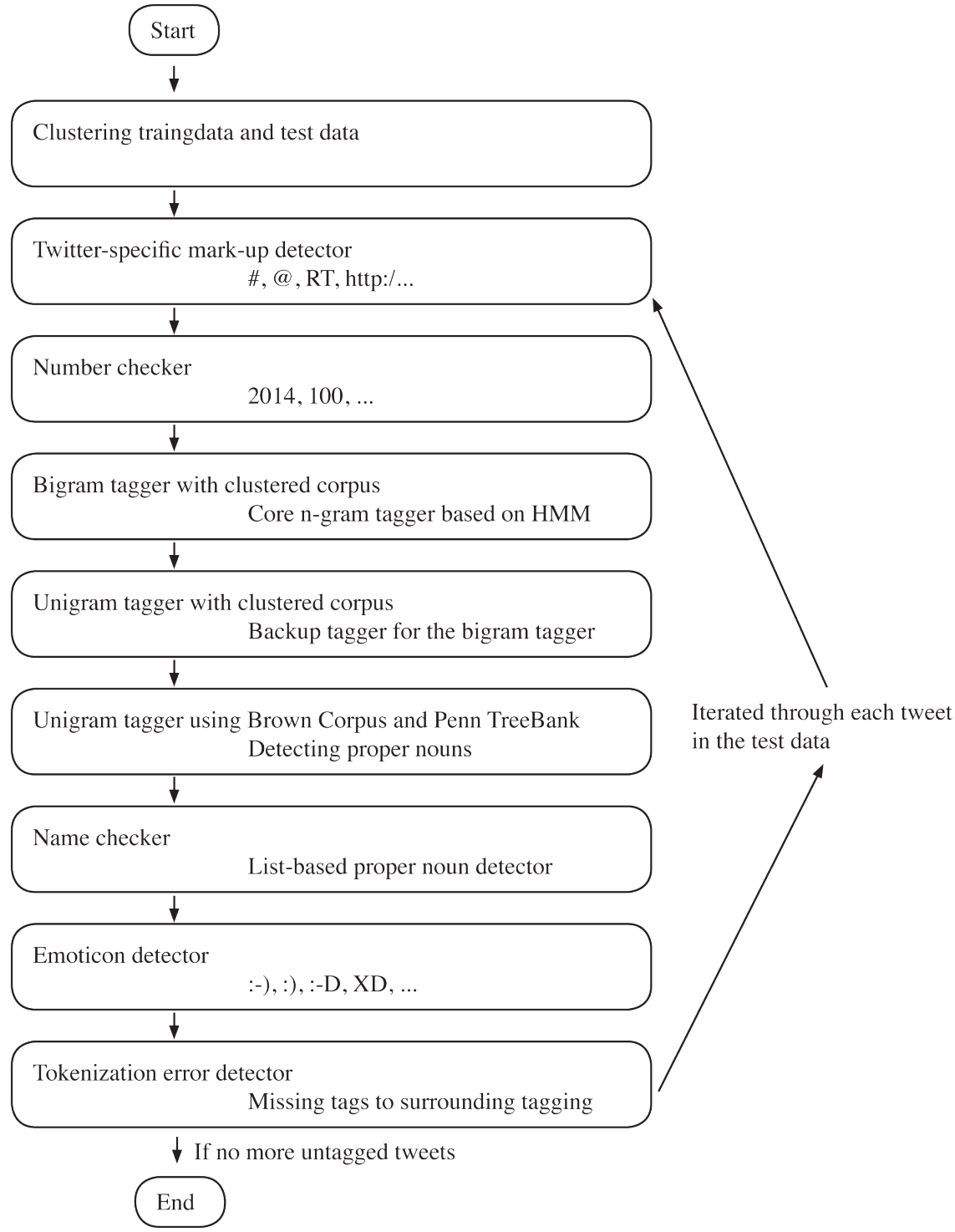


FIGURE 10: Tagger core execution order

The basic idea of our tagging implementation is to loop through each tweet in the test data, process the data and compare the processing results to training data by using NLTK n-gram tagging methods. Training data is a sepa-

rated part from manually tagged tweets. The other part is used as test data. The result processing includes string comparisons to different listings such as list of emoticons and list of names (proper nouns). The tweets in the test data and training data are pre-tokenized using whitespace characters as delimiters. Besides the n-gram tagging, the processing methods are considered as additional tools in this study since they are domain specific extension to tweet tagging and they are not directly related to actual tagging algorithm. Since the nature of informal text differs from standard text, the additional tools are used only to describe the complexity of the total tagging problem in informal context, and how it may be improved comparing to standard text tagging.

5.3.3 Arrangements

Even though Owoputi et al.'s (2013) study is based on Gimpel et al.'s (2011) research, they are using really different approaches with different additional tools and different tagger algorithms. For example for additional proper and common noun detection Gimpel et al. (2011) are using WSJ and Brown Corpus as-is, where Owoputi et al. (2013) are using lowercased Brown corpus. In this study we try to gain advantages of both approaches by using lowercased Brown corpus and Penn TreeBank as is. This is simply done by lowercasing the words in NLTK's library of Brown Corpus.

Penn TreeBank and Brown Corpus have different tag set from each other and from the tag set of our data. The tag set we are using is originally based on tag set defined by Gimpel et al. (2011) and it is more coarse-grained than tag sets in Penn TreeBank and Brown Corpus. Gimpel et al.'s (2011) tag set is described in attachment 3. Because of the tagging difference between slang and Standard English, we modify the Standard English corpus to use different tag set by replacing current tags with the tags suggested in Gimpel et al.'s (2011) study. Attachment 4 shows how Brown Corpus tags are replaced with Gimpel et al.'s (2011) tags. Attachment 5 shows the replacement for Penn TreeBank tags. The replacement is done to our tagger application instead of directly to corpus. This effects the running time a bit but replacement errors are avoided when replacing pairs are written in a table.

We constructed a basic tagger and different additional tools suggested in Gimpel et al. (2011) and Owoputi et al.'s (2013) studies. The core of our tagging application is n-gram tagger containing different n-gram sizes. All the higher n-gram sizes have backup taggers with one lower n-gram size, which again have a backup tagger. For instance trigram tagger has a bigram backup tagger, which again have a unigram backup tagger. The default POS tag is marked as "None" if the tag cannot be reasoned. The rest of the tools are additional domain related improvements to the basic core since in general, n-gram taggers can be really efficient by themselves while used with a standard language.

The first addition is word clustering. The cluster is the same that is used in Owoputi et al. (2013) research. The cluster contains 216,856 tokens where a token referres to word-to-token pair. Because of the informal nature of the slang, many written forms of slang words differ from standard language. Clustering

tool takes each text token and looks it up in the cluster data. When the word is found, the tool looks the cluster where the token is found and returns the mostly used text token from the cluster. This approach helps with spelling mistakes as well. In some cases this increases ambiguity, but it can help to normalize the written form of words.

An example of clustered text is described in TABLE 1. The first column "Cluster Identifier" describes the identifier for a cluster. The second column "Text Token" describes possibilities of written forms. The third column "Appearance Amount" describes the number of occurrences in different tweets. In this case if we look up word "meny" from the cluster, in return we get word "many", since it occurs more often than the misspelled comparison. After this, the new word "many" is more likely to be found from our corpus with a correct tag.

TABLE 1: Example of a word cluster with appearance amounts

Cluster Identifier	Text Token	Appearance Amount
111111111110	somany	57
111111111110	mant	93
111111111110	meny	102
111111111110	manyy	206
111111111110	many	332160
111111111110	mny	365

The second additional tool is one that detects names and numbers. Plain numbers are easy to detect by using just built-in functions of the Python programming language. For the name detection, Gimpel et al. (2011) have constructed a list of names commonly used in conversations, because they noticed that their tagger accuracy initially fell on proper noun detection. This list is combined with a list of names offered in NLTK toolkit. Name casing is lowered to avoid casing mismatch. These names are compared with lowercased text tokens.

The third additional tool detects a specific mark-up for Twitter. This includes emoticons, URLs, retweet markings (RT), hashtags (#) and links to other Twitter accounts (@). This tool is using naïve method by only looking the first character from a text. If the first character is Twitter-specific special marking, the given word is tagged as RT.

The fourth additional tool contains Brown Corpus and Penn TreeBank based n-gram taggers with different n-gram sizes. The idea for this addition is to detect possible Standard English words, which are not included in the training data. As well, since the taggers and tagging algorithms are performing differently with each other while using Standard English, the performance of these taggers can be compared to previous studies.

All the Twitter messages are annotated separately. There are three different token sets to be processed for each message. The first set is one message without any modification. The second set contains same tokens in lower case. This is due to Gimpel et al. (2011) research finding that some of the proper

nouns are easier to find in all lower case mode. The third set is the lower cased message corrected with most likely written forms from Owoputi et al. (2013) token cluster. The experiment is done by varying the configuration of the tools and proceeding with the best results from earlier executions.

5.3.4 Configuration

The configuration is settled by varying the execution order of the core tagger and additional tools proceeding with the best results from earlier executions. As for the result for each run, the analysis of each tag is printed out. The format of the development analysis is a copy of Gimpel et al.'s (2011) accuracy table, which shows recall and most common confusion for each tag. Our extension to table shows correct and missed amounts for each token and displays the amplitude of the most common error. With these extensions it is easier to monitor which kind of configurations should be made to the tagger. Also it is easier to note possible programming mistakes for our additional tool when the error is somewhere else than in the syntax.

Our initial executions show some guidelines for our final execution order. For example tagger tends to confuse Twitter-specific mark-up to punctuation marks. This means that the third additional tool mentioned in chapter 5.3.2 detecting Twitter-specific marks, should be used before the core tagger.

We tried to gain tagging accuracy by trying to detect Standard English trigram and bigram connections before our core tagger execution. This caused the tagging accuracy to fall couple of percentage points. This indicates that Internet slang has its own rules of sentence structure and it should primary be annotated with slang corpus.

As mentioned in earlier studies (Owoputi et al., 2013), as well our tagger lacks of accuracy in detecting nouns. To ease these problems, it is beneficial to add Standard English unigram tagger after our core slang tagger. With our selected tools the most beneficial configuration to this is to use unigram tagger using Brown Corpus with extension of unigram tagger using Penn TreeBank. The accuracy is not gaining at all when using higher n-grams in this case. A simple explanation to this is that there are no actual language related issues to be fixed and as mentioned earlier written format between Standard English and slang is notably different. Missing nouns are only looked-up from Standard English corpus, which can be detected using unary rules. As well, these unigram taggers are getting their highest accuracy when the given text is lowercased. Still after this, there are nouns without any POS tag. This is why it is most beneficial to set the name detection addition to the very end of the execution process. As an addition our tagger is configured to avoid annotation problems caused by tokenization. These are occurring most notably in Twitter-specific mark-up and in Internet addresses. The correction is made only in the case that there is no other detectable tag to the given token.

In conclusion, our tagger has similar configurations than Owoputi et al., (2013). The main differece between the taggers is the algorithm used for HMM based processing. As well, there are method level differences in the implemen-

tation when comparing additional tools. Still the operating principles between the taggers are the same. The most important factor for our experiment to be comparable is still to use tha same data which was used in previous experiments.

5.4 Experiment Results

This chapter describes all the different experiments made with our tagger described earlier in chapter 5.3.3. In general, we tested the effect of different additional tools n-gram and corpus sizes and how they affect to the tagger accuracy.

NLP system evaluation methods are introduced in chapter 2.2. Even though NLP systems are generally evaluated and compared with combinational evaluation value F-measure, still only precision is used to describe our experiment results.

5.4.1 Ablation Experiment

TABLE 2 contains our final tagger accuracies with different configurations or as Owoputi et al. (2013) calls it "ablation experiment". As well, results from earlier-studies are listed to the same table as a comparison. The tests are made with OCT27 data and DAILY547 data, which are used in previous studies. Our final tagger reaches a tolerable accuracy level slightly over 87% compare to other taggers.

The main idea of the accuracy table is to compare the algorithm performance. Owoputi et al. (2013) are using MEHMM and Gimpel at al. (2011) are using CRF as the tagging algorithm. The idea in these previous experiments is to gain the best possible accuracy for a POS tagger. Since the approach in the tagger implementation is different it has an effect to results, which can be seen from the TABLE 2. Previous taggers are notably more accurate when executed with all tagger features. With OCT27 test data Owoputi et al. (2013) are gaining near 92% accuracy and Gimpel at al. (2011) over 89% accuracy rate. Our additional methods have much higher impact to the final tagger than additional methods used in the previous studies. Still, the additional methods are rule-based conditions, since they are not included in the statistical algorithm model. This means that the most comparable lines in the TABLE 2 are our tagger only clusters (and transitions), Owoputi et al. (2013) tagger only clusters (and transitions) and Gimpel et al. (2011) base tagger. Gimpel et al.'s (2011) base tagger accuracy in DAILY547 column is in parenthesis, since the result describes tagger accuracy while executed with development data. DAILY547 data were not used with Gimpel et al. (2011) base tagger.

Even though clustering is additional tool its idea is only to ease the manual annotation process, since unlabeled conversational data is easy to acquire from the web. From the comparable numbers mentioned, we can conclude that

MEHMM is way better algorithm than N-grams tested in our experiment, since the accuracy difference is over 8% points. Our final tagger contains bigram tagger since it has slightly better accuracy than any other n-gram tagger. Our tagger is still really close to Gimpel et al. (2011) tagger using CRF tagging algorithm. The difference is roughly around 1% points while they both reach accuracy rate of 82% (± 1%). The difference is so small that we can consider CRF and N-gram approaches to be equally accurate.

There are a lot of differences between our tagger and taggers used in previous experiments. One notable difference is in word tokenization. Even though it was assumed that the tokenization was already done by other external application, still Owoputi et al.'s (2013) tagger is splitting some tokens into multiple parts. When we have a token, which cannot be tagged with our tagger we get an error. Owoputi et al. (2013) in the other hand is splitting the token into multiple parts which all have a possibility to be successfully tagged. This thing was noticed while counting the inter-annotator agreement with Owoput et al. (2013).

In general, since the inter-annotator agreement is only 88%, it can be concluded that the MEHMM and N-grams are succeeding in different areas. Still MEHMM is more suitable algorithm to be used with slang since the accuracy is way better in Owoputi et al. (2013) tagger even without taggers features, which more clearly shows the effect of the algorithm. Still, a notable downside using n-gram tagging algorithms is that they are not performing well with unclustered words. As can be seen from the TABLE 2, the accuracy with unclustered data varies between 66% and 77% depending on whether the additional features are used or not.

TABLE 2: Tagging accuracies based on Gimpel et al.'s (2013) ablation experiment and comparison to earlier studies

Tagger	**OCT27**	**DAILY547**
Final tagger	87,25%	87,82%
with clusters; without tagdicts, namelists	81,43%	81,98%
without clusters; with tagdicts, namelists	76,96%	77,56%
only clusters (and transitions)	81,39%	81,89%
without clusters, tagdicts, namelists	66,21%	67,50%
Inter-annotator agreement (Owoput et al., 2013)	88,05%	88,79%
Owoput et al. (2013) all features	91,60%	92,80%
with clusters; without tagdicts, namelists	91,15%	92,38%
without clusters; with tagdicts, namelists	89,81%	90,81%
only clusters (and transitions)	89,50%	90,54%
without clusters, tagdicts, namelists	86,86%	88,30%
Gimpel et al. (2011) all features	89,37%	89,17%
Gimpel et al. (2011) base	83,38%	(82,70%)

Even though our tagger cannot compete in tagging speed, the issue is not in the algorithm efficiency, since MEHMM and n-grams are both based on the same principle of HMM. Owoputi et al. (2013) describe how they use modified Viterbi algorithm for predicting tags. They also describe that the time cost of the basic Viterbi prediction is counted as $O(|x|K^2)$, where K is total number of tags and $|x|$ describes given words. This is aligned with the HMM time consump-

tion described in chapter 4.2. The modification of Viterbi algorithm is based on MEHMM greedy decoding, which makes the algorithm 3 times faster than normal Viterbi decoding. Still, even the slower basic Viterbi is decoded faster than our tagger execution. Since our most efficient tagger is bigram tagger, there is only one previous state transition. This means that in our case the time efficiency is constant since K is always 2 when modeled to Owoputi et al. (2013) time consumption formula. From this, we can conclude that the time consumption of n-gram algorithms could be tolerable in different environment and the open source NLTK tagger on the other hand lacks of speed. Still the time consumption of our tagger is quite close to the Gimpel et al. (2011) CRF tagger comparing to speed reported in Owoputi et al. (2013) study.

MEHMM and N-grams used in this experiment are both based on HMM. CRF on the other hand have completely different working principles. After all, since this study is not focused on developing additional tools to gain the best possible tagger performance, the accuracy could be much higher if the features from previous studies would be adopted in our tagger as-is. Even though the tagging accuracies varied between this study and previous experiments, there is still another efficiency issue, which might be important for a final NLP system. The execution time of our tagger is around 2 minutes, which means that one tweet can be tagged in 4 seconds. Contrary Owoputi et al.'s (2013) tagger annotates all 500 tweets in 4.4 seconds. Our tagger lacks in execution speed even more while using Brown Corpus or Penn TreeBank. The tagging with these features takes over two hours.

5.4.2 Common Tagging Errors

Tag-specific accuracies are described in TABLE 3. The table is based on Gimpel et al. (2011) tag recall table on full model by using 500 tweet test data. The first column called "Tag" describes the tags used in our tagging experiment. Tags and word classes are described in attachment 3. The second column "Accuracy" describes the tag specific recall rate in our experiment. The third column "Gimpel et al. (2011)" describes recall rates of same tags in previous study. Fourth column "Tokens" represents the number of tokens and fifth column "Missed" describes how many of the tokes were tagged incorrectly. The sixth column "Confusion" describes the most common confusion and how likely it appeared in our experiment.

In general, based on second and third column on TABLE 3, our tagger fails in similar places than Gimpel et al. (2011) CRF tagger. The absolute most likely confusions are proper (^) and common (N) nouns as Gimpel et al. (2011) earlier stated. These two are mostly mixed with each other and with verbs (V). Noun related errors covers over 40% of all tagging failures. There is similar confusion with pronouns (O) and determiners (D), but the absolute error amount is relatively smaller. As well, possessive + nominal (S) and possessive + proper nouns (X) are not tagged correctly even once. Still the impact of those errors are less than 0,2% points, since there are only few words tagged with the word classes in our test data, which leaves the error quite irrelevant.

There are no missing tags in the last column indicating the most common confusion. This means that the word ambiguity is the greatest problem of our tagger, since most of the words can be found from the training data or from the clusters used in the experiment. Even when our tagger is executed without any features the distribution of tag specific errors is similar compared to tagger using all possible features.

Our additional tools seem to cover missing tags quite well. After all, the total effect for additional tools is only around 6% points, since the most likely problem of our tagger without additional tools is missing tags. From this we can conclude that the actual problem is in the n-gram algorithm since it does not cope well with word ambiguity compared to MEHMM.

TABLE 3: Tag-specific accuracies and errors based on Gimpel et al. (2011) accuracy table

Tag	Accuracy	Gimpel et al. (2011)	Tokens	Missed	Confusion
@	100 %	99 %	330	0	
#	100 %	89 %	78	0	
U	99 %	97 %	117	1	,(100%)
,	98 %	98 %	880	18	~(78%)
&	95 %	98 %	127	6	V(67%)
P	93 %	95 %	616	44	R(39%)
O	92 %	97 %	505	38	D(58%)
V	92 %	91 %	1053	80	N(40%)
D	92 %	95 %	449	38	O(58%)
L	91 %	93 %	129	11	D(45%)
$	91 %	89 %	85	8	#(38%)
~	89 %	91 %	212	23	,(83%)
!	83 %	82 %	186	31	O(16%)
E	83 %	88 %	63	11	,(27%)
R	81 %	83 %	339	64	P(33%)
N	81 %	85 %	981	187	^(43%)
A	76 %	79 %	367	88	V(25%)
T	72 %	72 %	36	10	P(70%)
^	66 %	71 %	495	169	N(41%)
Z	23 %	45 %	22	17	^(41%)
G	20 %	26 %	70	56	#(21%)
X	0 %	0 %	6	6	R(50%)
S	0 %	0 %	6	6	A(83%)

Based on tag specific accuracies, we counted the sentence retrieval accuracy, which in our tagger is around 24%. The accuracy is counted by isolating different tweets from each other and multiplying all the tags within a sentence. The calculation is done with manually tagged test data (correct tags) and with tag specific accuracies counted based on our tagger annotating results. With this counting method, the sentence retrieval accuracy is weighted with token number of occurrences, which gives more comparable sentence retrieval amount. This is similar compared to POS tagger proposed in Derczynski et al.'s (2013) study.

5.4.3 The Impact of the Corpus on Accuracy

As stated earlier in chapter 3.3.1 with Standard English, the corpus size affects the efficiency of n-gram tagger. FIGURE 11 shows n-gram tagger accuracy dependency on the corpus size as it behaves varying the OCT27 data size. This experiment includes six different size training data, which in this case is referred as corpus. The largest corpus contains 20,770 tokens. Other corpus sizes are multiples from one fifth of the original OCT27 data size. The tagger is also executed without any corpus data. FIGURE 11 describes that corpus size effects significantly on the tagger accuracy.

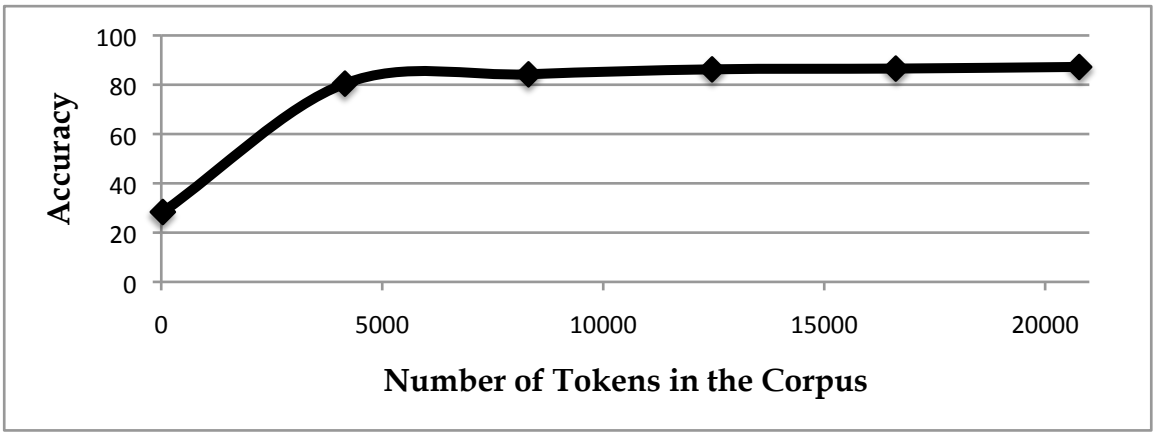


FIGURE 11: Accuracy depending on the corpus size

The proportional improvement after 5,000 tokens seems to decrease and after 15,000 tokens it seems to stop since there is no increase in the accuracy. In fact, the next test, after 15,000 tokens corpus, gives slightly lower accuracy rate. This accuracy decrease is caused by noisy training data as Liu (2003) described it. As well, when the tagger is executed without any corpus, tagger still includes additional tools such as twitter specific markup detector and proper noun detector. These features by them selves give 30% tagging accuracy rate. Even though nouns cover about 20% of our total data, the rest of the accuracy percentage points indicates that Twitter messages contains a lot of domain-specific markings. That is why rule-based extensions might be beneficial improvements for POS tagger.

5.4.4 N-Gram Effect

N-gram size has effect on tagging accuracy while tagging Standard English (de Holanda Maia & Xexéo, 2011). This effect of the n-gram size can be seen in attachment 2. Still, our experiment contains n-gram effect of tagging accuracy in slang context. Additional tools are used as features on the tests. According to our initial test they are not affecting the accuracy differences with different n-gram sizes. FIGURE 12 describes n-gram tagger accuracy depending on the n-gram size. All the n-gram sizes on FIGURE 12 experiment contain piled backup

taggers meaning that each size has lower size backup taggers. This way the n-gram taggers benefit of adding higher n-gram taggers. The experiment contains different configurations by varying clustering target. The different configurations are as the following: clustered training data with clustered test data (CC), unclustered training data with clustered test data (UC), unclustered training data with unclustered test data (UU), and clustered training data with unclustered test data (CU). According to the experiment n-gram size does not have significant effect on the total accuracy. There is only a slight improvement noticeable between bigram tagger and other taggers. The most noticeable factor in FIGURE 12 is still that how clustering is effecting on the final outcome. As can be seen from the FIGURE 12, the most promising method is to use clustering with the given slang text. As well, clustering the traing data is beneficial.

Clustering does not guarantee high accuracy rate. This can be seen in FIGURE 12. Our total unclustered test UU performs still better than the partly clustered test CU, which has clustered training data but unclustered test data. This differs from the full clustered test over 20% points. Since unclustered tagger performs better than corpus-clustered version CU, the clustering causes some mismatches, which could be result of word ambiguity. There are some loose connections between slang words since the n-gram tagger performs better with size two than with size one. Still, as can be seen from the FIGURE 12, the tagging is mostly based on unigram rules.

As an extension, we tested slang tagging accuracy with different size n-gram algorithms with Standard English using Brown Corpus. The accuracy stayed constantly in 56%. This case is not seen in the FIGURE 12. This only indicates that there are overlapping parts between English slang and Standard English, but they have a lot of differences as well. This already describes the importance of slang tagging studies and the fact they should be separated from the standard language.

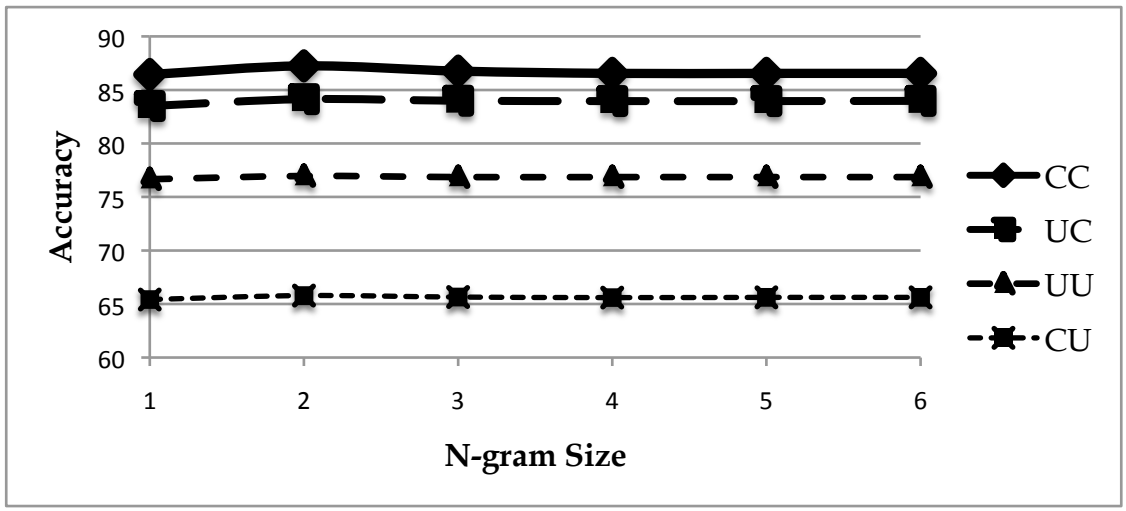


FIGURE 12: N-gram tagger accuracy depending on the cumulated n-gram size. CC: clustered training data with clustered test data; UC: unclustered training data with clustered test data; UU: unclustered training data with unclustered test data; CU: clustered training data with unclustered test data

Even though increasing the n-gram size is not increasing the accuracy, there still can be higher hidden Markov property connections between tag transitions. For to test these connections FIGURE 13 describes n-gram tagger accuracy without any backup taggers or any features. Experiment includes tests with clustered training and clustered test data; as well it includes tests with unclustered training and unclustered test data.

As an extension, Standard English tagger using Brown Corpus is also added to chart to describe how NLTK performs with standard language. FIGURE 13 shows that most of the word-to-tag connections are unigram connections, since unigram accuracy is between 66% and 81% depending whether the corpus and test data are clustered or not. Compared to this, bigram connections are already significantly dropping. This indicates that slang words have only loose connections with each other. With clustered corpus and clustered test data, the accuracy on bigram tagger drops to slightly over 14%. After this, higher n-gram taggers seem to stay a bit above 6% in accuracy. This indicates that the longer path connections exist, but there are only few distinguishable longer paths.

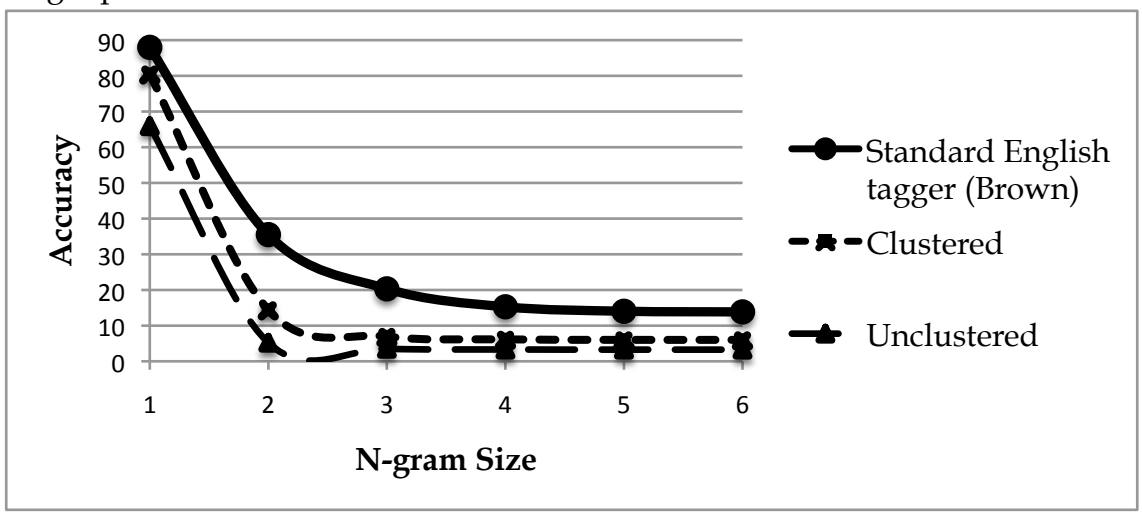


FIGURE 13: N-gram tagger accuracy depending on the plain n-gram size with Standard English tagger using Brown Corpus, with clustered training and test data, and with unclustered training and test data

Even though higher size n-gram taggers do not have high accuracy rates, still the main problem is missing tags. This can be seen from the tagging results. The tags, which are in this case found, are tagged really accurately. The accuracy of the found tags is over 98% with unclustered data, and over 90% with clustered data. Even though unclustered tagger finds tags more accurately, still the absolute amount of correct annotations made is much lower. With this information, it is possible to place extra n-gram taggers first in the tagging execution order. This increases total results about 1% points. Even though this increases tagger performance it does not change the fact that MEHMM is performing better in tagging task than n-gram taggers.

6 ANALYSIS OF EXPERIMANTAL STUDY

This chapter contains analysis of our experiment results in general and related to each experiment separately. Experiments cover five different areas. Tests cover ablation experiment, testing of tag-specific accuracies, and corpus size effect to tagging accuracy, and effect of n-gram size. Findings from the previous studies are also compared to our study results in this chapter.

6.1 General Performance

The idea of the empirical study is to test n-gram algorithms and their suitability for slang in POS tagging context. N-gram taggers are not performing as well as MEHMM algorithm in Owoputi et al. (2013) experiment. Even though the algorithms are both based on HMM, our n-gram tagger lacks both in speed and accuracy in all different ablation experiment tests. Since the features are ruled out in different test, it can be concluded that MEHMM is a better algorithm for slang tagging than any of the n-gram algorithms. Even though MEHMM on Owoputi et al. (2013) tagger works faster than our different n-gram algorithms, the slow execution speed is partly caused by the NLTK environment used in our experiment. Owoputi et al. (2013) and Gimpel et al. (2011) have constructed their whole tagging program from a scratch modeling Viterbi implementation of HMM. In our case, NLTK is a wide collection of different methods, which are not optimized for any task. This difference is one reason for the difference in execution speed. This difference can be seen easily from tagger execution. Owoputi et al.'s (2013) tagger does the total annotation process faster than NLTK loads its Python libraries. Owoputi et al.'s (2013) tagger was tested with Intel Core i5 2.4 GHz laptop.

6.2 Ablation Experiment

An interesting part of the ablation experiment is inter-annotator agreement between our n-gram tagger and Owoputi et al. (2013) MEHMM tagger. The agreement is only 88%. This is a surprisingly low number since both algorithms are based on the same principle even though the implementation is a bit different. This indicates that many of the errors taggers made differ from each other. As well, tagger accuracies differed nearly 20 % points when the additional features were not used. Since the tagging results are so different, this means that MEHMM and n-gram should be considered and tested as different algorithms in the future as well. Nevertheless, n-gram approach is more naïve approach to implement HMM than MEHMM is.

CRF tagger used in Gimpel et al. (2011) study performs quite similarly as n-grams tested in our experiment. There are no significant improvement between unigram and higher sized n-gram taggers. On the other hand with standard language increasing n-gram size is increasing the accuracy rate to certain point. This accuracy change can be seen from attachment 2. Since there are no important improvements while n-grams are used with slang, most of the tags can be reasoned with a simple unigram tagger. This decreases the relevance of a tagging algorithm if the algorithm is reasoning the tags only with unary connections.

6.3 Tag-Specific Accuracies

There are a lot of differences in tag-specific accuracies within our study. In general, our n-gram tagger performs similarly as CRF tagger developed by Gimpel et al. (2011). The total accuracies are near the same leve. This implies to tag-specific accuracies as well. Still, there are some differences between annotation results.

While n-gram tagger is executed with clusters there is no missing tag problem, most of the words are getting at least some kind of tagging. This means that our n-gram algorithms are able to reason some kind of answer for tagging question in most cases. The problems are mainly in incorrect tagging. This refers more to an algorithm related problem since most of the time the errors are mistagged words rather than untagged words. Tag specific problems are similar whether the POS tagger is executed with word clusters or without. Only the amount of found tags is different. This means that most of the tagging problems are related to word ambiguity, which in slang context is mostly caused by noisy data. Partly this can be seen when tagger is executed without additional tools and without clusters. Even though Twitter-specific markings cause some missing tags, there are still many other missing tags on regular word classes as well. POS tagger using slang-based training data is still performing lot better than tagger using Standard English. The accuracy difference is over 30% points.

As described in Gimpel et al. (2011), nouns have a lot of annotating problems. This is an issue for our tagger as well even after using additional tools such as word clustering and name lists. Because of this and since tagging problems are mostly related to word ambiguity, it is quite clear that even though algorithm selection in slang tagging is important, slang tagger cannot cover all the tags only with algorithm-based solutions. Higher accuracy rates are really hard to achieve without extending slang taggers with rule-based tools. Owoputi et al. (2013) have developed these tools, which have pre- and post-processing characteristics. Still, even though word clusters contain a lot of information how different slang words are written in general and name lists are quite extensive, some of the additional tools are mostly domain related, since Twitter-specific markup cover a huge part of the total data. Currently all compared taggers have based their algorithm related decision-making process on corpus. Still, since there are common confusions between certain word classes, a tool to guide decision-making process with these commonly confused tags could increase the total accuracy of the POS tagging system.

Different algorithms are able to annotate different word classes with different accuracy. This difference might be beneficial, since it creates an opportunity to build a joint tagger containing different algorithms, where they could be used to support the core processor in decision-making.

6.4 Effect of a Corpus

The corpus size has the most notable effect on the tagging results. There is no significant improvement on the results after the corpus size is over 15,000 tokens. The most noticeable effect is between none and 5000 token training data size. One major problem is still that the training data contains a lot of noisy data. Owoputi et al. (2013) describes their cluster data gathering method contains occurrence frequency filtering. The text tokens appearing 40 times or more are taken into account for clustering application. Similar filtering could improve training data collection as well. Corpus of course should contain rare word-to-tag connections as well to cover more different kind of data, but this filtering method could be used as a weighted guide in data gathering process. This different data gathering process might ease the noise related problem in training data and cover different kind of use-cases more thoroughly.

Standard English corpus is not useful in annotating slang. It can cover only a few of the tag transition connections, which slang corpus is already covering. As couple of Standard English corpora were used in Owoputi et al. (2013) research to find missing nouns, our experiment shows that even though some nouns can be found with this method, it does not notably increase the total system accuracy. Name listings instead are useful addition to total system, but they make a lot of annotating mistakes as well. Slang used in Twitter should be considered as different language from Standard English when considering automatic text annotation.

6.5 N-gram Size

Our experiment shows n-gram size does not have much effect on the final tagger accuracy, no matter whether the training data or test data is clustered or not. This is different from Standard English since higher n-gram sizes are improving the tagging results (de Holanda Maia et al., 2011). With slang, some of the tag-to-tag connections are still improving the results since bigram tagger is performing better than naïve unigram tagger on cumulative tagger experiment. Our non-cumulative experiment on the other hand shows that even though cumulative n-gram tagger accuracy does not increase when the tagger increases, still there are some connections between tags. This means that statistical methods are considerable choice for being a slang tagger. Even after this, since most of the tagging decisions can be reasoned by using unary rules this means that the tagging decisions, which cannot be reasoned are mostly cause of noisy data. As well, this statement is backed up by the fact that the same tagger performs similarly with Standard English on varying n-gram size, but the accuracy percentages are only higher than with slang. This means that slang has a looser word ordering. Other factor could be the typical structure of Twitter messages. Gimpel et al. (2011) suggests that some word classes are more likely to appear in certain places than with certain tags. It implies that tag-to-tag connections might be longer than just connections between adjacent tags. As well, it partly explains why n-gram taggers lack on performance with higher n-gram sizes. This finding could be used to improve tagger decision-making. On the other hand, one reason for small amount of tag-to-tag connections could be noisy data. One solution to this could be a preprocessing tool to detect common and proper nouns before other tagging decision are made.

6.6 Comparison to Previous Studies

Only few existing studies are containing POS tagging experiments related to slang. Most of the studies have been published during resent years since the slang data have become more available than before. The results of our study were mostly aligned with previous studies. Still this study extends the previous findings to slang annotation research as well.

As mentioned earlier, HMM does not perform well with small amount of training data (Ekbal et al., 2007). This is true according to our experiment as well. Our experiment defines the threshold of the training data amount in slang context, which is somewhere between 5,000 and 15,000 manually annotated tokens at least considering Twitter messages.

Kouloumpis et al. (2011) presented Twitter sentiment analysis containing POS tagging. The tagger they were using was an n-gram tagger like the one in our experiment. As their tagger accuracy remains around 60%, our tagger out performs this over 20% points. Mostly the reason to the difference is the clustering method suggested in Owoputi et al.'s (2013) research. Without clustering

and other additional tools, our tagger accuracy performs really similarly as Kouloumpis et al.'s (2011) POS tagger. This describes the importance of additional tools for slang POS tagger and clarifies how important algorithm selection is even in slang POS tagging. Contrary to n-gram tagger Owoputi et al.'s (2013) MEHMM reached over 86% accuracy without clustering or other additional tools.

Our n-gram algorithm lacks in accuracy compared to MEHMM used in Owoputi et al.'s (2013). This is aligned with Singh et al. (2013) statement that generally HMM outperforms n-gram solutions. The reason for this is in the implementation difference of these two HMM solutions. While n-gram is only looking at nearby tags, MEHMM is considering longer path. This is beneficial since as Gimpel et al. (2011) described the tag relations might be longer than only near by tags. As well, one reason for this is frequency of ambiguous words.

7 CONCLUSIONS

This chapter is covering connections to previous experiments and the conclusions of empirical study are described. This chapter contains conclusions of the key issues, which are POS tagging, slang related methods, meaning of the experiment results, and a comparison to previous similar experiments.

7.1 Objectives and Key Findings

The goal of this study is to describe the most suitable algorithm for POS tagger in slang context. The actual research problem in this study is to find out how n-gram algorithms are performing in automatic slang annotation compared to previously experimented algorithms such as MEHMM and CRF.

This study finds that MEHMM is the most suitable algorithm for text annotation in slang context. The performance is as good as using CRF algorithm. All of the algorithms are working with different principles and giving different kind of tagging results even though final accuracies between CRF and n-grams are close to the same.

7.2 POS Tagging in General

Automatic text annotation by POS tagging is a way for machines to understand a natural language. POS tagging is part of NLP scope, which again is a part of AI. POS tagging is a low level technology in many other NLP solutions. For example it is important part in machine translation applications (e.g. Momtazi et al., 2010).

Preferred POS tagging algorithm depends on the natural language in given text. This can be explained by differences in natural language structures. POS tagging methods are divided in statistical and rule-based approaches (Lv, 2010). Both of the approaches have different algorithms to process natural lan-

guage. Statistical approaches can be most of the time used in many different cases and they have good performance in European languages. Most common POS tagging algorithms are HMM, CRF and SVM. For example, HMM have reported to have good accuracy in many different languages especially European languages such as English. In fact with English language, HMM is performing better than any other of the approaches. As well, HMM can be implemented in different ways. N-grams, MEHMM and Viterbi algorithm are all HMM related algorithm, but they have different working principles. Still, sometimes rule-based approaches are giving better accuracies or they can be used as an extension for statistical approaches. For example Portuguese (de Holanda Maia & Xexéo, 2011) and South Asian language Telegu (Hasan, 2007) is succeeding much better in rule-based approaches.

POS tagging is most likely to succeed when the style of tagged text is corresponding the corpus in use. Efficiency and accuracy of the POS tagging process for given text are depending on a tagging algorithm and the size of the corpus. Success of POS tagger depends on language suitability of selected tagging algorithm and on prior language analysis. In rule-based approaches the prior language analysis is included in the algorithm, but in statistical approaches corpus size is directly proportional to the tagging accuracy.

7.3 Slang Tagging

Slang tagging differs from tagging a standard language. Slang has more ambiguity problems than standard language and written form of a single word can randomly vary since misspellings are common or intentional (Ritter et al., 2011). In annotation perspective this ambiguity and misspelling problem is considered as a feature of a different language than a problem of standard language. Slang taggers are clustering the words with a same meaning into a single group to ease the tagging process, which in other case might be much slower process.

In conclusion, POS tagging for slang is a challenging issue and it should be studied more. Standard language POS tagging is not completely accurate process yet either (Manning, 2011). Even though slang tagging has its challenges; English written slang can already be annotated with higher accuracy than some standard languages. For instance English slang can be automatically annotated with 93% (Owoputi et al., 2013) accuracy rate while Hindi is only reaching a bit over 70% accuracy rate (Hasan, 2007).

Most of the advances in slang annotation are because of the domain-specific methods developed to identify the word classes. The ambiguity amount of the text tokens is too high for statistical methods to reach accuracy levels near 100% only by themselves. Still, word clustering is a promising approach to ease ambiguity problem and speed up the execution process. (Owoputi et al., 2013) In addition clustering could be used in different purposes as well. One example of this could be spelling checkers in writing tools.

POS tagging is still usually a smaller part for larger NLP system (Bird et al., 2009). Since there are many other system components with different execu-

tion speeds, the total execution time of the system might rise relatively high (Chien et al., 1993). In some cases speed is preferred over accuracy, since time-consuming algorithms might be crucial to the final system, if the accuracy is not notably affected (Owoputi et al., 2013).

7.4 Influence of the Experiment Results

The findings of this study can be used as a guide in POS tagging experiments in the future, and as well to give guidelines how slang POS taggers should be constructed. The most important finding is that slang words do not have as strong tag-to-tag connection as Standard English has. More closely, some connections might be even to further connections than only nearby tokens. This means that n-gram algorithms are not the best choice for a POS tagger processing slang. As well, this finding creates a possibility that some other languages, which are not performing very well with HMM annotation, there still might be connections to tags located further in the sentence.

This study points out the weaknesses of slang annotation tools, which are mostly related to noise and detecting nouns. In order to mitigate the noise problem, this study points out the benefit of word clustering. As well, this study shows the importance of rule-based additional tools in slang tagging.

Improvement in slang POS tagging might become important in automatic translation tools where POS tagging is used as lower level technology. As well, this study represents tools for handling text-based noise. These findings might be beneficial for some other NLP tools, which are facing noise related problems.

There are other fields as well, where POS tagger is used as an important part of larger system. One of these different systems is sentiment analyzer. It can be used as a prediction tool for a stock market trends (Bollen, Mao & Zeng, 2012). If the POS tagging fails to understand the key meaning of a sentence, the sentiment analysis might give totally opposite answer than it should.

7.5 Limitations of the Study

Mostly the additional tools used in tagging process are domain-specific and they most likely are not transferrable to other contexts. Word clustering on the other hand can be directly used in other environments. Mostly additional tools are originally based on observations on common language usage in twitter. As well, Only English Language and English Slang are used in our study. Additional tools might not be transferrable to other languages or slangs.

The actual tagging algorithm examined in the study is used with an open-source platform NLTK, which is a toolkit for general NLP tasks. NLTK might cause some efficiency problems considering time efficiency.

7.6 Future work

Tagging accuracy could be improved with an additional NLP tool, which would parse and detect the text style of the parsed text tokens. A reason for this is the fact that taggers are succeeding better when the corpus text styles are corresponding to the given text. Sometimes a single text includes different writing styles and each different style would have the best tagging match in different corpus. This kind of improvement would still need future studies since there are no similar tools yet. A challenge in this approach is the processing speed of the proposed section. POS tagging can already be relatively time-consuming part in NLP application. A text style detector might increase the time consumption even more.

Slang annotation is still quite a new field of study. Even though previous advances in automatic standard language annotation have been beneficial kick-start for slang annotation, there is still a lot to be discovered. For instance, slang tagging should be tested with various different tagging algorithms and they should be tested with different languages to gain the knowledge of best algorithms in each different use-cases. These kinds of studies would be one step towards simultaneous language interpreter application, which would understand slang and be able to operate in everyday life.

HMM can be implemented different ways. This study compares MEHMM and n-gram usage with English slang. Because of the different results, MEHMM and n-gram should be considered and tested as different algorithms in the future as well. In slang context MEHMM should be used as primary option since it seems to outperform CRF and n-gram algorithms. Still, MEHMM should be applied to slang annotation in the case where the base language has a good performance with HMM based tagging algorithms.

Some word classes are causing notably more ambiguity than others. Mostly this is related to nouns. Additional tools for noun detection could partly ease this problem. This could be beneficial for study of NER. NER may also have some methods already, which may improve noun detection.

One important study finding is that clustering significantly improves slang POS tagging. Our experiment points out that clustering can improve POS tagging accuracy while it is used in certain way. This indicates that closer studies for slang clustering in POS tagging context should be made related to data collection and usage.

REFERENCES

Agarwal, S., Godbole, S., Punjani, D. & Roy, S. (2007). How much noise is too much: A study in automatic text classification Data Mining, 2007. In ICDM 2007. Seventh IEEE International Conference on, (p. 3-12). Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4470224

Allen, J. (1995). Natural Language Understanding (2nd edition). Redwood City, CA, USA : Benjamin-Cummings Publishing Co. and Inc. ISBN 0-8053-0334-0.

Alpaydin, E. (2010). Introduction to Machine Learning (2nd edition). Cambridge, MA, USA: The MIT Press. ISBN 026201243X, 9780262012430.

Awasthi, P., Rao, D. & Ravindran, B. (2006). Part of speech tagging and chunking with hmm and crf. In NLP Association of India (NLPAI) Machine Learning Contest.

Bai, S. (2009). Language models learning for domain-specific natural language user interaction Robotics and Biomimetics (ROBIO), In 2009 IEEE International Conference on, (p. 2480). Also available at http://ieeexplore. ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5420442

Banchs, R. E. & Codina, J. (2009). On the incidence of part-of-speech on polarity identification of user-generated-contents in spanish. In Proceedings of the First Workshop on Opinion Mining and Sentiment Analysis at CAEPIA-TTIA.

Banko, M. & Moore, R. C. (2004). Part of speech tagging in context In Proceedings of the 20th International Conference on Computational Linguistics COLING 04. Stroudsburg, PA, USA: Association for Computational Linguistics. Also available at http://dx.doi.org/10.3115/ 1220355.1220435

Bird, S., Klein, E. & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. North, Sebastopol, CA 95472 : O'Reilly Beijing. ISBN 978-0-596-51649-9 Also available at http://www.nltk.org/book

Biswas, S. (2009). A hybrid oriya named entity recognition system: Integrating hmm with maxent. Emerging Trends in Engineering and Technology (ICETET), In 2009 2nd International Conference on, (p. 639-643). Washington, DC, USA: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5395485

Bollen, J., Mao, H. & Zeng, X.-J. (2012) Twitter mood predicts the stock market. In Journal of Computational Science 2(1), 1–8. Also available at http://arxiv.org/pdf/1010.3003.pdf

Brill, E. (1995). Unsupervised learning of disambiguation rules for part of speech tagging. In Natural Language Processing Using Very Large Corpora (p. 1-13). Kluwer Academic Press.

Chao, C.-C. (1993). Hidden markov models for the burst error statistics of viterbi decoding Communications, 1993. In ICC '93, Technical Program,

Conference Record, IEEE International Conference on, Volume 2, (p. 751-755). Geneva, Switzerland: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=397374

Chien, L.-F., Chen, K.-J. & Lee, L.-S. (1993). A best-first language processing model integrating the unification grammar and markov language model for speech recognition applications Speech and Audio Processing, IEEE Transactions on, 1(2), 221-240. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=222880

Churbanov, A. G. & Winters-Hilt, S. (2008). Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory. In Proceedings of BMC Bioinformatics, 9.

Cnet.com, Augmented-reality contact lenses to be human-ready at CES (2014). http://www.cnet.com/news/augmented-reality-contact-lenses-to-be-human-ready-at-ces/ (9th of May 2014)

Dandapat, S., Sarkar, S. & Anupam, B. (2004). A hybrid model for part-of-speech tagging and its application to bengali. In Okatan, Ali, International Conference on Computational Intelligence (p. 169-172). Istanbul, Turkey: International Computational Intelligence Society. ISBN 975-98458-1-4.

De Holanda Maia, M. R. & Xexéo, G. B. (2011). Part-of-speech tagging of portuguese using hidden markov models with character language model emissions. In Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology, (p. 159-163). Cuiabá, MT, Brazil: Sociedade Brasileira de Computação.

Denis, P. & Sagot, B. (2012). Coupling an annotated corpus and a lexicon for state-of-the-art pos tagging. In Language Resources and Evaluation, 46(4), 721-736. Secaucus, NJ, USA: Springer-Verlag Inc.

Derczynski, L., Ritter, A., Clark, S. & Bontcheva, K. (2013). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. Proceedings of Recent Advances in Natural Language Processing, (p. 198–206). Hissar, Bulgaria: ACL. Also available at http://aclweb.org/anthology/R/R13/R13-1026.pdf

Ekbal, A., Haque, R. & Bandyopadhyay, S. (2007). Bengali part of speech tagging using conditional random field, In Proceedings of the 7th International Symposium of Natural Language Processing (SNLP 2007), (p. 131-136). Thailand: Springer. Also available at http://www.computing.dcu.ie/~rhaque/SNLP_rejwanul.pdf

Emotiv.com, Brain Controlled Wheelchair (2013). http://www.emotiv.com/ideas/brain-controlled-wheelchair-6.html (9th of May 2014)

Forney, G. D. (1973). The viterbi algorithm, In Proceedings of the IEEE, 61(3), 268-278. Also available at http://dx.doi.org/10.1109/proc.1973.9030

Francis, W. N., & Kucera, H. (1979). Brown corpus manual. Providence, Rhode Island, USA: Department of Linguistics, Brown University. Also available at http://icame.uib.no/brown/bcm.html

Friedman, C., Shagina, L., Lussier, Y. & Hripcsak, G., (2004). Automated Encoding of Clinical Documents Based on Natural Language Processing. In Journal of the American Medical Informatics Association, 11(5), 392-402.

Fromm, P. (1998). Natural language processing for dynamic environments Industrial Electronics Society, 1998. In IECON '98, Proceedings of the 24th Annual Conference of the IEEE, (p. 2018). Niedemhausen, Germany: Elektra. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=724028

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. & Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2 HLT '11 (p. 42-47). Stroudsburg, PA, USA: Association for Computational Linguistics. ISBN 978-1-932432-88-6. Also available at http://dl.acm.org/citation.cfm?id=2002736.2002747

Google.com, Inside Google Translate. https://translate.google.com/about/intl/en_ALL/ (9th of May 2014)

Gupta, S., Malik, S., Pollock, L. & Vijay-Shanker, K. (2013). Part-of-speech tagging of program identifiers for improved text-based software engineering tools Program Comprehension (ICPC), In 2013 IEEE 21st International Conference on, (p. 3-12). San Francisco, CA, USA: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6613828

Greene, B.B. & Rubin, G.M. (1981). Automatic grammatical tagging of English. Providence, RI, USA: Department of Linguistics, Brown University.

Hasan, F. M. (2007) Comparison of unigram, bigram, hmm and brill's pos tagging approaches for some south asian languages. Mohakhali, Dhaka, Bangladesh: Center for Research on Bangla Language Processing, BRAC University. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.3448&rep=rep1&type=pdf (9th of May 2014)

Huang, Z., Eidelman, V. & Harper, M. (2009). Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers NAACL-Short '09 (p. 213-216). Stroudsburg, PA, USA: Association for Computational Linguistics. Also available at http://dl.acm.org/citation.cfm?id=1620853.1620911.

Huberman, B. A., Romero, D. M. & Wu, F. (2009). Social networks that matter: Twitter under the microscope. First Monday, 14(1). Available at http://dx.doi.org/10.2139/ssrn.1313405

Ibtimes.com, How Siri Works: iPhone's 'Brain' Comes from Natural Language Processing, Stanford Professors to Teach Free Online Course (2011). http://www.ibtimes.com/how-siri-works-iphones-brain-comes-natural-language-processing-stanford-professors-teach-free-online (9th of May 2014)

Jung, S.-Y., Park, Y. C., Choi, K. S. & Kim, Y. (1996). Markov random field based english part-of-speech tagging system. In COLING '96: Proceedings of the 16th conference on Computational linguistics, 1(1), 236-242. Stroudsburg,

PA, USA: Association for Computational Linguistics. Also available at http://portal.acm.org/citation.cfm?id=992628.992671

Jurafsky, D. & Martin, J. H. (2009). Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd edition), Chapter 6, Hidden Markov and Maximum Entropy Models, (p. 233-281). Upper Saddle River, New Jersey: Prentice Hall.

Kaku, M. (2011). Physics of the Future: How Science Will Shape Human Destiny and Our Daily Lives by the Year 2100. New York, NY, USA: Knopf Doubleday Publishing Group. ISBN 9780385530811.

Kouloumpis, E., Wilson, T. & Moore, J. (2011). Twitter sentiment analysis: The good the bad and the omg! In Adamic, Lada A., Baeza-Yates, Ricardo A. & Counts, Scott, ICWSM. Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media. Menlo Park, CA, USA: The AAAI Press. Also available at http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2857/3251

Kowalski, G. (2011). Information Retrieval Architecture and Algorithms. Ashburn, VA, USA: Springer.

Kuo, H.-K. J. (2006). Maximum entropy direct models for speech recognition Audio, Speech, and Language Processing, IEEE Transactions on, 14(3), 873-881. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1621200

Langanke, U. H. (2008). Language technology - chomsky grammars and/or fuzzy sets? Applied Machine Intelligence and Informatics. In SAMI 2008, 6th International Symposium on, (p. 85-89). Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4469140

Liu, Y. (2003). Using hidden markov model for information extraction based on multiple templates. Natural Language Processing and Knowledge Engineering. Proceedings. 2003 International Conference on, (p. 394-399). Beijing, China : IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1275937

Loftsson, H. (2009). Correcting a pos-tagged corpus using three complementary methods In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009) (p. 523-531). Athens, Greece: Association for Computational Linguistics. Also available at http://www.aclweb.org/anthology/E09-1060

Lv, C. (2010). An efficient corpus based part-of-speech tagging with gep. In Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on, (p. 289-292). Washington, DC, USA: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5663526

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I CICLing'11 (p. 171-189). Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-19399-6. Also available at http://dl.acm.org/citation.cfm?id=1964799.1964816

Mansouri, A., Suriani, L. A. & Mamat, A. (2008). A new fuzzy support vector machine method for named entity recognition. In Computer Science and Information Technology, 2008. ICCSIT '08. International Conference on, (p. 24-28). Singapore: IEEE Computer Society.

Mashape.com, List of 25+ natural language processing apis (2013). http://blog.mashape.com/post/48946187179/list-of-25-natural-language-processing-apis (9th of May 2014)

McCallum, A., Freitag, D. & Pereira, F. C. N. (2000). Maximum entropy markov models for information extraction and segmentation. In Proceedings of the Seventeenth International Conference on Machine Learning ICML '00 (p. 591-598). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. Also available at http://dl.acm.org/citation.cfm?id=645529.658277.

Momtazi, S. & Klakow, D. (2010). Language Model-Based Sentence Classification for Opinion Question Answering Systems. In Proceedings of the International Multiconference on, Computer Science and Information Technology (p. 251–255). Saarbrücken, Germany: IEEE Computer Society. Also available at http://dx.doi.org/10.1109/IMCSIT.2009.5352718

Morwal, S. & Jahan, N. (2013). Named entity recognition using hidden markov model (HMM): An experimental result on hindi, urdu and marathi languages. International Journal of Advanced Research in Computer Science and Software Engineering, 3(4), 671-675. Also available at http://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I4-0462.pdf

Nguyen, P.-T. (2011). Two entropy-based methods for detecting errors in pos-tagged treebank Knowledge and Systems Engineering (KSE). In 2011 Third International Conference on, (p. 150-156). Washington, DC, USA: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6063458

Outahajala, M. (2013). The development of a fine grained class set for amazigh pos tagging Computer Systems and Applications (AICCSA), In 2013 ACS International Conference on, (p. 1-8). Ifrane, Morocco: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6616440

Owoputi, O., O'Connor, B., C., Dyer, C., Gimpel, K., Schneider, N. & Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In Proceedings of NAACL-HLT (p. 380-390). Atlanta, GA, USA: NAACL.

Pak, A. & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), (p. 1320-1326). Valletta, Malta: European Language Resources Association.

Rabiner, L. & Juang, B.-H. (1986). An introduction to hidden markov models ASSP Magazine, 3(1), 4-16.

Readwrite.com, Apple's Siri Teams Up With Wolfram Alpha (2010). http://readwrite.com/2010/07/08/apples_siri_teams_up_with_wolfram_alpha#awesm=~oDLmIq3KycUNKi (9th of May 2014)

Ritter, A., Clark, S., Mausam, & Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP '11 (p. 1524-1534). Stroudsburg, PA, USA: Association for Computational Linguistics. ISBN 978-1-937284-11-4. Also available at http://dl.acm.org/citation.cfm?id=2145432.2145595

Schubert, L. & Tong, M. (2003). Extracting and Evaluating General World Knowledge from the Brown Corpus. In Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning, Volume 9, (p.7-13). Stroudsburg, PA, USA: Association for Computational Linguistics. Also available at http://dx.doi.org/10.3115/1119239.1119241

Sharma, S. K. (2011). Using hidden markov model to improve the accuracy of punjabi pos tagger. In Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on, 2, (p. 697-701). Shanghai, China: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5952600

Singh, J., Joshi, N. & Mathur, I. (2013). Development of marathi part of speech tagger using statistical approach. In Proceedings of 2013 International Conference on Advances in Computing, Communications and Informatics. Mysore, India: IEEE Computer Society.

Swets.com, Natural language processing (NLP) and academic literature search (Part I) (2013). http://www.swets.com/blog/natural-language-processing-nlp-and-academic-literature-search-part-i (9th of May 2014)

Takano, W. (2012). Bigram-based natural language model and statistical motion symbol model for scalable language of humanoid robots. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, (p. 1232-1237). Saint Paul, MN, USA: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6225331

Taylor, A. M., M., & Santorini, M. (2003). The Penn Treebank: An Overview, Volume 20 (p. 5-22). Netherlands: Springer. ISBN 978-1-4020-1335-5 Also available at http://dx.doi.org/10.1007/978-94-010-0201-1_1

Washingtonpost.com, Everything You Need to Know About Google Glasses (2014), http://www.washingtonpost.com/blogs/the-switch/wp/2014/02/27/everything-you-need-to-know-about-google-glass/ (9th of May 2014)

Wen-qiu, Z. (2012). A conditional random field with loop and its inference algorithm. In Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on, (p. 11-14). Sanya, Hainan, China: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6173136

Yang, X. & Liang, W. (2010). An n-gram-and-wikipedia joint approach to natural language identification. In Universal Communication Symposium (IUCS), 2010 4th International, (p. 332). Beijing, China: IEEE Computer Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5666010

Zin, K. K. (2009). Part of speech tagging for myanmar using hidden markov model. In Current Trends in Information Technology (CTIT), 2009 International Conference on the, (p. 1-6). Dubai, UAE: IEEE Computer

Society. Also available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5423133

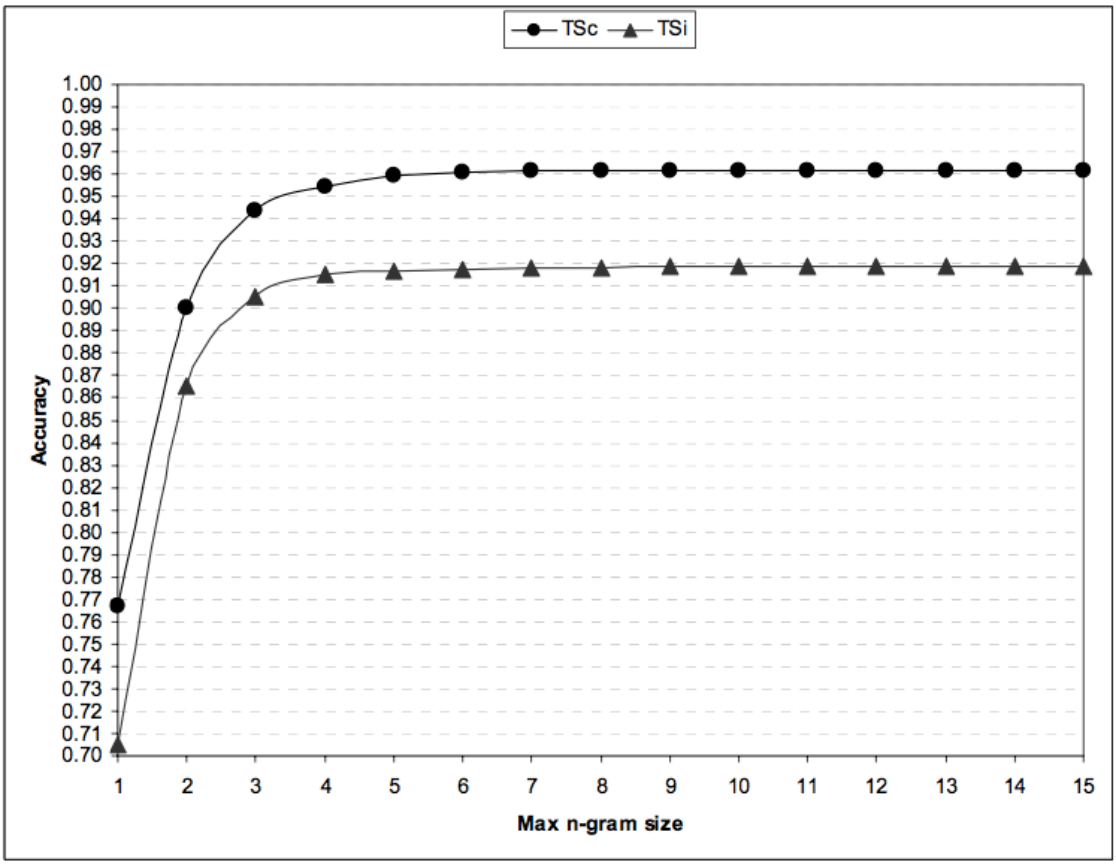
Xie, H., Tang, W., Wen, Y. & Li S. (2013). HMM-based primary user activity in GSM via spectrum measurements. In Wireless Communications & Signal Processing (WCSP), 2013 International Conference on, (p. 1-6). Hangzhou, Zhejiang, China: IEEE Computer Society. Also available at http://dx.doi.org/10.1109/WCSP.2013.6677061

ATTACHMENT 1 – PENN TREEBANK TAG-SET

CC	Coordinating conj.	TO	infinitival *to*
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present pple
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sg. present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sg. present
JJS	Adjective, superlative	WDT	Wh-determiner
LS	List item marker	WP	Wh-pronoun
MD	Modal	WP$	Possessive *wh*-pronoun
NN	Noun, singular or mass	WRB	Wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	”	Right close double quote

Penn TreeBank tag set (Taylor et al., 2003, p. 8)

ATTACHMENT 2 – N-GRAM ACCURACY



HMM accuracy versus max n-gram sizes (de Holanda Maia & Xexéo, 2011, p. 162). TSc and TSi are referring to different tag sets.

ATTACHMENT 3 – GIMPEL TAG-SET

Tag	Description	Examples	%
Nominal, Nominal + Verbal			
N	common noun (NN, NNS)	books someone	13.7
O	pronoun (personal/WH; not possessive; PRP, WP)	it you u meeee	6.8
S	nominal + possessive	books' someone's	0.1
^	proper noun (NNP, NNPS)	lebron usa iPad	6.4
Z	proper noun + possessive	America's	0.2
L	nominal + verbal	he's book'll iono (= *I don't know*)	1.6
M	proper noun + verbal	Mark'll	0.0
Other open-class words			
V	verb incl. copula, auxiliaries (V*, MD)	might gonna ought couldn't is eats	15.1
A	adjective (J*)	good fav lil	5.1
R	adverb (R*, WRB)	2 (i.e., *too*)	4.6
!	interjection (UH)	lol haha FTW yea right	2.6
Other closed-class words			
D	determiner (WDT, DT, WP$, PRP$)	the teh its it's	6.5
P	pre- or postposition, or subordinating conjunction (IN, TO)	while to for 2 (i.e., *to*) 4 (i.e., *for*)	8.7
&	coordinating conjunction (CC)	and n & + BUT	1.7
T	verb particle (RP)	out off Up UP	0.6
X	existential *there*, predeterminers (EX, PDT)	both	0.1
Y	**X** + verbal	there's all's	0.0
Twitter/online-specific			
#	hashtag (indicates topic/category for tweet)	#acl	1.0
@	at-mention (indicates another user as a recipient of a tweet)	@BarackObama	4.9
~	discourse marker, indications of continuation of a message across multiple tweets	RT and : in retweet construction RT @user : hello	3.4
U	URL or email address	http://bit.ly/xyz	1.6
E	emoticon	:-) :b (: <3 o__O	1.0
Miscellaneous			
$	numeral (CD)	2010 four 9:30	1.5
,	punctuation (#, $, '', (,), ,, ., :, ``)	!!! .... ?!?	11.6
G	other abbreviations, foreign words, possessive endings, symbols, garbage (FW, POS, SYM, LS)	ily (*I love you*) wby (*what about you*) 's ♫ --> awesome...I'm	1.1

"The set of tags used to annotate tweets. The last column indicates each tag's relative frequency in the full annotated data (26,435 tokens). (The rates for M and Y are both < 0.0005.)" (Gimpel et al., 2011)

ATTACHMENT 4 – BROWN CORPUS TAG REPLACEMENTS

TABLE 4: Brown Corpus (Greene et al., 1981) POS tags (Brown) replaced with matching Gimpel et al. (2011) tags (Gimpel)

Brown	Gimpel	Brown	Gimpel	Brown	Gimpel
(	,	EX	X	FW-UH	G
)	,	EX+BEZ	Y	FW-VB	G
*	,	EX+HVD	Y	FW-VBD	G
,	,	EX+HVZ	L	FW-VBG	G
--	,	EX+MD	Y	FW-VBN	G
.	,	FW-*	G	FW-VBZ	G
:	,	FW-AT	G	FW-WDT	G
ABL	X	FW-AT+NN	G	FW-WPO	G
ABN	X	FW-AT+NP	G	FW-WPS	G
ABX	X	FW-BE	G	HV	V
AP	X	FW-BER	G	HV*	V
AP$	S	FW-BEZ	G	HV+TO	V
AP+AP	X	FW-CC	G	HVD	V
AT	D	FW-CD	G	HVD*	V
BE	V	FW-CS	G	HVG	V
BED	V	FW-DT	G	HVN	V
BED*	V	FW-DT+BEZ	G	HVZ	V
BEDZ	V	FW-DTS	G	HVZ*	V
BEDZ*	V	FW-HV	G	IN	P
BEG	V	FW-IN	G	IN+IN	P
BEM	V	FW-IN+AT	G	IN+PPO	P
BEM*	V	FW-IN+NN	G	JJ	A
BEN	V	FW-IN+NP	G	JJ$	A
BER	V	FW-JJ	G	JJ+JJ	A
BER*	V	FW-JJR	G	JJR	A
BEZ	V	FW-JJT	G	JJR+CS	A
BEZ*	V	FW-NN	G	JJS	A
CC	&	FW-NN$	G	JJT	A
CD	$	FW-NNS	G	MD	V
CD$	$	FW-NP	G	MD*	V
CS	&	FW-NPS	G	MD+HV	V
DO	V	FW-NR	G	MD+PPSS	V
DO*	V	FW-OD	G	MD+TO	V
DO+PPSS	V	FW-PN	G	NN	N
DOD	V	FW-PP$	G	NN$	S
DOD*	V	FW-PPL	G	NN+BEZ	S
DOZ	V	FW-PPL+VBZ	G	NN+HVD	S
DOZ*	V	FW-PPO	G	NN+HVZ	S
DT	D	FW-PPO+IN	G	NN+IN	S
DT$	S	FW-PPS	G	NN+MD	S
DT+BEZ	D	FW-PPSS	G	NN+NN	N
DT+MD	D	FW-PPSS+HV	G	NNS	N
DTI	X	FW-QL	G	NNS$	S
DTS	S	FW-RB	G	NNS+MD	M
DTS+BEZ	S	FW-RB+CC	G	NP	^
DTX	D	FW-TO+VB	G	NP$	Z

TABLE 4 continues on the next page.

TABLE 4 continues here from the previous page.

Brown	Gimple	Brown	Gimpel
NP+BEZ	Z	RP+IN	R
NP+HVZ	Z	TO	P
NP+MD	Z	TO+VB	V
NPS	^	UH	!
NPS$	Z	VB	V
NR	N	VB+AT	V
NR$	S	VB+IN	V
NR+MD	^	VB+JJ	V
NRS	N	VB+PPO	V
OD	$	VB+RP	V
PN	O	VB+TO	V
PN$	O	VB+VB	V
PN+BEZ	L	VBD	V
PN+HVD	L	VBG	V
PN+HVZ	L	VBG+TO	V
PN+MD	L	VBN	V
PP$	S	VBN+TO	V
PP$$	S	VBZ	V
PPL	S	WDT	V
PPLS	S	WDT+BER	V
PPO	S	WDT+BER+PP	V
PPS	O	WDT+BEZ	V
PPS+BEZ	L	WDT+DO+PPS	V
PPS+HVD	L	WDT+DOD	V
PPS+HVZ	L	WDT+HVZ	L
PPS+MD	L	WP$	S
PPSS	O	WPO	O
PPSS+BEM	L	WPS	O
PPSS+BER	L	WPS+BEZ	S
PPSS+BEZ	L	WPS+HVD	S
PPSS+BEZ*	L	WPS+HVZ	L
PPSS+HV	L	WPS+MD	L
PPSS+HVD	L	WQL	P
PPSS+MD	L	WRB	R
PPSS+VB	L	WRB+BER	R
QL	P	WRB+BEZ	R
QLP	P	WRB+DO	R
RB	R	WRB+DOD	R
RB$	S	WRB+DOD*	R
RB+BEZ	R	WRB+DOZ	R
RB+CS	R	WRB+IN	R
RBR	R	WRB+MD	R
RBR+CS	R		
RBT	R		
RN	R		
RP	R		

ATTACHMENT 5 – PENN TREEBANK TAG REPLACEMENTS

TABLE 5: Penn TreeBank (Taylor et al., 2003) POS tags (Penn) replaced with matching Gimpel et al. (2011) tags (Gimpel)

Penn	Gimpel
CC	&
CD	$
DT	D
EX	X
FW	G
IN	P
JJ	A
JJR	A
JJS	A
LS	G
MD	V
NN	N
NNS	N
NNP	^
NNPS	^
PDT	X
POS	G
PRP	O
PRP$	D
RB	R
RBR	R
RBS	R
RP	T
TO	P
UH	!
VB	V
VBD	V
VBG	V
VBN	V
VBP	V
VBZ	V
WDT	D
WP	O
WP$	D
WRB	R

ATTACHMENT 6 – N-GRAM TAGGER CODE

```
import nltk, nltk.data
from nltk.corpus import conll2000, treebank, brown
from nltk.tokenize import word_tokenize, wordpunct_tokenize, sent_tokenize
from nltk.tag.sequential import UnigramTagger
# tagger training data
tagged_sents = conll2000.tagged_sents('train.txt')
# tagger training data with clustered words
clustered_sents = conll2000.tagged_sents('train_clustered.txt')
names = {} # dict of names
brown_replace_tags = {} # dict for replacing Brown tags with Owoputi et al. tags
penn_replace_tags = {} # dict for replacing Penn tags with Owoputi et al. tags
emoticons = [] # list of popularly used emoticons

# Builds a dict with all possible tokens as keys where each item
# is containing a reference to proper cluster
class TwitterCluster:
   clusters = {}
   # builds clusters to attribute dict called "clusters"
   def __init__(self):
      ...

   # Returns the most likely token in a cluster containing given token
   def get_most_likely_token(self, token):
      ...

# This class is based on standard unigram tagger.
# The class tags twitter specific markup.
class TwitterTagging(UnigramTagger):
   # Calls UnigramTagger constructor
   def __init__(self, train=None, model=None, backoff=None, cutoff=0, \
      verbose=False):
      ...

   # Replaces tag chooser with tweet markup finder
   def choose_tag(self, tokens, index, history):
      ...

# This class is based on standard unigram tagger.
# The class is used for noun detection
class UnigramReplace(UnigramTagger):
   # Calls UnigramTagger constructor
   def __init__(self, train=None, model=None, backoff=None, cutoff=0, \
      verbose=False):
      ...

   # Replaces tag chooser with noun finder
   def choose_tag(self, tokens, index, history):
      ...
```

The code continues on the next page.

The code continues here from the previous page.

```
# Methods to annotate single tweet
class Tweet(object):
    tokens = []
    clustered_tokens = []
    tagging_results = []

    # Builds list of tokens and clustered tokens with corresponding key numbers
    def __init__(self, linestring):
        ...

    # Returns tokinzed tweet
    def get_tokens(self):
        ...

    # Returns tokinzed tweet with clustered words
    def get_likely_tokens(self):
        ...

    # Builds list of tagging results with same key values than given tokens
    def set_tagging(self, tags):
        ...

    # Checks names from current tweet by comparing untagged words to tweet list
    def check_names(self):
        ...

    # Checks whether the token is found in emoticon list
    def check_emoticons(self):
        ...

    # Sets missing tags to have same tagging as previous correct tag
    def check_relateness(self):
        ...

    # Compares tagger results to correct tagging
    def compare_result(self, resultfile, tweetno):
        ...
```

The code continues on the next page.

The code continues here from the previous page.

```
class ReadTweets():
   tweets = []
   # constructs "tweets" attribute array by iterating each line in the file and
   # placing a tweet-object instance
   def __init__(self):

   # Actual tagging process
   def tag(self, filename):
      resultfile = open(filename, 'w')
      for tw in self.tweets:
         i += 1
         tokenized = t.get_tokens()
         tokenized_likely = t.get_likely_tokens()
         basetagger = nltk.DefaultTagger('None')
         tw_tagging = TwitterTagging (tagged_sents, backoff=basetagger)
         tw.set_tagging(tw_tagging.tag(tokenized))
         # Tagger core begins
         tagger_level_1 = nltk.UnigramTagger(tagged_sents, backoff=basetagger)
         tagger_level_2 = nltk.BigramTagger(tagged_sents, backoff= tagger_level_1)
         tw.set_tagging(tagger_level_2.tag(tokenized))
         # Tagger core ends
         # Use of additional tools begins
         penn_names = UnigramReplace(treebank.tagged_sents(), \
            proper_tags = penn_replace_tags, backoff=basetagger)
         brown_names = UnigramReplace(brown.tagged_sents(), \
            proper_tags = brown_replace_tags, backoff=penn_names)
         tw.set_tagging(brown_names.tag(tokenized_likely))
         tw.check_names()
         tw.check_emoticons()
         tw.check_relateness()
         # Use of additional tools begins
         # building a filen containing tagging results
         tw.compare(resultfile, i)
      resultfile.close()


tweet_reader = ReadTweets() # tagger instance
tweet_reader.tag('untagged_tweets.txt') # running the tagger
```

ATTACHMENT 7 – N-GRAM TAGGER CLASS DIAGRAM