Natural Language Processing

Unit 1: Lexical analysis

January 2020

Lexical analysis

Definition

The lexical analysis in NLP deals with the study at the level of words with respect to their lexical meaning and part-of-speech. This level of linguistic processing utilizes a language's lexicon, which is a collection of individual lexemes. A lexeme is a basic unit of lexical meaning; which is an abstract unit of morphological analysis that represents the set of forms or "senses" taken by a single morpheme.

Why is important?

- Determine the words in the domain of our applications can reduce complexity significativaly

Why is important?

- Determine the words in the domain of our applications can reduce complexity significativaly
- Text structure, placements and nature of the words provides valuable information for classification tasks

Why is important?

- Determine the words in the domain of our applications can reduce complexity significativaly
- Text structure, placements and nature of the words provides valuable information for classification tasks
- Enrichment process of the text provide valuable features for classifcation

Why is important?

- Determine the words in the domain of our applications can reduce complexity significativaly
- Text structure, placements and nature of the words provides valuable information for classification tasks
- Enrichment process of the text provide valuable features for classifcation
- Can transform the input for sintactic and semantic analysis

Lexicon

Lexicon

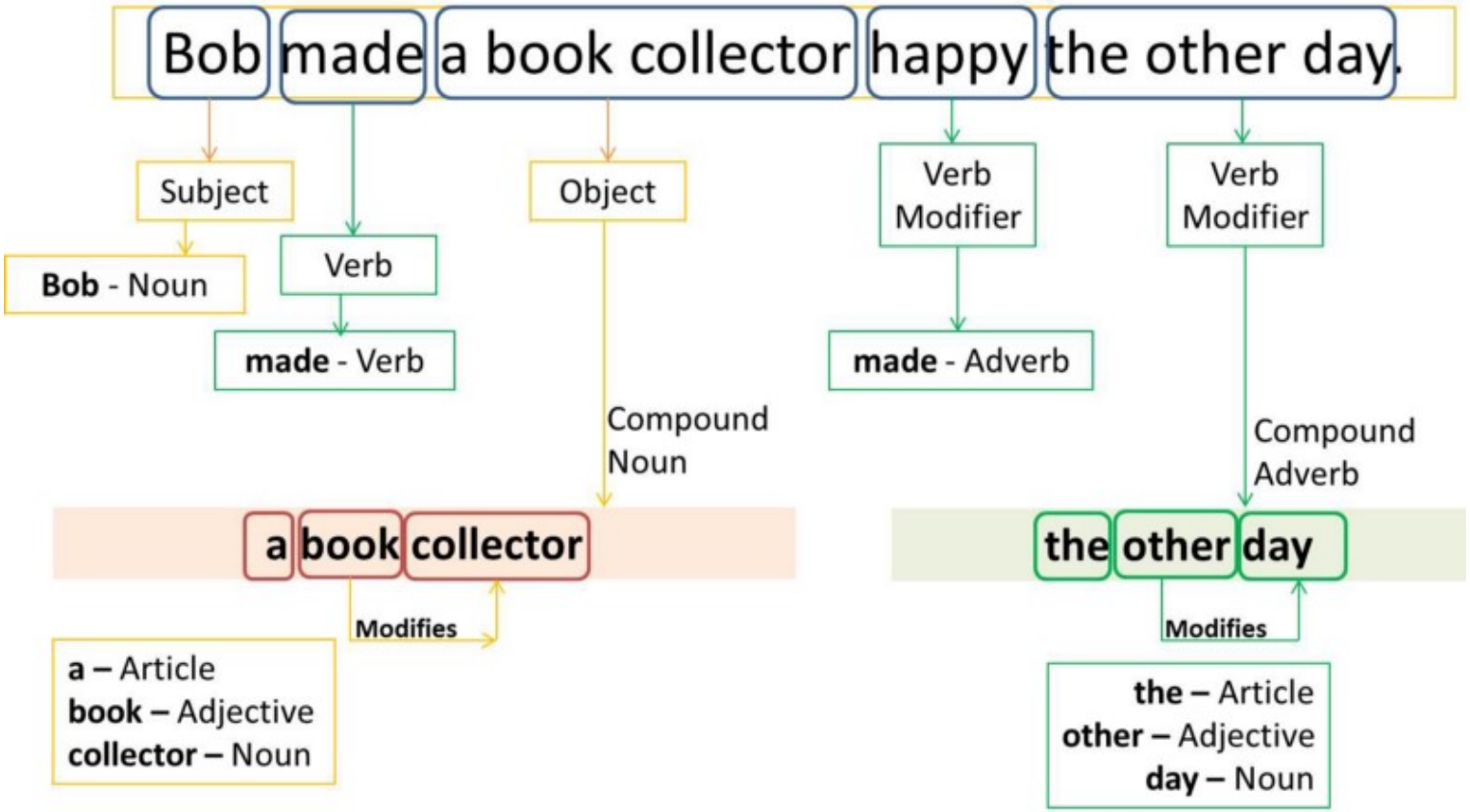
A lexicon, or lexical resource, is a collection of words and/or phrases along with associated information, such as part-of-speech and sense definitions. Lexical resources are secondary to texts, and are usually created and enriched with the help of texts.

Part of Speech Tagging

Definition

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

Part of Speech Tagging (POS)



Retrieved from https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/

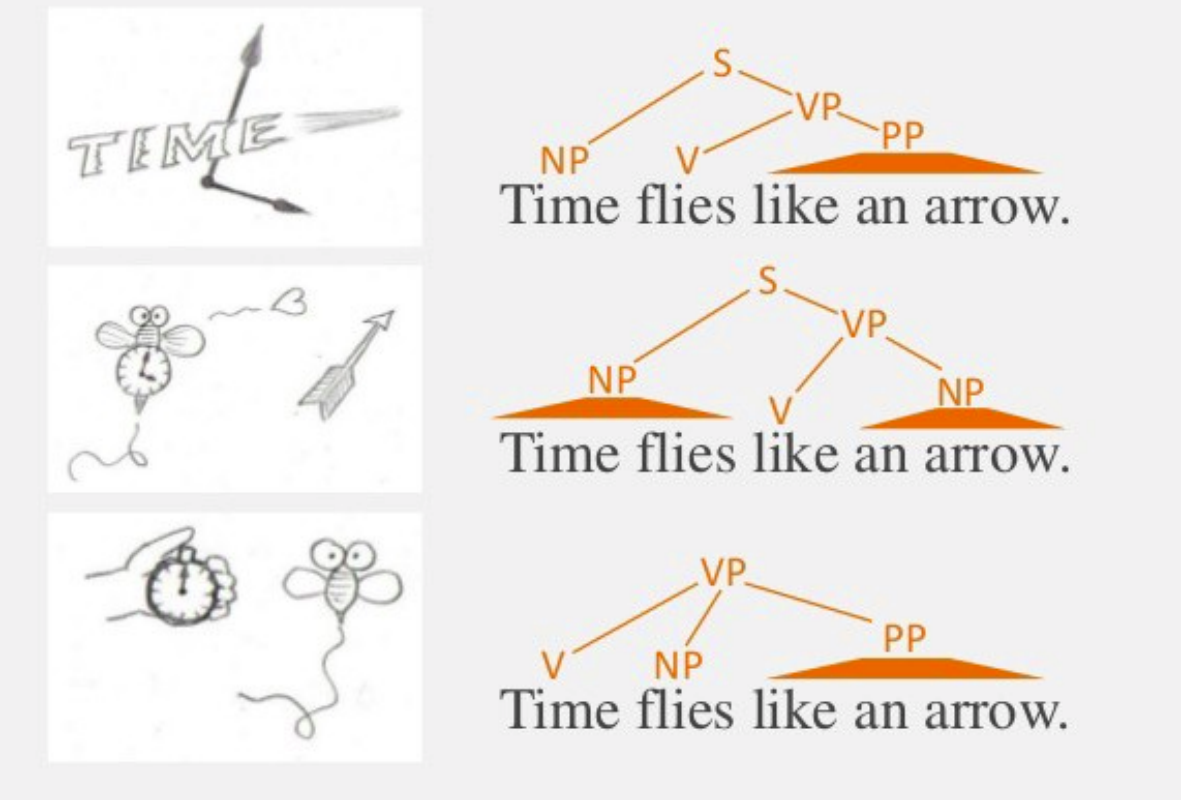
POS

Identifying part of speech tags is much more complicated than simply mapping words to their part of speech tags:

- POS tagging is not something that is generic
- A single word may have a different part of speech tag in different:
  - sentences
  - contexts

POS

Word sense ambiguity



Retrieved from https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/

Word Sense Disambiguation

The above example shows us that a single sentence can have three different POS tag sequences assigned to it that are equally likely. That means that it is very important to know what specific meaning is being conveyed by the given sentence whenever it's appearing. Word sense disambiguation, as we are trying to find out THE sequence is the main problem POS tagging is trying to solve.

POS tagger

POS-tagging algorithms fall into two distinctive groups:

- Rule-Based POS Taggers
- Stochastic POS Taggers

Automatic part of speech tagging is an area of natural language processing where statistical techniques have been more successful than rule-based methods.

Rule Based Tagging

E. Brill's tagger, one of the first and most widely used English POS-taggers, employs rule-based algorithms. It goes through the training data and finds out the set of tagging rules that best define the data and minimize POS tagging errors. Rules are found out using the corpus provided and a set of rule templates that the model can use to come up with new features.

Rule Based Tagging

- Typical rule-based approaches use contextual information to assign tags to unknown or ambiguous words.
- Defining a set of rules manually is an extremely cumbersome process and is not scalable at all.
- Disambiguation is done by analyzing the linguistic features of the word, its preceding word, its following word, and other aspects.

Example of a rule: If an ambiguous/unknown word X is preceded by a determiner and followed by a noun, tag it as an adjective.

Stochastic Part of speech tagging

The simplest stochastic taggers disambiguate words based solely on the probability that a word occurs with a particular tag. In other words, the tag encountered most frequently in the training set with the word is the one assigned to an ambiguous instance of that word.
An alternative to the word frequency approach is to calculate the probability of a given sequence of tags occurring. This is sometimes referred to as the n-gram approach, referring to the fact that the best tag for a given word is determined by the probability that it occurs with the n previous tags.

Stochastic Part of speech tagging

The next level of complexity that can be introduced into a stochastic tagger combines the previous two approaches, using both tag sequence probabilities and word frequency measurements. This is known as the **Hidden Markov Model (HMM)**.

Markov Models

Markov Chain is essentially the simplest known Markov model, that is it obeys the Markov property.
The Markov property suggests that the distribution for a random variable in the future depends solely only on its distribution in the current state, and none of the previous states have any impact on the future states.

$$P(q_1, ..., q_n) = \prod_{i=1}^{n} P(q_i|q_{i-1}) \tag{1}$$

Markov Models for POS

The probability of a certain chain of tokens to be labeled with acertain set of tags
In the part of speech tagging problem, the observations are the words themselves in the given sequence.
As for the states, which are hidden, these would be the POS tags for the words.
The transition probabilities would be the probability of the current word having a particular tag given that the previous tag was X.

POS problem

In tagging problems, each x(i) would be a sequence of words X1 X2 X3 .... Xn(i), and each y(i) would be a sequence of tags Y1 Y2 Y3 ... Yn(i)(we use n(i)to refer to the length of the i'th training example). X would refer to the set of all sequences x1 . . . xn, and Y would be the set of all tag sequences y1 . . . yn. Our task would be to learn a function f : X $\rightarrow$ Y that maps sentences to tag sequences. We can model the joint probability as:

$$p(x, y) = p(y)p(x|y) \tag{2}$$

Markov Models for POS

To calculate the transition probabilities for a Markov Model in the POS context we need a text corpus which has the labels of the words diveded by sentences:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Retrieved from https://www.freecodecamp.org/news/a-deep-dive-into-part-of-speech-tagging-using-viterbi-algorithm-17c8de32e8bc/

Markov models hidden state transitions

To calculate the probability we use the classic approach of favorable cases vs all cases. For example if we want to know which tags are probable to be the beggining of the sentence we count which tags are present at the beggining of a sentence and how many times we see them. After that we can build a diagram to express the transitions.

Markov Models for POS

Let's count how many sentences start with the tag VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count how many sentences start with the tag VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count how many sentences start with the tag VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

2 out of 10

Markov Models for POS

Let's count how many sentences start with the tag NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count how many sentences start with the tag NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count how many sentences start with the tag NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

8 out of 10

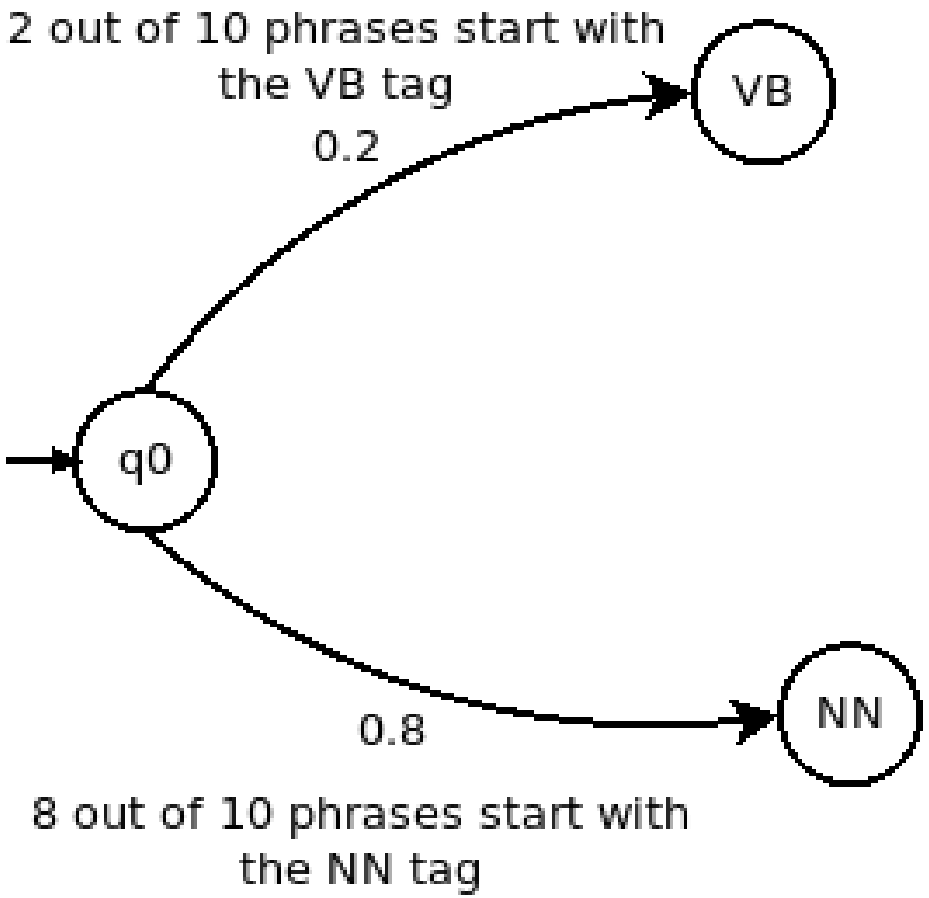
Markov Models for POS

So the probabilites of a sentence to start wirh a tag are calculated dividing the oservations of each tag between the total observations of starting tags:

- NN: $\frac{8}{10} = 0.8$
- VB: $\frac{2}{10} = 0.2$

Markov Models for POS

We can express that probability using a diagram:



Markov Models for POS

The probability of a transition following the Markov property only depends on the current state, so to calculate the probability of a following tag given a specific one is looking for the bigrams (concatenation of two tags) starting with the desired tag.

Markov Models for POS

Let's count the appearences of the bigram NN/IN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

NN/IN

Markov Models for POS

Let's count the appearences of the bigram NN/IN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count the appearences of the bigram NN/IN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

$$\text{NN/IN} = 2$$

Markov Models for POS

Let's count the appearences of the bigram NN/NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

NN/NN

Markov Models for POS

Let's count the appearences of the bigram NN/NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count the appearences of the bigram NN/NN:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

$$\text{NN/NN} = 10$$

Markov Models for POS

Let's count the appearences of the bigram NN/VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

NN/VB

Markov Models for POS

Let's count the appearences of the bigram NN/VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

Markov Models for POS

Let's count the appearences of the bigram NN/VB:

eat/VB breakfast/NN at/IN morning/NN time/NN
take/VB time/NN with/IN arrow/NN projects/NN
horse/NN riders/NN like/VB the/DT airport/NN
paper/NN flies/VB on/IN hydrogen/NN gas/NN
bees/NN sting/VB like/IN some/DT flies/NN
beans/NN soil/VB an/DT iron/NN grill/NN
flies/NN smell/VB an/DT arrow/NN drink/NN
people/NN like/VB an/DT army/NN arrow/NN
dinner/NN time/NN flies/VB all/DT day/NN
horse/NN flies/NN time/VB morning/NN rays/NN

$$\text{NN/VB} = 8$$

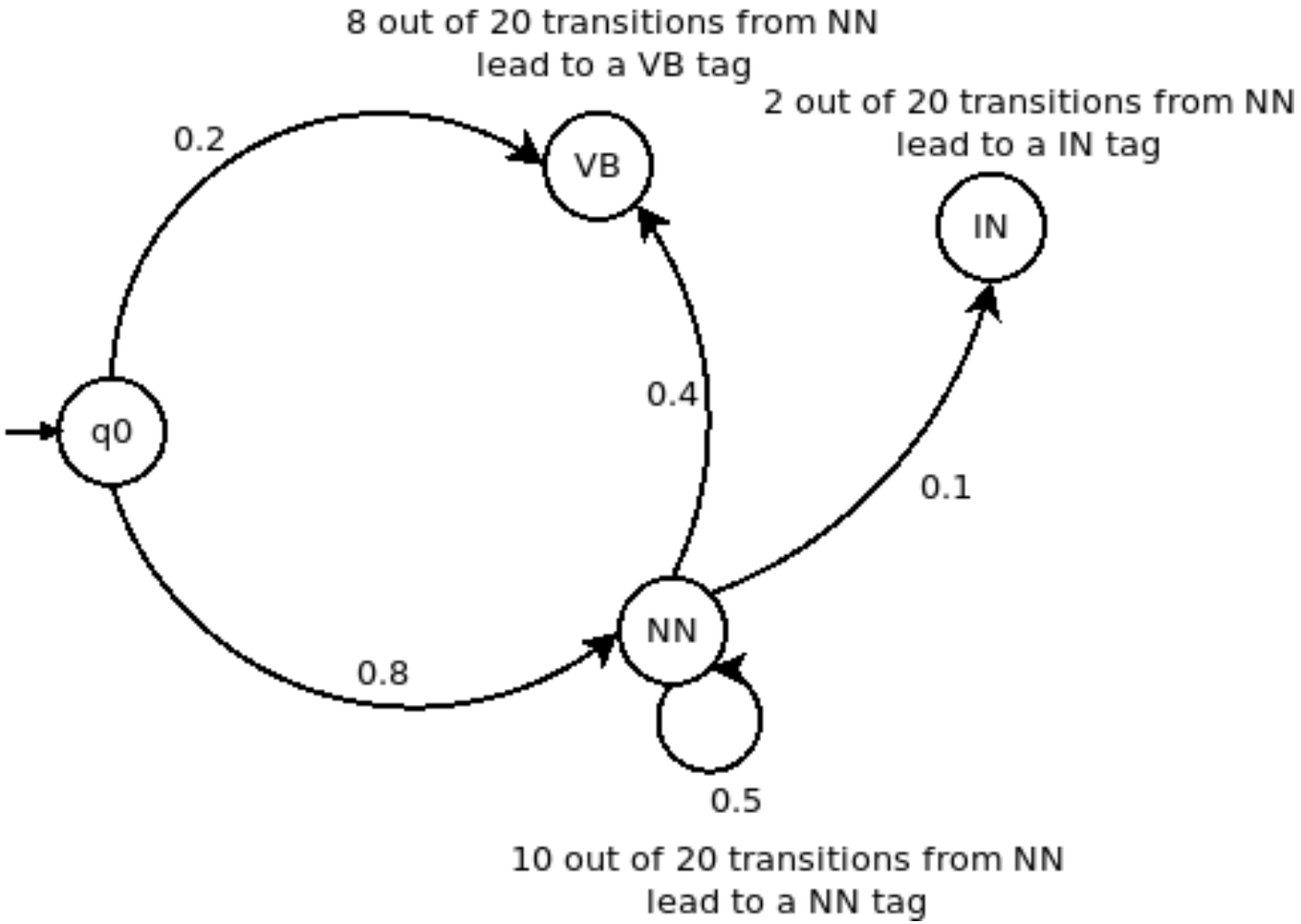
Markov Models for POS

So the probabilites of transitions for the NN tags are calculated dividing the oservations of the bigram between the total observations of all bigrams starting with NN:

- NN/IN: $\frac{2}{20} = 0.1$
- NN/NN: $\frac{10}{20} = 0.5$
- NN/VB: $\frac{8}{20} = 0.4$

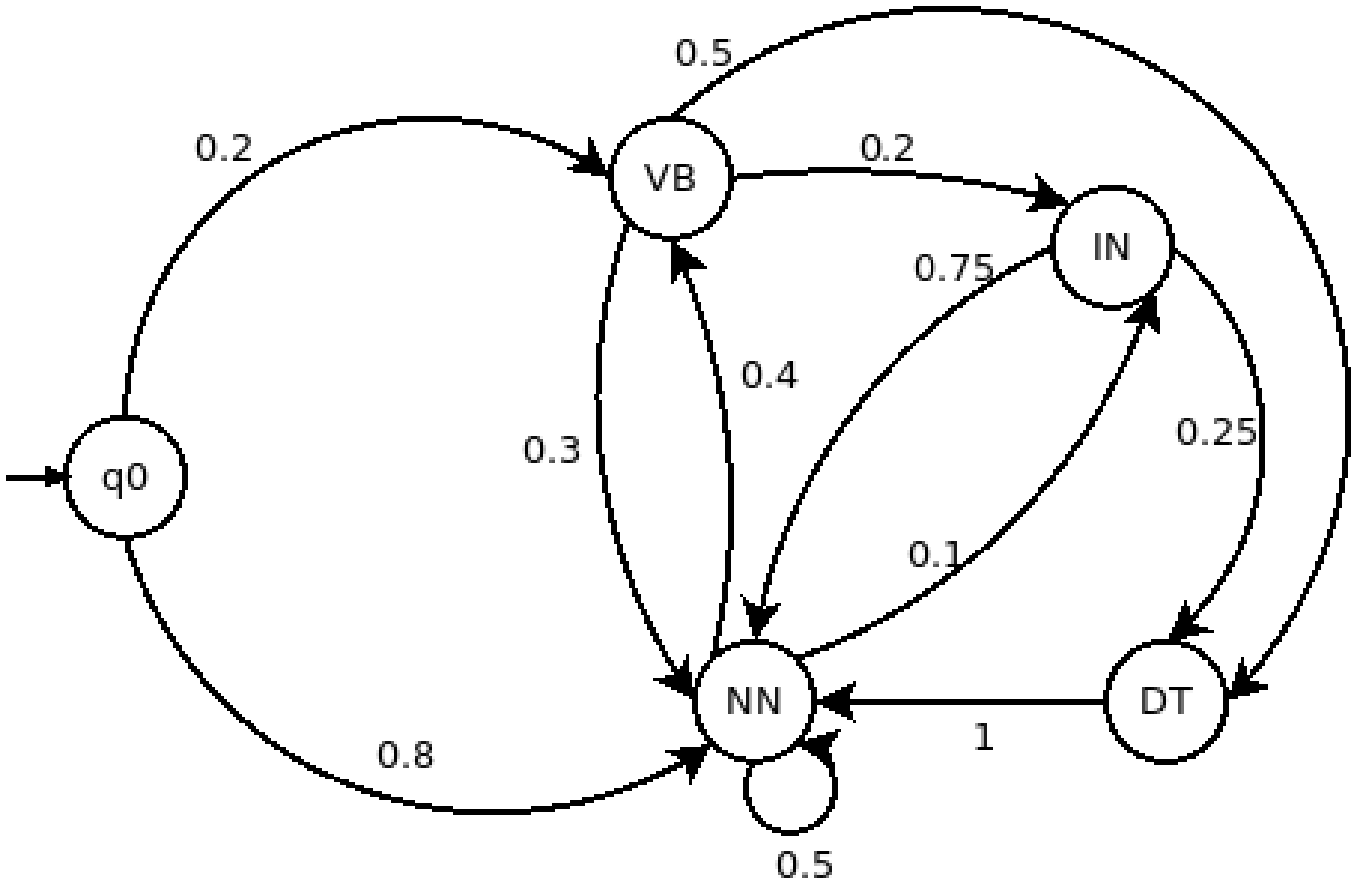
Markov Models for POS

We can express that probability using a diagram:



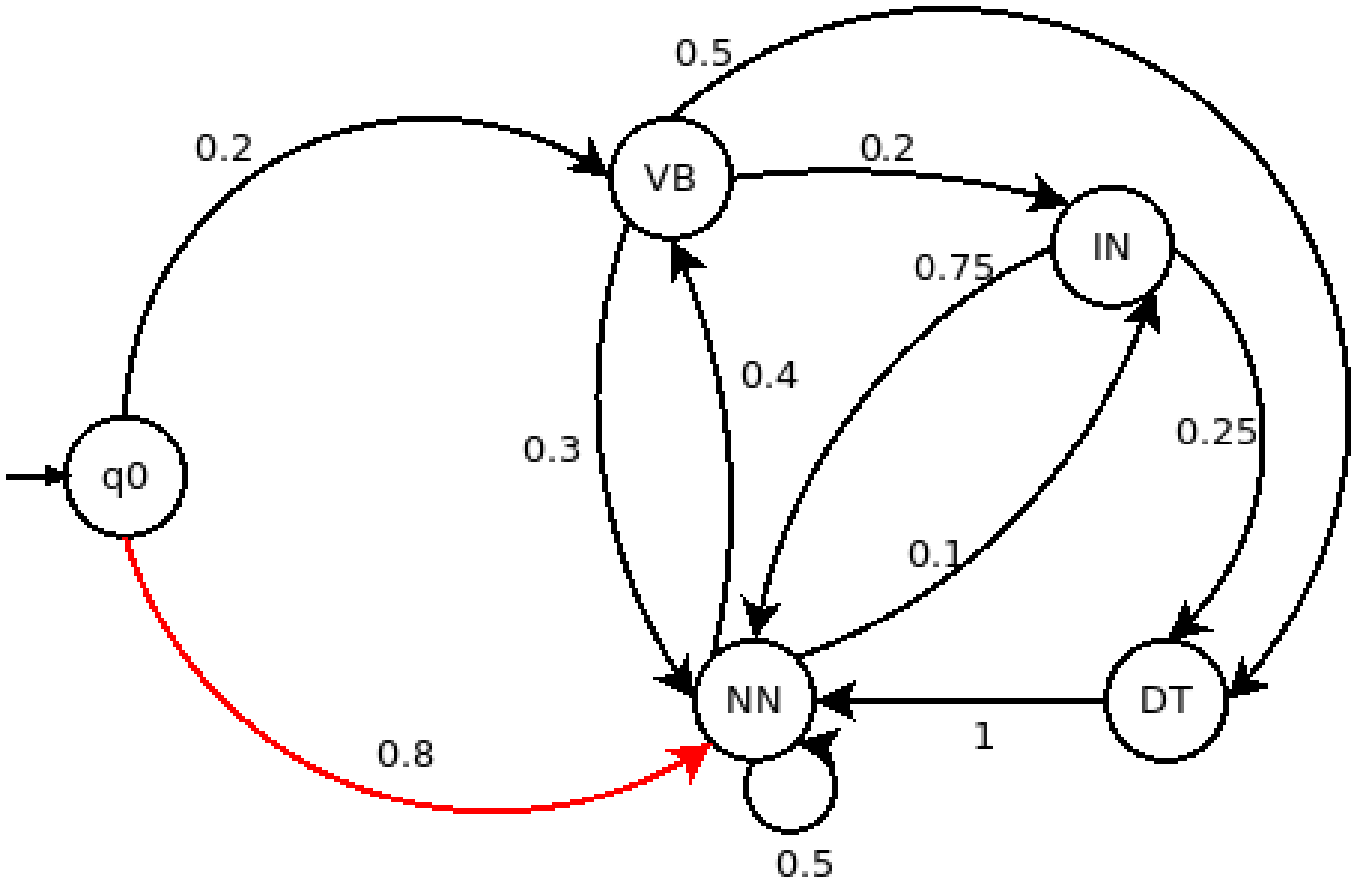
Marko Models for POS

If we follow the same process for each tags we will end with a diagram like the following:



Markov Models for POS

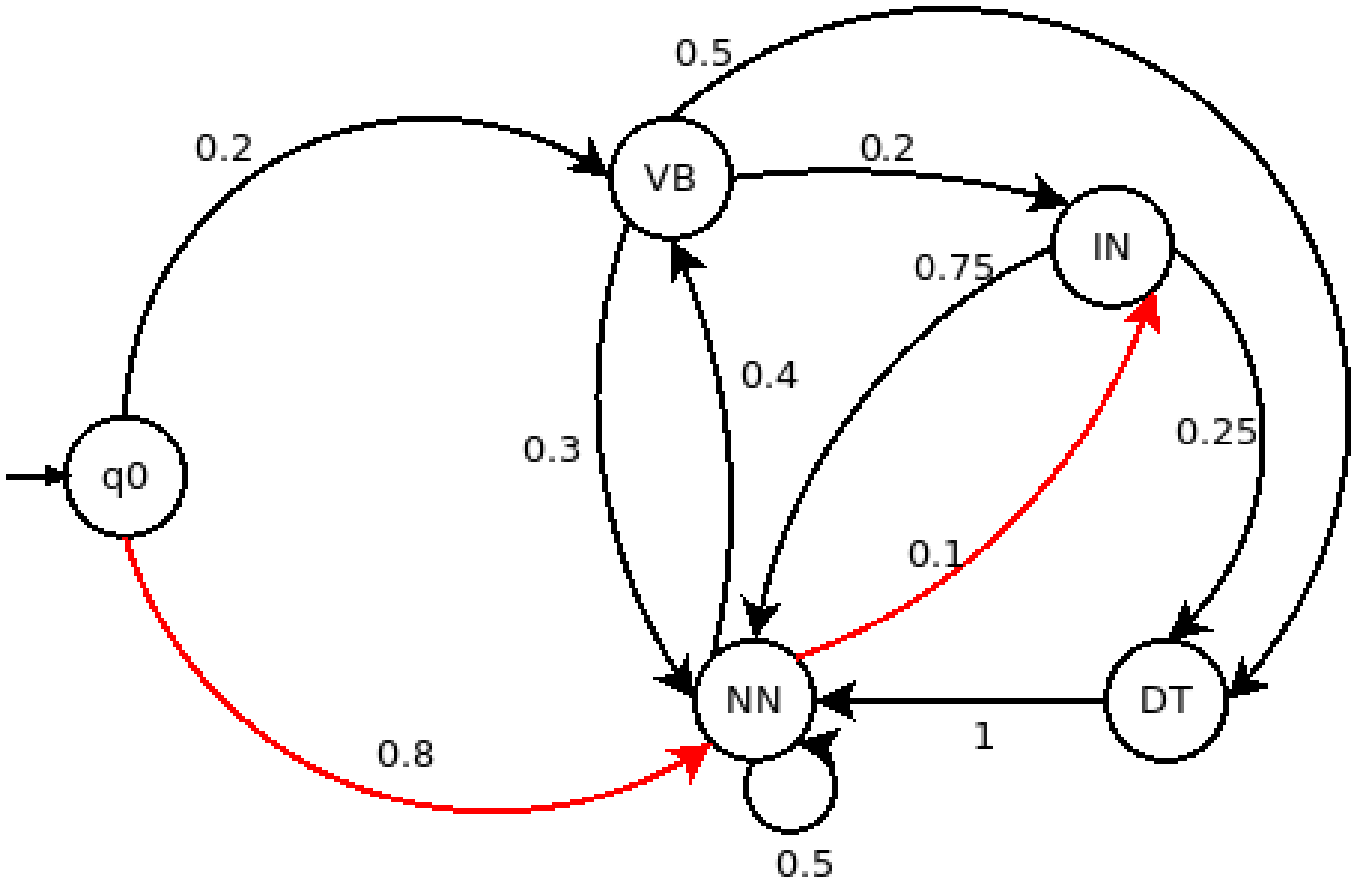
To calculate the probability of the chain NN/IN/DT we perform the product:



Probability of the chain NN, IN, DT = 0.8

Markov Models for POS

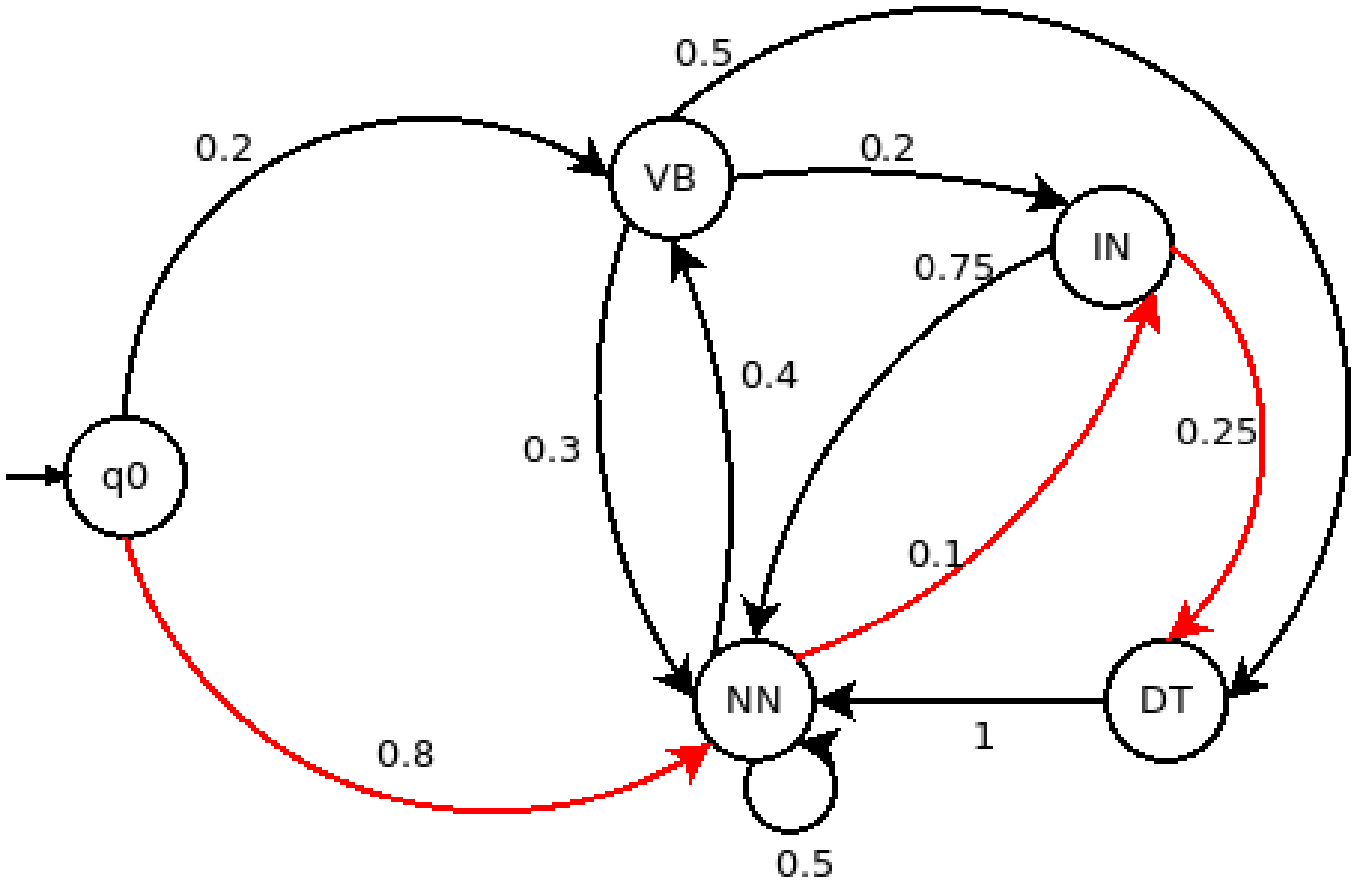
To calculate the probability of the chain NN/IN/DT we perform the product:



Probability of the chain NN, IN, DT $= 0.8 * 0.1$

Markov Models for POS

To calculate the probability of the chain NN/IN/DT we perform the product:



Probability of the chain NN, IN, DT $= 0.8 * 0.1 * 0.25 = 0.02$

Markov Models for POS

Even if they are not expressed in the diagram the probabilities to go from one tag to any other exists since we have a stochastic parser, which meas the transition state diagram is a fully connected graph. If I have 20 different tags and a chain of length 10 how many possible markovian chains can I build?

Markov Models for POS

Even if they are not expressed in the diagram the probabilities to go from one tag to any other exists since we have a stochastic parser, which meas the transition state diagram is a fully connected graph. If I have 20 different tags and a chain of length 10 how many possible markovian chains can I build?

$$20^{10} = 20000000000$$

each one performing 10 multiplications which means approx 205 teraflops

Viterbi algorithm

How can we determine the best chains given a set of markovian chains:

Viterbi algorithm

How can we determine the best chains given a set of markovian chains:

Viterbi algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).

Viterbi algorithm intuition

https://www.youtube.com/watch?v=6JVqutwtzmo

Viterbi for POS

Viterbi algorithm uses:

Q : *Set of possible states*

V : *Set of observable symbols*

A : *Set of probabilties of transitions*

B : *Set of probabilities of observations*

Viterbi for POS

To perform a POS tag using viterbi algorithm we need to build:

Q : *All possible tags*

V : *All words in lexicon*

A : *Probability transitions between categories*

B : *Probability of a word to belong a certain category*

Let's code

Assignments

Assignment 3: Build a pos tagger **class** using the resources seen in class, considering a simple bigram model.

- The class must be able to build it's model from a given dataset (a file with the strings as lines).
- The class must implement a evaluate chain method which return the probability of a given chain of tags.
- If the word it's unseen use the nltk POS tagger.
- The class must perform a pre processing using the methods developed before (Convert them in a class).
- Include a jupyter notebook explaining the functionality of the class.
- Add an example of the processing using the simple twiter database provided.

References

[1] Jurafsky D., Martin J.: Speech and Language Processing 2nd. ed. (2009).

[2] Bird S., Klein E., Loper, E.: Natural Language Processing with Python, (2009). ISBN: 978-0-596-51649-9

[3] Indurkhya N., Damerau F. Handbook of Natural Language Processing, Second Edition (2010). ISBN: 978-1-4200-8593-8

[4] Kao, A., Poteet S. Natural Language Processing and Text Mining (2007). ISBN: 78-1-84628-175-4

[5] https://kavita-ganesan.com/text-preprocessing-tutorial

[6] https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/

[7] https://www.freecodecamp.org/news/a-deep-dive-into-part-of-speech-tagging-using-viterbi-algorithm-17c8de32e8bc/