**1. INTRODUCTION**

In corpus linguistics parts of speech tagging, is also called grammatical tagging or [word-category](https://en.wikipedia.org/wiki/Lexical_category) disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular [part of speech](https://en.wikipedia.org/wiki/Parts_of_speech), based on both its definition, as well as its context—i.e. [relationship with adjacent and related words](https://en.wikipedia.org/wiki/Lexicography) in a [phrase](https://en.wikipedia.org/wiki/Phrase), [sentence](https://en.wikipedia.org/wiki/Sentence_(linguistics)), or [paragraph](https://en.wikipedia.org/wiki/Paragraph). A simplified form of this is commonly taught o school-age children, in the identification of words as nouns, verbs, adjectives, adverbs etc.

**1.1 DOMAIN OF THE PROJECT**

The domain of this project is Artificial Intelligence (AI) and Natural Language Processor (NLP). A brief description of these are given below as follows

**Artificial Intelligence (AI):**

Artificial intelligence (AI) is the [intelligence](https://en.wikipedia.org/wiki/Intelligence) exhibited by machines or software. It is also the name of the academic [field of study](https://en.wikipedia.org/wiki/Field_of_study) which studies how to create computers and computer [software](https://en.wikipedia.org/wiki/Software) that are capable of intelligent behavior. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", in which an [intelligent agent](https://en.wikipedia.org/wiki/Intelligent_agent) is a system that perceives its environment and takes actions that maximize its chances of success. [John McCarthy](https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)), who coined the term in 1955, defines it as "the science and engineering of making intelligent machines".

AI research is highly technical and specialized, and is deeply divided into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues. Some subfields focus on the solution of specific [problems](https://en.wikipedia.org/wiki/Artificial_intelligence#Goals). Others focus on one of several possible [approaches](https://en.wikipedia.org/wiki/Artificial_intelligence#Approaches) or on the use of a particular [tool](https://en.wikipedia.org/wiki/Artificial_intelligence#Tools) or towards the accomplishment of particular [applications](https://en.wikipedia.org/wiki/Artificial_intelligence#Applications).

The central problems of AI research include reasoning, knowledge, planning, learning, natural language processing, perception and the ability to move and manipulate objects. [General intelligence](https://en.wikipedia.org/wiki/Artificial_general_intelligence) is still among the field's long-term goals. Currently popular approaches include [statistical methods](https://en.wikipedia.org/wiki/Artificial_intelligence#Statistical), [computational intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence#Sub-symbolic) and [traditional symbolic AI](https://en.wikipedia.org/wiki/Artificial_intelligence#Symbolic). There are a large number of tools used in AI, including versions of [search and mathematical optimization](https://en.wikipedia.org/wiki/Artificial_intelligence#Search_and_optimization), [logic](https://en.wikipedia.org/wiki/Artificial_intelligence#Logic), [methods based on probability and economics](https://en.wikipedia.org/wiki/Artificial_intelligence#Probabilistic_methods_for_uncertain_reasoning), and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, mathematics, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology.

**Natural Language Processor (NLP):**

Natural language processing (NLP) is a field of [computer science](https://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), and [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) concerned with the interactions between [computers](https://en.wikipedia.org/wiki/Computer) and [human languages](https://en.wikipedia.org/wiki/Natural_language). As such, NLP is related to the area of [human–computer interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction). Many challenges in NLP involve [natural language understanding](https://en.wikipedia.org/wiki/Natural_language_understanding), that is, enabling computers to derive meaning from human or natural language input, and others involve [natural language generation](https://en.wikipedia.org/wiki/Natural_language_generation).

[Natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) gives machines the ability to read and [understand](https://en.wikipedia.org/wiki/Natural_language_understanding) the languages that humans speak. A sufficiently powerful natural language processing system would enable [natural language user interfaces](https://en.wikipedia.org/wiki/Natural_language_user_interface) and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval), [question answering](https://en.wikipedia.org/wiki/Question_answering) and [machine translation](https://en.wikipedia.org/wiki/Machine_translation).

A common method of processing and extracting meaning from natural language is through semantic indexing. Increases in processing speeds and the drop in the cost of data storage makes indexing large volumes of abstractions of the user's input much more efficient.

**1.2 PROBLEM DEFINITION**

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken. This is not rare—in [natural languages](https://en.wikipedia.org/wiki/Natural_language), a large percentage of word-forms are ambiguous. For example, even "dogs", which is usually thought of as just a plural noun, can also be a verb:

*“The sailor dogs the hatch.”*

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

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

**1.3 IMPLEMENTATION**

The implementation of the parts of speech tagging was done using different methods. These methods are given as follows:

* The corpus brown.
* Use of Hidden Markov Models.
* Dynamic programming methods.
* Unsupervised taggers.
* Other taggers and methods.

**The Corpus Brown:**

The [Brown Corpus](https://en.wikipedia.org/wiki/Brown_Corpus) was painstakingly "tagged" with part-of-speech markers over many years. A first approximation was done with a program by Greene and Rubin, which consisted of a huge handmade list of what categories, could co-occur at all. For example, article then noun can occur, but article verb cannot. The program got about 70% correct. Its results were repeatedly reviewed and corrected by hand, and later users sent in errata, so that by the late 70s the tagging was nearly perfect.

This corpus has been used for innumerable studies of word-frequency and of part-of-speech, and inspired the development of similar "tagged" corpora in many other languages

**Use of Hidden Markov Models:**

The researchers in Europe began to use [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_model) (HMMs) to disambiguate parts of speech, when working to tag the [Lancaster-Oslo-Bergen Corpus](https://en.wikipedia.org/wiki/Lancaster-Oslo-Bergen_Corpus)of British English. HMMs involve counting cases (such as from the Brown Corpus), and making a table of the probabilities of certain sequences. For example, once you've seen an article such as 'the', perhaps the next word is a noun 40% of the time, an adjective 40%, and a number 20%. Knowing this, a program can decide that "can" in "the can" is far more likely to be a noun than a verb or a modal.

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

**Dynamic Programming Methods:**

In 1987, [Steven DeRose](https://en.wikipedia.org/wiki/Steven_DeRose) and [Ken Church](https://en.wikipedia.org/w/index.php?title=Kenneth_W._Church&action=edit&redlink=1) independently developed [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) algorithms to solve the same problem in vastly less time. Their methods were similar to the [Viterbi algorithm](https://en.wikipedia.org/wiki/Viterbi_algorithm) known for some time in other fields. DeRose used a table of pairs, while Church used a table of triples and a method of estimating the values for triples that were rare or nonexistent in the Brown Corpus. Both methods achieved accuracy over 95%. DeRose's 1990 dissertation at [Brown University](https://en.wikipedia.org/wiki/Brown_University) included analyses of the specific error types, probabilities, and other related data, and replicated his work for Greek, where it proved similarly effective.

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

**Unsupervised taggers:**

The methods already discussed involve working from a pre-existing corpus to learn tag probabilities. It is, however, also possible to [bootstrap](https://en.wikipedia.org/wiki/Bootstrapping_(linguistics)) using "unsupervised" tagging. Unsupervised tagging techniques use an untagged corpus for their training data and produce the tag set by induction. That is, they observe patterns in word use, and derive part-of-speech categories themselves.

For example, statistics readily reveal that "the", "a", and "an" occur in similar contexts, while "eat" occurs in very different ones. With sufficient iteration, similarity classes of words emerge that is remarkably similar to those human linguists would expect; and the differences themselves sometimes suggest valuable new insights.

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

**Other taggers and methods:**

Some current major algorithms for part-of-speech tagging include the [Viterbi algorithm](https://en.wikipedia.org/wiki/Viterbi_algorithm), [Brill Tagger](https://en.wikipedia.org/wiki/Brill_Tagger), [Constraint Grammar](https://en.wikipedia.org/wiki/Constraint_Grammar), and the [Baum-Welch algorithm](https://en.wikipedia.org/wiki/Baum-Welch_algorithm) (also known as the forward-backward algorithm). [Hidden Markov model](https://en.wikipedia.org/wiki/Hidden_Markov_model) and [visible Markov model](https://en.wikipedia.org/wiki/Markov_model) taggers can both be implemented using the [Viterbi algorithm](https://en.wikipedia.org/wiki/Viterbi_algorithm). The rule-based Brill tagger is unusual in that it learns a set of rule patterns, and then applies those patterns rather than optimizing a statistical quantity. Unlike the Brill tagger where the rules are ordered sequentially, the POS and morphological tagging toolkit [RDRPOSTagger](http://rdrpostagger.sourceforge.net/) stores rules in the form of a [Ripple down Rules](https://en.wikipedia.org/wiki/Ripple_Down_Rules) tree.

Many [machine learning](https://en.wikipedia.org/wiki/Machine_learning) methods have also been applied to the problem of POS tagging. Methods such as [SVM](https://en.wikipedia.org/wiki/Support_vector_machine), [Maximum entropy classifier](https://en.wikipedia.org/wiki/Maximum_entropy_classifier), [Perceptron](https://en.wikipedia.org/wiki/Perceptron), and [Nearest-neighbor](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm) have all been tried, and most can achieve accuracy above 95%..

However, many significant taggers are not included. Thus, it should not be assumed that the results reported there are the best that can be achieved with a given approach; nor even the best that have been achieved with a given approach.

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

**1.4 MOTIVATION**

HMM is a wide area for development, where fascinating things like speech recognition, text-to-speech, speech-to-text conversions and facial recognition and parts-of-speechtagging are being developed. One our team mate came up with this idea of artificially recognizing the parts-of-speech/ the inside meaning of each word of the sentence. Then the whole team gave a thought about it, and then we agreed that it is a good area to do a project.

We can mimic some aspects of this kind of reasoning in an artificial agent. In particular, if the activity exhibits strong underlying structure, we can use this structure to recognize where we are in the activity, how it progressed, and how it might proceed.

Processes with a strong underlying structure invite easy simplification into finite state spaces. But for many natural processes, standard deterministic finite-state machines are not enough. Even if we know the state space, the current state of the process can rarely be known for certain, and will have to be estimated from whatever evidence is available. Moreover, we need to deal with the possibility that the evidence for this estimation will usually be noisy, incomplete, or incorrect.

Noisy processes can be successfully estimated using stochastic techniques. Hidden Markov models (HMMs), in particular, have been successfully used to track finite-state processes based on noisy evidence. For example, in speech recognition HMMs track the production of words as movement through a space of phonemes and sounds, based on very noisy evidence but very concrete transition rules; in robot navigation HMMs model the movement of a robot through physical space based on occasional and potentially erroneous sensor readings, and so on. The same approach can also be used to track processes in games, such as the movement of the player through physical or abstract state spaces.

**1.5 OBJECTIVES OF THE PROJECT**

* To find the parts-of-speech of each word in the sentence of the given input file.
* To maintain accuracy efficiently.
* To give same result consistently i.e. when a same line is given twice it should give the same result as previous.
* To be able to train on the training set and work for the test set.
* To give accurate results consistently.
* Information Retrieval.
* Word-Sense Disambiguation.
* Text-to-Speech.
* To give the parts of speech accurately.

**1.6 LIMITATIONS**

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

“The big green fire truck.”

* A second important example is the [use/mention distinction](https://en.wikipedia.org/wiki/Use/mention_distinction), as in the following example, where "blue" could be replaced by a word from any POS (the Brown Corpus tag set appends the suffix "-NC" in such cases):

“The word “blue” has 4 letters.”

* Words in a language other than that of the "main" text are commonly tagged as "foreign", usually in addition to a tag for the role the foreign word is actually playing in context.
* There are also many cases where POS categories and "words" do not map one to one.

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

Markov models are probabilistic finite automata that are used for many kinds of (sequential) disambiguation tasks such as:

1. Speech recognition

2. Spell checking

3. Part-of-speech tagging

4. Named entity recognition

A (discrete) Markov model runs through a sequence of states emitting signals. If the state sequence cannot be determined from the sequence of emitted signals, the model is said to be hidden.

A Markov model consists of five elements:

1. A finite set of states.
2. A finite signal alphabet.
3. Initial probabilities P(s) defining the probability of starting in state s.
4. Transition probabilities.
5. Emission probabilities.

We use Expectation-Maximization Algorithm (EM). EM is a general class of algorithms for finding the maximum likelihood estimator of parameters in probabilistic models. It is an iterative algorithm where we alternate between calculating the expectation of the log likelihood of the model given the parameters and then finding the parameters that maximizes the expected log likelihood. EM applies roughly the same number of observations for each state.

Part-of-speech-tags, Examples:

PART-OF-SPEECH TAG EXAMPLES

NOUN NN Aircraft, data

NOUN, PLURAL NNS women, books

ADVERB RB Often, particularly

**2.2 EXISTING SYSTEM**

There are many models which are used in determining the syntactic parts-of-speech. But, a correct classification is required. There are different approaches. They are:

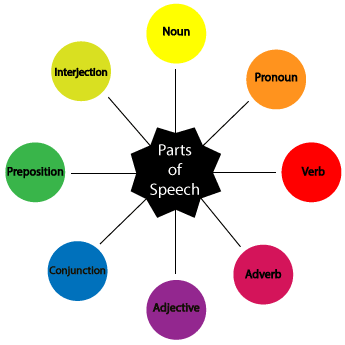
* HMM tagging = The bold approach: 'Use all the information you have and guess’
* Constrain Grammar (CG) tagging = The cautious approach: 'Don't guess, just eliminate the impossible.
* Transmation-based (TB) tagging = The whimsical approach: 'Guess first, then change your mind if necessary!

**Constrain Grammar(CG):** This is was the most frequently used grammar.The Constraint Grammar concept was launched by [Fred Karlsson](https://en.wikipedia.org/wiki/Fred_Karlsson) in 1990 (Karlsson 1990; Karlsson et al., eds, 1995), and CG taggers and parsers have since been written for a large variety of languages, routinely achieving accuracy.

It is a methodological paradigm for [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) (NLP). Linguist-written, context dependent rules are compiled into a grammar that assigns grammatical tags ("readings") to words or other tokens in running text. Typical tags address [lemmatisation](https://en.wikipedia.org/wiki/Lemmatisation) ([lexeme](https://en.wikipedia.org/wiki/Lexeme) or [base form](https://en.wikipedia.org/wiki/Base_form)), [inflexion](https://en.wikipedia.org/wiki/Inflexion), [derivation](https://en.wikipedia.org/wiki/Derivation_%28linguistics%29), [syntactic function](https://en.wikipedia.org/wiki/Syntactic_function), dependency, [valency](https://en.wikipedia.org/wiki/Valency_%28linguistics%29), [case roles](https://en.wikipedia.org/wiki/Case_role), [semantic](https://en.wikipedia.org/wiki/Semantic) type etc. Each rule either adds, removes, selects or replaces a tag or a set of grammatical tags in a given sentence context. Context conditions can be linked to any tag or tag set of any word anywhere in the sentence, either locally (defined distances) or globally (undefined distances). Context conditions in the same rule may be linked, i.e. conditioned upon each other, negated, or blocked by interfering words or tags. Typical CGs consists of thousands of rules, which are applied set-wise in progressive steps, covering ever more advanced levels of analysis.

Earlier, for the assignment of POS tags the following principles were adopted:

* The word-by-word principle: tags were assigned to each token separately. This principle also applied in cases where the token was part of a multi-word expression. Thus ter and zake in the multi-word expression ter zake are tagged as two independent tokens. The same principle was also applied to words that usually only occur as part of a mulit-word (eg nervosa which normally only occurs in combination with anorexia in anorexia nervosa). Compound words were tagged as single words: eg daarlangs received a single tag (adverb) instead of an adverbial pronoun followed by a postposition.
* The form-before-function principle: according to this principle the formal (form) and morphological criteria take precedence over functional or semantic criteria. Thus for example maandag was always tagged as a noun, even in cases where it occurred in adverbial position (as for example in maandag sta ik niet graag op).
* The principle of obligatorily contextual disambiguation: according to this principle words that are potentially ambiguous are assigned a single tag, viz. the tag that is appropriate in the given context.
* All output was manually verified.



**2.1.1 Variable-Order Markov Model(VOM)**

This is an important class of models that extend the well-known markov chain models. In markov models, each random variable in sequence depends on a fixed number of random variables. Whereas, in variable-order models this number of random variables may vary based on the specific observed realization. These models are also called as context trees.

Effectively, for a given training sequence, the VOM models are found to obtain better model parameterization than other models, which leads to a better variance-bias tradeoff of the learned models. Various efficient algorithms have been devised for estimating the parameters of this model. VOM models have been successfully applied to areas such as coding and data compression, document compression and others.

**2.1.2 Variable-Memory Markov Model**

This is another approach to disambiguating syntactically ambiguous words. This model dynamically adapts their history length based on the training data, and hence it uses fewer parameters. The VMM is an approximation of an unlimited order markov source. It can incorporate both the static and dynamic information systematically, while keeping the ability to change the model due to future observations.

The VMM algorithm is based on minimizing the statistical prediction error of a Markov model, measured by various symbols. The memory is extended significantly. This approach is easy to implement, the learning algorithm and classification of new tags are computationally efficient, and the results achieved are encouraging.

**2.1.3 Variable-Order Bayesian Network**

This model provides an important extension of both the Bayesian network models and the variable-order markov models. Generally, each random variable depends on a subset of random variables. In this model, these subsets may vary based on the specific realization of observed variables. The observed realizations are often called the context and hence, VOBN models are also known as context-specific Bayesian networks.

This method is very flexible and is an advantage in classification and analysis applications.

**2.1.4 Higher Order Markov Model**

In this we can build more “memory” into our states by using a higher order markov model. In an nth order markov models.

P(|,,….)=P(|….).

The higher the order the less reliable is the parameters. Higher order models remember more “history”. Additional history can have predictive value.

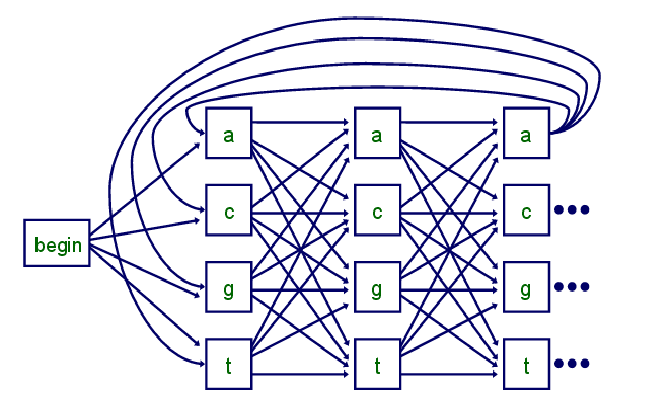
Example: –predict the next word in this sentence fragment

“... the\_\_(duck, end, grain, tide, wall, ...?)

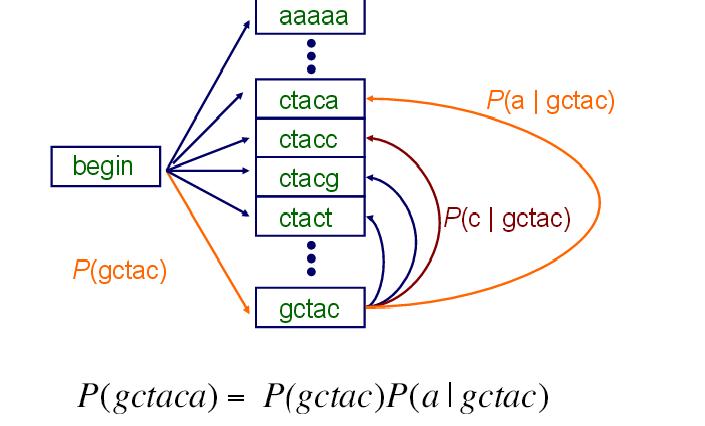
Another example is Inhomogeneous Markov model, where we can have different distributions at different positions in the sequence.

Generally, in the Markov chain models we have considered so far, the probabilities do not depend on our position in a given sequence.

In an Inhomogeneous Markov model, we can have different distributions at different positions in the sequence.



The above diagram shows a inhomogenous markov chain.



The above diagram is a fifth order markov chain.

**2.3 PROPOSED SYSTEM**

A Hidden Markov model is a statical model in which the system is being modeled is assumed to be a markov process with hidden states. Taking each word in the sentence as a state in the hidden-markov model we try to find the most accurate POS of the word using probabilities.

HMM is a sequential model where we use the markov chain rule for the layers of the system. HMM uses hidden layers in between the sequence of the layers to get the accurate layers.

**2.2.1 Markov Process**

A Markov process or Markov process, named after the Russian mathematician Andrey Markov, is a stochastic process that satisfies the Markov property. A Markov process can be thought of as 'memory less': loosely speaking, a process satisfies the Markov property if one can make predictions for the future of the process based solely on its present state just as well as one could knowing the process's full history. i.e., [conditional](https://en.wikipedia.org/wiki/Conditional_probability) on the present state of the system, its future and past are [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)). The following diagram is an example of a markov process:



A Markov process is a stochastic process with the following properties:

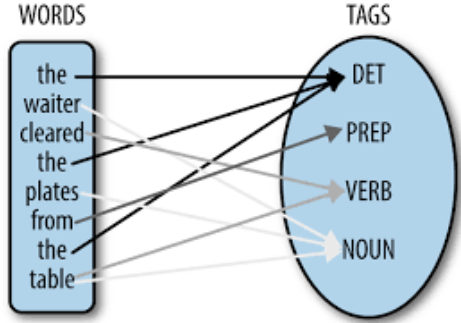
1. The number of possible outcomes or states is finite.
2. The outcome at any stage depends only on the outcome of the previous stage.
3. The probabilities are constant over time.

If x0 is a vector which represents the initial state of a system, then there is a matrix M such that the state of the system after one iteration is given by the vector Mx0. Thus we get a chain of state vectors: x0, Mx0, M2x0, . . . where the state of the system after n iterations is given by Mnx0. Such a chain is called a Markov chain and the matrix M is called a transition matrix. The state vectors can be of one of two types: an absolute vector or a probability vector. An absolute vector is a vector whose entries give the actual number of objects in a give state, as in the first example. A probability vector is a vector where the entries give the percentage (or probability) of objects in a given state. We will take all of our state vectors to be probability vectors from now on. Note that the entries of a probability vector add up to 1.

A stochastic process is a sequence of events in which the outcome at any stage depends on some probability.

**Theorem In Markov Chain Theory:**

* If T is a regular transition matrix, then as n approaches infinity, Tn→S where S is a matrix of the form [v, v,…,v] with v being a constant vector.
* If T is a regular transition matrix of a Markov chain process, and if X is any state vector, then as n approaches infinity, TnX→p, where p is a fixed probability vector (the sum of its entries is 1), all of whose entries are positive.



**2.2.2 Hidden Markov Model**

A hidden Markov model (HMM) is a [statistical](https://en.wikipedia.org/wiki/Statistical_model) [Markov model](https://en.wikipedia.org/wiki/Markov_model) in which the system being modeled is assumed to be a [Markov process](https://en.wikipedia.org/wiki/Markov_process) with unobserved (hidden) states. A HMM can be presented as the simplest [dynamic Bayesian network](https://en.wikipedia.org/wiki/Dynamic_Bayesian_network). The mathematics behind the HMM were developed by [L. E. Baum](https://en.wikipedia.org/wiki/Leonard_E._Baum)and coworkers. It is closely related to an earlier work on the optimal nonlinear [filtering problem](https://en.wikipedia.org/wiki/Filtering_problem_(stochastic_processes)) by [Ruslan L. Stratonovich](https://en.wikipedia.org/wiki/Ruslan_L._Stratonovich), who was the first to describe the [forward-backward procedure](https://en.wikipedia.org/wiki/Forward%E2%80%93backward_algorithm).

In simpler [Markov models](https://en.wikipedia.org/wiki/Markov_model) (like a [Markov chain](https://en.wikipedia.org/wiki/Markov_chain)), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

Hidden Markov models are especially known for their application in [temporal](https://en.wikipedia.org/wiki/Time) pattern recognition such as [speech](https://en.wikipedia.org/wiki/Speech_recognition), [handwriting](https://en.wikipedia.org/wiki/Handwriting_recognition), [gesture recognition](https://en.wikipedia.org/wiki/Gesture_recognition), [part-of-speech tagging](https://en.wikipedia.org/wiki/Part-of-speech_tagging), musical score following, [partial discharges](https://en.wikipedia.org/wiki/Partial_discharge) and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics).

A hidden Markov model can be considered a generalization of a [mixture model](https://en.wikipedia.org/wiki/Mixture_model) where the hidden variables (or [latent variables](https://en.wikipedia.org/wiki/Latent_variables)), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other. Recently, hidden Markov models have been generalized to pairwise Markov models and triplet Markov models which allow consideration of more complex data structures and the modeling of non- stationary data.

**Mathematical Description:**

A basic, Non-Bayesian hidden Markov model can be described as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Description: N |  | Description: = |  | number of states |
|  |  | Description: T |  |  | - | number of observations |
|  |  | Description: \theta_{i=1 \dots N} |  | -Description: = |  | emission parameter for an observation associated with state Description: i |
|  |  | Description: \phi_{i=1 \dots N, j=1 \dots N} |  | Description: = |  | probability of transition from state Description: i to state Description: j |
|  |  | Description: \boldsymbol\phi_{i=1 \dots N} |  | Description: = |  | Description: N-dimensional vector, composed of Description: \phi_{i,1 \dots N}; must sum to Description: 1, the row of the matrix Description: \phi_{i=1 \dots N, j=1 \dots N} |
|  |  | Description: x_{t=1 \dots T} |  | Description: = |  | (hidden) state at time Description: t |
|  |  | Description: y_{t=1 \dots T} |  | Description: = |  | observation at time Description: t |
|  |  | Description: F(y|\theta) |  | Description: = |  | probability distribution of an observation, parametrized on Description: \theta |
|  |  | Description: x_{t=2 \dots T} |  | Description: \sim |  | Description: \operatorname{Categorical}(\boldsymbol\phi_{x_{t-1}}) |
|  |  | Description: y_{t=1 \dots T} |  | Description: \sim |  | Description: F(\theta_{x_t}) |

In a Bayesian setting, all parameters are associated with random variables, as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Description: N,T |  | Description: = |  | as above |
|  |  | Description: \theta_{i=1 \dots N}, \phi_{i=1 \dots N, j=1 \dots N}, \boldsymbol\phi_{i=1 \dots N} |  | Description: = |  | as above |
|  |  | Description: x_{t=1 \dots T}, y_{t=1 \dots T}, F(y|\theta) |  | Description: = |  | as above |
|  |  | Description: \alpha |  | Description: = |  | shared hyperparameter for emission parameters |
|  |  | Description: \beta |  | Description: = |  | shared hyperparameter for transition parameters |
|  |  | Description: H(\theta|\alpha) |  | Description: = |  | prior probability distribution of emission parameters, parametrized on Description: \alpha |
|  |  | Description: \theta_{i=1 \dots N} |  | Description: \sim |  | Description: H(\alpha) |
|  |  | Description: \boldsymbol\phi_{i=1 \dots N} |  | Description: \sim |  | Description: \operatorname{Symmetric-Dirichlet}_N(\beta) |
|  |  | Description: x_{t=2 \dots T} |  | Description: \sim |  | Description: \operatorname{Categorical}(\boldsymbol\phi_{x_{t-1}}) |
|  |  | Description: y_{t=1 \dots T} |  | Description: \sim |  | Description: F(\theta_{x_t}) |

These characterizations use Description: F and Description: H to describe arbitrary distributions over observations and parameters, respectively. Typically Description: H will be the [conjugate prior](https://en.wikipedia.org/wiki/Conjugate_prior) of Description: F. The two most common choices of Description: F are [Gaussian](https://en.wikipedia.org/wiki/Gaussian_distribution) and [categorica](https://en.wikipedia.org/wiki/Categorical_distribution)l.

**Applications Of Hmm:**

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

* [Single Molecule Kinetic analysis](https://en.wikipedia.org/wiki/Single-molecule_experiment)
* [Cryptanalysis](https://en.wikipedia.org/wiki/Cryptanalysis)
* [Speech recognition](https://en.wikipedia.org/wiki/Speech_recognition)
* [Speech synthesis](https://en.wikipedia.org/wiki/Speech_synthesis)
* [Part-of-speech tagging](https://en.wikipedia.org/wiki/Part-of-speech_tagging)
* Document Separation in scanning solutions
* [Machine translation](https://en.wikipedia.org/wiki/Machine_translation)
* [Partial discharge](https://en.wikipedia.org/wiki/Partial_discharge)
* [Gene prediction](https://en.wikipedia.org/wiki/Gene_prediction)
* [Alignment of bio-sequences](https://en.wikipedia.org/wiki/Sequence_alignment)
* [Time Series Analysis](https://en.wikipedia.org/wiki/Time_series)
* [Activity recognition](https://en.wikipedia.org/wiki/Activity_recognition)
* [Protein folding](https://en.wikipedia.org/wiki/Protein_folding)
* Metamorphic Virus Detection
* DNA Motif Discovery

**2.2.3 Algorithms**

**2.2.3.1 The Viterbi Algorithm**

The Viterbi algorithm is a [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the most [likely](https://en.wikipedia.org/wiki/Likelihood_function) sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of [Markov information sources](https://en.wikipedia.org/wiki/Markov_information_source) and [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_model).

The algorithm has found universal application in decoding the [convolutional codes](https://en.wikipedia.org/wiki/Convolutional_code) used in both [CDMA](https://en.wikipedia.org/wiki/CDMA) and [GSM](https://en.wikipedia.org/wiki/GSM) digital cellular, [dial-up](https://en.wikipedia.org/wiki/Dial-up) modems, satellite, deep-space communications, and [802.11](https://en.wikipedia.org/wiki/802.11) wireless LANs. It is now also commonly used in [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [speech synthesis](https://en.wikipedia.org/wiki/Speech_synthesis), [diarization](https://en.wikipedia.org/wiki/Diarization" \o "Diarization), [keyword spotting](https://en.wikipedia.org/wiki/Keyword_spotting), [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics), and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics). For example, in [speech-to-text](https://en.wikipedia.org/wiki/Speech-to-text) (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic

The Viterbi algorithm is named after [Andrew Viterbi](https://en.wikipedia.org/wiki/Andrew_Viterbi), who proposed it in 1967 as a decoding algorithm for [convolutional codes](https://en.wikipedia.org/wiki/Convolution_code) over noisy digital communication links.[[2]](https://en.wikipedia.org/wiki/Viterbi_algorithm#cite_note-2) It has, however, a history of [multiple invention](https://en.wikipedia.org/wiki/Multiple_invention), with at least seven independent discoveries, including those by Viterbi, [Needleman and Wunsch](https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm), and [Wagner and Fischer](https://en.wikipedia.org/wiki/Wagner%E2%80%93Fischer_algorithm).

"Viterbi (path, algorithm)" has become a standard term for the application of dynamic programming algorithms to maximization problems involving probabilities.

**2.2.3.2 The Forward-Backward Algorithm**

The forward–backward algorithm is an [inference](https://en.wikipedia.org/wiki/Inference) [algorithm](https://en.wikipedia.org/wiki/Algorithm) for [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_models) which computes the [posterior](https://en.wikipedia.org/wiki/Posterior_probability) [marginal](https://en.wikipedia.org/wiki/Marginal_probability) of all hidden state variables given a sequence of observations/emissions Description: o_{1:t}:= o_1,\dots,o_t, i.e. it computes, for all hidden state variables Description: X_k \in \{X_1, \dots, X_t\}, the distribution Description: P(X_k\ |\ o_{1:t}).

This inference task is usually called smoothing. The algorithm makes use of the principle of [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) to compute efficiently the values that are required to obtain the posterior marginal distributions in two passes. The first pass goes forward in time while the second goes backward in time; hence the name forward–backward algorithm.

The term forward–backward algorithm is also used to refer to any algorithm belonging to the general class of algorithms that operate on sequence models in a forward–backward manner. In this sense, the descriptions in the remainder of this article refer but to one specific instance of this class.

The algorithm involves three steps:

1. Computing forward probabilities
2. Computing backward probabilities
3. Computing smoothed values.

The forward and backward steps may also be called "forward message pass" and "backward message pass" - these terms are due to the message-passing used in general [belief propagation](https://en.wikipedia.org/wiki/Belief_propagation) approaches. At each single observation in the sequence, probabilities to be used for calculations at the next observation are computed. The smoothing step can be calculated simultaneously during the backward pass. This step allows the algorithm to take into account any past observations of output for computing more accurate results.

The forward–backward algorithm can be used to find the most likely state for any point in time. It cannot, however, be used to find the most likely sequence of states.

**2.2.3.3 Expectation-Maximization**

In [statistics](https://en.wikipedia.org/wiki/Statistics), an expectation–maximization (EM) algorithm is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) for finding [maximum likelihood](https://en.wikipedia.org/wiki/Maximum_likelihood) or [maximum a posteriori](https://en.wikipedia.org/wiki/Maximum_a_posteriori) (MAP) estimates of [parameters](https://en.wikipedia.org/wiki/Parameter) in [statistical models](https://en.wikipedia.org/wiki/Statistical_model), where the model depends on unobserved [latent variables](https://en.wikipedia.org/wiki/Latent_variable). The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the [log-likelihood](https://en.wikipedia.org/wiki/Likelihood_function#Log-likelihood) evaluated using the current estimate for the parameters, and maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

The EM algorithm is used to find (locally) [maximum likelihood](https://en.wikipedia.org/wiki/Maximum_likelihood) parameters of a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) in cases where the equations cannot be solved directly. Typically these models involve [latent variables](https://en.wikipedia.org/wiki/Latent_variable) in addition to unknown [parameters](https://en.wikipedia.org/wiki/Parameters) and known data observations. That is, either there are [missing values](https://en.wikipedia.org/wiki/Missing_values) among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points. For example, a [mixture model](https://en.wikipedia.org/wiki/Mixture_model) can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.

Finding a maximum likelihood solution typically requires taking the derivatives of the [likelihood function](https://en.wikipedia.org/wiki/Likelihood_function) with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. In statistical models with latent variables, this usually is not possible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.

**Applications:**

EM is frequently used for [data clustering](https://en.wikipedia.org/wiki/Data_clustering) in [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision). In [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), two prominent instances of the algorithm are the [Baum-Welch algorithm](https://en.wikipedia.org/wiki/Baum-Welch_algorithm) and the [inside-outside algorithm](https://en.wikipedia.org/wiki/Inside-outside_algorithm) for unsupervised induction of [probabilistic context-free grammars](https://en.wikipedia.org/wiki/Probabilistic_context-free_grammar)

**2.2.3.4 Baum-welch algorithm**

In electrical engineering, computer science, statistical computing and bioinformatics, the Baum–Welch algorithm is used to find the unknown parameters of a [hidden Markov model](https://en.wikipedia.org/wiki/Hidden_Markov_model) (HMM). It makes use of the [forward-backward algorithm](https://en.wikipedia.org/wiki/Forward-backward_algorithm) and is named for [Leonard E. Baum](https://en.wikipedia.org/wiki/Leonard_E._Baum) and [Lloyd R. Welch](https://en.wikipedia.org/wiki/Lloyd_R._Welch).

A [Hidden Markov Model](https://en.wikipedia.org/wiki/Hidden_Markov_Model) describes the joint probability of a collection of 'hidden' and observed discrete random variables. It relies on the assumption that the Description: i^{th} hidden variable given the Description: (i-1)^{th} hidden variable is independent of previous hidden variables, and the current observation variables depend only on the current hidden state.

The Baum–Welch algorithm uses the well-known [EM algorithm](https://en.wikipedia.org/wiki/EM_algorithm) to find the [maximum likelihood](https://en.wikipedia.org/wiki/Maximum_likelihood) estimate of the parameters of a hidden Markov model given a set of observed feature vectors.

**Applications:**

* Speech recognition

### Cryptanalysis

### Applications in bio-informatics

**3.ANALYSIS**

**3.1 SOFTWARE REQUIREMENTS**

The first step in developing anything is to state the requirements. Being vague about your objective only postpone decision to a later state where changes are much more costly. The minimum software requirements specifications for developing this project are as follows:

* Java language.
* Stanford POS tagger (for training data).
* Eclipse IDE.
* Operating System.
* HMM (hidden markov model).
* Markov chains.
* Probability.
* Transition probability.
* Conditional probability.
* Bayes theorem.

**3.1.1 Java Language**

Java is a general-purpose [computer programming language](https://en.wikipedia.org/wiki/Programming_language) that is [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "[write once, run anywhere](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)" (WORA), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to [byte code](https://en.wikipedia.org/wiki/Java_bytecode) that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture). As of 2016, Java is one of the most [popular programming languages in use](https://en.wikipedia.org/wiki/Measuring_programming_language_popularity), particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by [James Gosling](https://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) (which has since been [acquired by Oracle Corporation](https://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle)) and released in 1995 as a core component of Sun Microsystems' [Java platform](https://en.wikipedia.org/wiki/Java_(software_platform)). The language derives much of its [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)) from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but it has fewer [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them.

The original and [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) Java [compilers](https://en.wikipedia.org/wiki/Compiler), virtual machines, and [class libraries](https://en.wikipedia.org/wiki/Library_(computing)) were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process), Sun relicensed most of its Java technologies under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License). Others have also developed alternative implementations of these Sun technologies, such as the [GNU Compiler for Java](https://en.wikipedia.org/wiki/GNU_Compiler_for_Java) (byte code compiler), [GNU Class path](https://en.wikipedia.org/wiki/GNU_Classpath) (standard libraries), and [Iced Tea](https://en.wikipedia.org/wiki/IcedTea)-Web (browser plugin for applets).

The latest version is Java 8, which is the only version currently supported for free by Oracle, although earlier versions are supported both by Oracle and other companies on a commercial basis.

**Principles**

There were five primary goals in the creation of the Java language:

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded, and dynamic".

**Versions**

Main article: [Java version history](https://en.wikipedia.org/wiki/Java_version_history)

As of 2015, only Java 8 is supported ("publicly"). Major release versions of Java, along with their release dates:

* JDK 1.0 (January 21, 1996)
* JDK 1.1 (February 19, 1997)
* J2SE 1.2 (December 8, 1998)
* J2SE 1.3 (May 8, 2000)
* J2SE 1.4 (February 6, 2002)
* J2SE 5.0 (September 30, 2004)
* Java SE 6 (December 11, 2006)
* Java SE 7 (July 28, 2011)
* Java SE 8 (March 18, 2014)

**Practices**

**Java platform**

Main articles: [Java (software platform)](https://en.wikipedia.org/wiki/Java_(software_platform)) and [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine)

[](https://en.wikipedia.org/wiki/File:Java_Control_Panel.JPG)

Java Control Panel, version 7

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called [Java bytecode](https://en.wikipedia.org/wiki/Java_bytecode), instead of directly to architecture-specific [machine code](https://en.wikipedia.org/wiki/Machine_code). Java bytecode instructions are analogous to machine code, but they are intended to be executed by a [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) (VM) written specifically for the host hardware. [End users](https://en.wikipedia.org/wiki/End_user) commonly use a [Java Runtime Environment](https://en.wikipedia.org/wiki/Java_virtual_machine) (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java [applets](https://en.wikipedia.org/wiki/Applet).

Standard libraries provide a generic way to access host-specific features such as graphics, [threading](https://en.wikipedia.org/wiki/Thread_(computing)), and [networking](https://en.wikipedia.org/wiki/Computer_network).

The use of universal byte code makes porting simple. However, the overhead of interpreting byte code into machine instructions makes interpreted programs almost always run more slowly than native [executables](https://en.wikipedia.org/wiki/Executable" \o "Executable). However, [just-in-time](https://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) compilers that compile byte codes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) for it, which translates the [Java byte code](https://en.wikipedia.org/wiki/Java_bytecode) into the platform's machine language.

**Implementations**

[Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) is the current owner of the official implementation of the Java SE platform, following their acquisition of [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) (still works for XP, while only later versions currently "publicly" supported), [Mac OS X](https://en.wikipedia.org/wiki/OS_X), [Linux](https://en.wikipedia.org/wiki/Linux) and [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)). Because Java lacks any formal standardization recognized by [Ecma International](https://en.wikipedia.org/wiki/Ecma_International" \o "Ecma International), ISO/IEC, ANSI, or other third-party standards organization, the Oracle implementation is the [de facto standard](https://en.wikipedia.org/wiki/De_facto_standard).

The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the [Java Development Kit](https://en.wikipedia.org/wiki/Java_Development_Kit) (JDK), which is intended for software developers and includes development tools such as the [Java compiler](https://en.wikipedia.org/wiki/Java_compiler), [Javadoc](https://en.wikipedia.org/wiki/Javadoc" \o "Javadoc), [Jar](https://en.wikipedia.org/wiki/JAR_(file_format)), and a [debugger](https://en.wikipedia.org/wiki/Debugger).

[OpenJDK](https://en.wikipedia.org/wiki/OpenJDK) is another notable Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with [Microsoft](https://en.wikipedia.org/wiki/Microsoft) after Sun claimed that the Microsoft implementation did not support [RMI](https://en.wikipedia.org/wiki/Java_remote_method_invocation) or [JNI](https://en.wikipedia.org/wiki/Java_Native_Interface) and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US$20 million, as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Java with [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows).

Platform-independent Java is essential to [Java EE](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition), and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

**Performance**

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++. However, Java programs' execution speed improved significantly with the introduction of [just-in-time compilation](https://en.wikipedia.org/wiki/Just-in-time_compilation) in 1997/1998 for [Java 1.1](https://en.wikipedia.org/wiki/Java_version_history), the addition of language features supporting better code analysis (such as inner classes, the StringBuilder class, optional assertions, etc.), and optimizations in the Java virtual machine, such as [HotSpot](https://en.wikipedia.org/wiki/HotSpot" \o "HotSpot) becoming the default for Sun's JVM in 2000. With Java 1.5, the performance was improved with the addition of the java. util. concurrent package, including [Lock free](https://en.wikipedia.org/wiki/Lock_free) implementations of the [Concurrent Maps](https://en.wikipedia.org/wiki/Java_ConcurrentMap) and other multi-core collections, and it was improved further Java 1.6.

Some platforms offer direct hardware support for Java; there are microcontrollers that can run Java in hardware instead of a software Java virtual machine, and [ARM](https://en.wikipedia.org/wiki/ARM_architecture) based processors can have hardware support for executing Java bytecode through their [Jazelle](https://en.wikipedia.org/wiki/Jazelle" \o "Jazelle) option (while its support is mostly dropped in current implementations of ARM).

**Automatic Memory Management**

Java uses an [automatic garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) to manage memory in the [object lifecycle](https://en.wikipedia.org/wiki/Object_lifetime). The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the [unreachable memory](https://en.wikipedia.org/wiki/Unreachable_memory) becomes eligible to be freed automatically by the garbage collector. Something similar to a [memory leak](https://en.wikipedia.org/wiki/Memory_leak) may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the [stack](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)), or explicitly allocated and deallocated from the [heap](https://en.wikipedia.org/wiki/Memory_management#DYNAMIC). In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a [memory leak](https://en.wikipedia.org/wiki/Memory_leak) occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of [smart pointers](https://en.wikipedia.org/wiki/Smart_pointer), but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e., those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style [pointer arithmetic](https://en.wikipedia.org/wiki/Pointer_(computer_programming)), where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's [primitive data types](https://en.wikipedia.org/wiki/Primitive_data_type) are either stored directly in fields (for objects) or on the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation) (for methods) rather than on the heap, as is commonly true for non-primitive data types (but see [escape analysis](https://en.wikipedia.org/wiki/Escape_analysis)). This was a conscious decision by Java's designers for performance reasons.

Java contains multiple types of garbage collectors. By default, HotSpot uses the [parallel scavenge garbage collector](https://en.wikipedia.org/w/index.php?title=Parallel_scavenge_garbage_collector&action=edit&redlink=1). However, there are also several other garbage collectors that can be used to manage the heap. For 90% of applications in Java, the [Concurrent Mark-Sweep](https://en.wikipedia.org/wiki/Concurrent_mark_sweep_collector) (CMS) garbage collector is sufficient. Oracle aims to replace CMS with the [Garbage-First collector](https://en.wikipedia.org/w/index.php?title=Garbage-First_collector&action=edit&redlink=1) (G1).

**Syntax**

The syntax of Java is largely influenced by [C++](https://en.wikipedia.org/wiki/C%2B%2B). Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language. All code is written inside classes, and every data item is an object, with the exception of the primitive data types, i.e. integers, floating-point numbers, [boolean values](https://en.wikipedia.org/wiki/Boolean_data_type" \o "Boolean data type), and characters, which are not objects for performance reasons. Java reuses some popular aspects of C++ (such as printf() method).

Unlike C++, Java does not support [operator overloading](https://en.wikipedia.org/wiki/Operator_overloading) or [multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) for classes, though multiple inheritance is supported for [interfaces](https://en.wikipedia.org/wiki/Interface_(Java)). This simplifies the language and aids in preventing potential errors and [anti-pattern](https://en.wikipedia.org/wiki/Anti-pattern) design.

Java uses comments similar to those of C++. There are three different styles of comments: a single line style marked with two slashes (//), a multiple line style opened with /\*and closed with \*/, and the [Javadoc](https://en.wikipedia.org/wiki/Javadoc" \o "Javadoc) commenting style opened with /\*\* and closed with \*/. The Javadoc style of commenting allows the user to run the Javadoc executable to create documentation for the program.

**Example:**

// This is an example of a single line comment using two slashes

/\* This is an example of a multiple line comment using the slash and asterisk.

This type of comment can be used to hold a lot of information or deactivate

code, but it is very important to remember to close the comment. \*/

package fibsandlies;

import java.util.HashMap;

/\*\*

\* This is an example of a Javadoc comment; Javadoc can compile documentation

\* from this text. Javadoc comments must immediately precede the class, method, or field being documented.

\*/

public class FibCalculator extends Fibonacci implements Calculator {

private static Map<Integer, Integer> memoized = new HashMap<Integer, Integer>();

/\*

\* The main method written as follows is used by the JVM as a starting point for the program.

\*/

public static void main(String[] args) {

memoized.put(1, 1);

memoized.put(2, 1);

System.out.println(fibonacci(12)); //Get the 12th Fibonacci number and print to console

}

/\*\*

\* An example of a method written in Java, wrapped in a class.

\* Given a non-negative number FIBINDEX, returns

\* the Nth Fibonacci number, where N equals FIBINDEX.

\* @param fibIndex The index of the Fibonacci number

\* @return The Fibonacci number

\*/

public static int fibonacci(int fibIndex) {

if (memoized.containsKey(fibIndex)) {

return memoized.get(fibIndex);

} else {

int answer = fibonacci(fibIndex - 1) + fibonacci(fibIndex - 2);

memoized.put(fibIndex, answer);

return answer;

}

}

}

**3.1.2 POS Tagger**

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers described in these papers (if citing just one paper, cite the 2003 one):

 Kristina Toutanova and Christopher D. Manning. 2000. [Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger](http://nlp.stanford.edu/~manning/papers/emnlp2000.pdf). In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

 Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. [Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network](http://nlp.stanford.edu/~manning/papers/tagging.pdf). In Proceedings of HLT-NAACL 2003, pp. 252-259.

The tagger was originally written by Kristina Toutanova. Since that time, Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, Michel Galley, and John Bauer have improved its speed, performance, usability, and support for other languages.

The system requires Java 1.8+ to be installed. Depending on whether you're running 32 or 64 bit Java and the complexity of the tagger model, you'll need somewhere between 60 and 200 MB of memory to run a trained tagger (i.e., you may need to give java an option like java -mx200m). Plenty of memory is needed to train a tagger. It again depends on the complexity of the model but at least 1GB is usually needed, often more.

Several downloads are available. The basic download contains two trained tagger models for English. The full download contains three trained English tagger models, an Arabic tagger model, a Chinese tagger model, a French tagger model, and a German tagger model. Both versions include the same source and other required files. The tagger can be retrained on any language, given POS-annotated training text for the language.

Part-of-speech name abbreviations: The English taggers use the Penn Treebank tag set. Here are some links to documentation of the Penn Treebank English POS tag set: [1993Computational Linguistics article in PDF](http://acl.ldc.upenn.edu/J/J93/J93-2004.pdf), [AMALGAM page](http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html), [Aoife Cahill's list](http://www.computing.dcu.ie/~acahill/tagset.html). See the included README-Models.txt in the models directory for more information about the tagsets for the other languages.

The tagger is licensed under the [GNU General Public License](http://www.gnu.org/licenses/gpl-2.0.html) (v2 or later). Source is included. The package includes components for command-line invocation, running as a server, and a Java API. The tagger code is dual licensed (in a similar manner to MySQL, etc.). Open source licensing is under the full GPL, which allows many free uses. For distributors of [proprietary software](http://www.gnu.org/licenses/gpl-faq.html#GPLInProprietarySystem), [commercial licensing](http://otlportal.stanford.edu/techfinder/technology/ID=26062) is available. If you don't need a commercial license, but would like to support maintenance of these tools, we welcome gift funding.

**3.1.3 Eclipse IDE:**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), Eclipse is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE). It contains a base [workspace](https://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](https://en.wikipedia.org/wiki/Plug-in_(computing)) system for customizing the environment. Eclipse is written mostly in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and its primary use is for developing Java applications, but it may also be used to develop applications in other [programming languages](https://en.wikipedia.org/wiki/Programming_language) through the use of plugins, including : [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [ABAP](https://en.wikipedia.org/wiki/ABAP), [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [COBOL](https://en.wikipedia.org/wiki/COBOL), [Fortran](https://en.wikipedia.org/wiki/Fortran), [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)), [Lasso](https://en.wikipedia.org/wiki/Lasso_(programming_language)),  [Lua](https://en.wikipedia.org/wiki/Lua_(programming_language)),  [NATURAL](https://en.wikipedia.org/w/index.php?title=NATURAL&action=edit&redlink=1),  [Perl](https://en.wikipedia.org/wiki/Perl" \o "Perl),[PHP](https://en.wikipedia.org/wiki/PHP),  [Prolog](https://en.wikipedia.org/wiki/Prolog),  [Python](https://en.wikipedia.org/wiki/Python_(programming_language)),  [R](https://en.wikipedia.org/wiki/R_(programming_language)),  [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language))  (including [Ruby on Rails](https://en.wikipedia.org/wiki/Ruby_on_Rails) framework) ,  [Rust](https://en.wikipedia.org/wiki/Rust_(programming_language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)" \o "Scala (programming language)), [Clojure](https://en.wikipedia.org/wiki/Clojure" \o "Clojure),  [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)),  [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)), and [Erlang](https://en.wikipedia.org/wiki/Erlang_(programming_language)" \o "Erlang (programming language)). It can also be used to develop packages for the software [Mathematics](https://en.wikipedia.org/wiki/Mathematica). Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial [codebase](https://en.wikipedia.org/wiki/Codebase) originated from [IBM VisualAge](https://en.wikipedia.org/wiki/IBM_VisualAge). The Eclipse [software development kit](https://en.wikipedia.org/wiki/Software_development_kit) (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the [Eclipse Public License](https://en.wikipedia.org/wiki/Eclipse_Public_License), Eclipse [SDK](https://en.wikipedia.org/wiki/Software_development_kit) is [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) (although it is incompatible with the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License)). It was one of the first IDEs to run under [GNU Classpath](https://en.wikipedia.org/wiki/GNU_Classpath) and it runs without problems under [IcedTea](https://en.wikipedia.org/wiki/IcedTea" \o "IcedTea).

**Architecture:**

Eclipse uses plug-ins to provide all the functionality within and on top of the runtime system. Its runtime system is based on [Equinox](https://en.wikipedia.org/wiki/Equinox_(OSGi)), an implementation of the [OSGi](https://en.wikipedia.org/wiki/OSGi" \o "OSGi) core framework specification.

In addition to allowing the Eclipse Platform to be extended using other [programming languages](https://en.wikipedia.org/wiki/Programming_language), such as [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), the plug-in framework allows the Eclipse Platform to work with typesetting languages like [LaTeX](https://en.wikipedia.org/wiki/LaTeX" \o "LaTeX) and networking applications such as [telnet](https://en.wikipedia.org/wiki/Telnet) and [database management systems](https://en.wikipedia.org/wiki/Database_management_system). The plug-in architecture supports writing any desired extension to the environment, such as for [configuration management](https://en.wikipedia.org/wiki/Configuration_management). Java and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) support is provided in the Eclipse [SDK](https://en.wikipedia.org/wiki/Software_development_kit), with support for other [version control systems](https://en.wikipedia.org/wiki/Version_control_system) provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include for [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language), for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in [incremental](https://en.wikipedia.org/wiki/Incremental_compiler) Java compiler and a full model of the Java source files. This allows for advanced [refactoring](https://en.wikipedia.org/wiki/Refactor) techniques and code analysis. The IDE also makes use of a workspace, in this case a set of [metadata](https://en.wikipedia.org/wiki/Metadata) over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements the [graphical control elements](https://en.wikipedia.org/wiki/Graphical_control_element) of the Java toolkit called [SWT](https://en.wikipedia.org/wiki/Standard_Widget_Toolkit), whereas most Java applications use the Java standard [Abstract Window Toolkit](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) or [Swing](https://en.wikipedia.org/wiki/Swing_(Java)). Eclipse's user interface also uses an intermediate [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) layer called [JFace](https://en.wikipedia.org/wiki/JFace" \o "JFace), which simplifies the construction of applications based on SWT. Eclipse was made to run on [Wayland](https://en.wikipedia.org/wiki/Wayland_(display_server_protocol)) during a [GSoC](https://en.wikipedia.org/wiki/Google_Summer_of_Code" \o "Google Summer of Code)-Project in 2014.

Language packs being developed by the "Babel project" provide translations into over a dozen [natural languages](https://en.wikipedia.org/wiki/Natural_language).

**Rich Client Platform:**

Eclipse provides the [Rich Client Platform](https://en.wikipedia.org/wiki/Rich_Client_Platform) (RCP) for developing general purpose applications. The following components constitute the rich client platform:

* [Equinox OSGi](https://en.wikipedia.org/wiki/Equinox_(OSGi)) – a standard bundling framework
* Core platform – boot Eclipse, run [plug-ins](https://en.wikipedia.org/wiki/Plug-in_(computing))
* [Standard Widget Toolkit](https://en.wikipedia.org/wiki/Standard_Widget_Toolkit) (SWT) – a portable [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit)
* [JFace](https://en.wikipedia.org/wiki/JFace) – viewer classes to bring [model view controller](https://en.wikipedia.org/wiki/Model_view_controller) programming to SWT, file buffers, text handling, text editors
* Eclipse Workbench – views, editors, perspectives, wizards

Examples of rich client applications based on Eclipse are:

* [IBM Notes](https://en.wikipedia.org/wiki/IBM_Notes) 8 and 9
* Novell/[NetIQ](https://en.wikipedia.org/wiki/NetIQ) Designer for Identity Manager
* [Apache Directory](https://en.wikipedia.org/wiki/Apache_Directory) Studio
* [Remote Component Environment](https://en.wikipedia.org/wiki/Remote_Component_Environment)

**Server platform:**

Eclipse supports development for [Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat), [GlassFish](https://en.wikipedia.org/wiki/GlassFish" \o "GlassFish) and many other servers and is often capable of installing the required server (for development) directly from the IDE. It supports remote debugging, allowing the user to watch variables and step through the code of an application that is running on the attached server.

**Web Tools Platform:**

The Eclipse [Web Tools Platform](https://en.wikipedia.org/wiki/Web_Tools_Platform) (WTP) project is an extension of the Eclipse platform with tools for developing Web and Java EE applications. It includes source and graphical editors for a variety of languages, wizards and built-in applications to simplify development, and tools and APIs to support deploying, running, and testing apps.

**Modeling platform:**

The Modeling project contains all the official projects of the Eclipse Foundation focusing on model-based development technologies. They are all compatible with the Eclipse Modeling Framework created by IBM. Those projects are separated in several categories: Model Transformation, Model Development Tools, Concrete Syntax Development, Abstract Syntax Development, Technology and Research, and Amalgam.

Model Transformation projects uses [EMF](https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework) based models as an input and produce either a model or text as an output. Model to model transformation projects includes [ATL](https://en.wikipedia.org/wiki/ATLAS_Transformation_Language), an open source transformation language and toolkit used to transform a given model or to generate a new model from a given [EMF](https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework) model. Model to text transformation projects contains [Acceleo](https://en.wikipedia.org/wiki/Acceleo" \o "Acceleo), an implementation of [MOFM2T](https://en.wikipedia.org/wiki/MOFM2T), a standard model to text language from the [OMG](https://en.wikipedia.org/wiki/Object_Management_Group). [Acceleo](https://en.wikipedia.org/wiki/Acceleo" \o "Acceleo) is an open source code generator that can generate any textual language (Java, PHP, Python, etc.) from [EMF](https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework) based models defined with any metamodel ([UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language), [SysML](https://en.wikipedia.org/wiki/Systems_Modeling_Language" \o "Systems Modeling Language), etc.).

Model Development Tools projects are implementations of modeling standard used in the industry like [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) or [OCL](https://en.wikipedia.org/wiki/Object_Constraint_Language) and their toolkit. Among those projects can be found implementations of the following standards:

* [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language)
* [SysML](https://en.wikipedia.org/wiki/Systems_Modeling_Language)
* [OCL](https://en.wikipedia.org/wiki/Object_Constraint_Language)
* [BPMN](https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation)
* IMM
* [SBVR](https://en.wikipedia.org/wiki/Semantics_of_Business_Vocabulary_and_Business_Rules)
* [XSD](https://en.wikipedia.org/wiki/XML_Schema_(W3C))
* NEDA

The Concrete Syntax Development project contains the Graphical Modeling Framework, an Eclipse-based framework dedicated to the graphical representation of [EMF](https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework) based models.

The Abstract Syntax Development project hosts the Eclipse Modeling Framework, core of most of the modeling project of the Eclipse Foundation and the framework available for [EMF](https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework) like [CDO](https://en.wikipedia.org/wiki/Connected_Data_Objects), EMF query or EMF validation.

Technology and Research projects are prototypes of Modeling project, this project is used to host all the modeling projects of the Eclipse Foundation during their incubation phase.

Amalgam provides the packaging and integration between all the available modeling tools for the Eclipse package dedicated to modeling tools.

**3.1.4 Operating System**

An operating system (OS) is [system software](https://en.wikipedia.org/wiki/System_software) that manages [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware) and [software](https://en.wikipedia.org/wiki/Computer_software) resources and provides common [services](https://en.wikipedia.org/wiki/Operating_system_services) for [computer programs](https://en.wikipedia.org/wiki/Computer_program). The operating system is a component of the [system software](https://en.wikipedia.org/wiki/System_software) in a computer system. [Application programs](https://en.wikipedia.org/wiki/Application_program) usually require an operating system to function.

[Time-sharing](https://en.wikipedia.org/wiki/Time-sharing) operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and [memory allocation](https://en.wikipedia.org/wiki/Dynamic_memory_allocation), the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes [system calls](https://en.wikipedia.org/wiki/System_call) to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer – from [cellular phones](https://en.wikipedia.org/wiki/Cellular_phone) and [video game consoles](https://en.wikipedia.org/wiki/Video_game_console) to [web servers](https://en.wikipedia.org/wiki/Web_server) and [supercomputers](https://en.wikipedia.org/wiki/Supercomputer).

Examples of popular desktop operating systems include [Apple](https://en.wikipedia.org/wiki/Apple_Inc.) [OS X](https://en.wikipedia.org/wiki/OS_X), [Linux](https://en.wikipedia.org/wiki/Linux) and its variants, and [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). So-called [mobile operating systems](https://en.wikipedia.org/wiki/Mobile_operating_system) include [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) and [iOS](https://en.wikipedia.org/wiki/IOS" \o "IOS)

|  |
| --- |
| **Operating systems** |
| Description: Operating system placement.svg[Description: About this image](https://en.wikipedia.org/wiki/File:Operating_system_placement.svg) |

### Types of operating systems

### Single- and multi-tasking

A single-tasking system can only run one program at a time, while a [multi-tasking](https://en.wikipedia.org/wiki/Multi-tasking) operating system allows more than one program to be running in concurrency. This is achieved by [time-sharing](https://en.wikipedia.org/wiki/Time-sharing), dividing the available processor time between multiple processes that are each interrupted repeatedly in time slices by a task-scheduling subsystem of the operating system. Multi-tasking may be characterized in preemptive and co-operative types. In preemptive multitasking, the operating system slices the [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) time and dedicates a slot to each of the programs. Unix-like operating systems, e.g., Solaris, [Linux](https://en.wikipedia.org/wiki/Linux), as well as [AmigaOS](https://en.wikipedia.org/wiki/AmigaOS" \o "AmigaOS) support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. [16-bit](https://en.wikipedia.org/wiki/16-bit) versions of Microsoft Windows used cooperative multi-tasking. [32-bit](https://en.wikipedia.org/wiki/32-bit) versions of both Windows NT and Win9x, used preemptive multi-tasking.

### Single- and multi-user

Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem. A [multi-user](https://en.wikipedia.org/wiki/Multi-user) operating system extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users.

### Distributed:

A [distributed operating system](https://en.wikipedia.org/wiki/Distributed_operating_system) manages a group of distinct computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they form a distributed system.

### Template:

In an OS, distributed and [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) context, [templating](https://en.wikipedia.org/wiki/Glossary_of_operating_systems_terms" \o "Glossary of operating systems terms) refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in [virtualization](https://en.wikipedia.org/wiki/Virtualization) and cloud computing management, and is common in large server warehouses.

### Embedded:

[Embedded operating systems](https://en.wikipedia.org/wiki/Embedded_operating_system) are designed to be used in [embedded computer systems](https://en.wikipedia.org/wiki/Embedded_system). They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

### Real-time:

A [real-time operating system](https://en.wikipedia.org/wiki/Real-time_operating_system) is an operating system that guarantees to process events or data within a certain short amount of time. A real-time operating system may be single- or multi-tasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

### Library:

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries. These libraries are composed with the application and configuration code to construct [unikernels](https://en.wikipedia.org/wiki/Unikernel" \o "Unikernel) – which are specialized, [single address space](https://en.wikipedia.org/wiki/Single_address_space_operating_system), machine images that can be deployed to cloud or embedded environments.

**3.1.5 HMM (Hidden Markov Model)**

A hidden Markov model (HMM) is a [statistical](https://en.wikipedia.org/wiki/Statistical_model) [Markov model](https://en.wikipedia.org/wiki/Markov_model) in which the system being modeled is assumed to be a [Markov process](https://en.wikipedia.org/wiki/Markov_process) with unobserved (hidden) states. A HMM can be presented as the simplest [dynamic Bayesian network](https://en.wikipedia.org/wiki/Dynamic_Bayesian_network). The mathematics behind the HMM were developed by [L. E. Baum](https://en.wikipedia.org/wiki/Leonard_E._Baum) and coworkers. It is closely related to an earlier work on the optimal nonlinear [filtering problem](https://en.wikipedia.org/wiki/Filtering_problem_(stochastic_processes)) by [Ruslan L. Stratonovich](https://en.wikipedia.org/wiki/Ruslan_L._Stratonovich" \o "Ruslan L. Stratonovich),who was the first to describe the [forward-backward procedure](https://en.wikipedia.org/wiki/Forward%E2%80%93backward_algorithm). In simpler [Markov models](https://en.wikipedia.org/wiki/Markov_model) (like a [Markov chain](https://en.wikipedia.org/wiki/Markov_chain)), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.

In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

Hidden Markov models are especially known for their application in [temporal](https://en.wikipedia.org/wiki/Time) pattern recognition such as [speech](https://en.wikipedia.org/wiki/Speech_recognition), handwriting, gesture, [part-of-speech tagging](https://en.wikipedia.org/wiki/Part-of-speech_tagging), musical score following  [partial discharges](https://en.wikipedia.org/wiki/Partial_discharge) and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics).

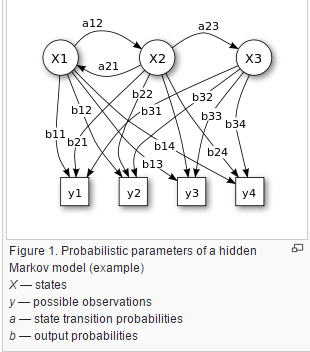
A hidden Markov model can be considered a generalization of a [mixture model](https://en.wikipedia.org/wiki/Mixture_model) where the hidden variables (or [latent variables](https://en.wikipedia.org/wiki/Latent_variables)), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other. Recently, hidden Markov models have been generalized to pair wise Markov models and triplet Markov models which allow consideration of more complex data structures and the modeling of non-stationary data.

**Description:**

In its discrete form, a hidden Markov process can be visualized as a generalization of the [Urn problem](https://en.wikipedia.org/wiki/Urn_problem) with replacement (where each item from the urn is returned to the original urn before the next step). Consider this example: in a room that is not visible to an observer there is a genie. The room contains urns X1, X2, X3, … each of which contains a known mix of balls, each ball labeled y1, y2, y3, … . The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn.

The genie has some procedure to choose urns; the choice of the urn for the n-th ball depends only upon a random number and the choice of the urn for the (n − 1)-th ball. The choice of urn does not directly depend on the urns chosen before this single previous urn; therefore, this is called a [Markov process](https://en.wikipedia.org/wiki/Markov_process). It can be described by the upper part of the Figure.

The Markov process itself cannot be observed, and only the sequence of labeled balls can be observed, thus this arrangement is called a "hidden Markov process". This is illustrated by the lower part of the diagram shown in Figure 1, where one can see that balls y1, y2, y3, y4 can be drawn at each state. Even if the observer knows the composition of the urns and has just observed a sequence of three balls, e.g. y1, y2 and y3 on the conveyor belt, the observer still cannot be sure which urn (i.e., at which state) the genie has drawn the third ball from. However, the observer can work out other information, such as the likelihood that the third ball came from each of the urns.



**Architecture:**

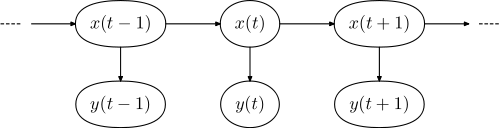
The diagram below shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable x(t) is the hidden state at time t (with the model from the above diagram, x(t) ∈ { x1, x2, x3 }). The random variable y(t) is the observation at time t (withy(t) ∈ { y1, y2, y3, y4 }). The arrows in the diagram (often called a [trellis diagram](https://en.wikipedia.org/wiki/Trellis_(graph))) denote conditional dependencies.

From the diagram, it is clear that the [conditional probability distribution](https://en.wikipedia.org/wiki/Conditional_probability_distribution) of the hidden variable x(t) at time t, given the values of the hidden variable x at all times, depends only on the value of the hidden variable x(t − 1); the values at time t − 2 and before have no influence. This is called the [Markov property](https://en.wikipedia.org/wiki/Markov_property). Similarly, the value of the observed variable y(t) only depends on the value of the hidden variable x(t) (both at time t).

In the standard type of hidden Markov model considered here, the state space of the hidden variables is discrete, while the observations themselves can either be discrete (typically generated from a [categorical distribution](https://en.wikipedia.org/wiki/Categorical_distribution)) or continuous (typically from a [Gaussian distribution](https://en.wikipedia.org/wiki/Gaussian_distribution)). The parameters of a hidden Markov model are of two types, transition probabilities and emission probabilities (also known as output probabilities). The transition probabilities control the way the hidden state at time t is chosen given the hidden state at timet-1.

The hidden state space is assumed to consist of one of N possible values, modeled as a categorical distribution. (See the section below on extensions for other possibilities.) This means that for each of the N possible states that a hidden variable at time t can be in, there is a transition probability from this state to each of the N possible states of the hidden variable at timet+1, for a total of N^2 transition probabilities. Note that the set of transition probabilities for transitions from any given state must sum to 1. Thus, the N \times Nmatrix of transition probabilities is a [Markov matrix](https://en.wikipedia.org/wiki/Stochastic_matrix). Because any one transition probability can be determined once the others are known, there are a total of N(N-1)transition parameters.

In addition, for each of the N possible states, there is a set of emission probabilities governing the distribution of the observed variable at a particular time given the state of the hidden variable at that time. The size of this set depends on the nature of the observed variable. For example, if the observed variable is discrete with M possible values, governed by a [categorical distribution](https://en.wikipedia.org/wiki/Categorical_distribution), there will be M-1 separate parameters, for a total of N(M-1) emission parameters over all hidden states. On the other hand, if the observed variable is an M-dimensional vector distributed according to an arbitrary [multivariate Gaussian distribution](https://en.wikipedia.org/wiki/Multivariate_Gaussian_distribution), there will be M parameters controlling the [means](https://en.wikipedia.org/wiki/Mean) and \frac {M(M+1)} 2 parameters controlling the [covariance matrix](https://en.wikipedia.org/wiki/Covariance_matrix), for a total of N \left(M + \frac{M(M+1)}{2}\right) = \frac {NM(M+3)} 2 = O(NM^2) emission parameters. (In such a case, unless the value of M is small, it may be more practical to restrict the nature of the covariance’s between individual elements of the observation vector, e.g. by assuming that the elements are independent of each other, or less restrictively, re independent of all but a fixed number of adjacent elements.)

[](https://en.wikipedia.org/wiki/File:Hmm_temporal_bayesian_net.svg)

**3.1.6 Probability**

Probability is the [measure](https://en.wikipedia.org/wiki/Measure_(mathematics)) of the likeliness that an [event](https://en.wikipedia.org/wiki/Event_(probability_theory)) will occur. Probability is quantified as a number between 0 and 1 (where 0 indicates impossibility and 1 indicates certainty). The higher the probability of an event, the more certain we are that the event will occur. A simple example is the toss of a fair (unbiased) coin. Since the two outcomes are equally probable, the probability of "heads" equals the probability of "tails", so the probability is 1/2 (or 50%) chance of either "heads" or "tails".

These concepts have been given an [axiomatic](https://en.wikipedia.org/wiki/Probability_axioms) [mathematical](https://en.wikipedia.org/wiki/Mathematics) formalization in [probability theory](https://en.wikipedia.org/wiki/Probability_theory) (see [probability axioms](https://en.wikipedia.org/wiki/Probability_axioms)), which is used widely in such [areas of study](https://en.wikipedia.org/wiki/Areas_of_study) as [mathematics](https://en.wikipedia.org/wiki/Mathematics), [statistics](https://en.wikipedia.org/wiki/Statistics), [finance](https://en.wikipedia.org/wiki/Finance), [gambling](https://en.wikipedia.org/wiki/Gambling), [science](https://en.wikipedia.org/wiki/Science) (in particular [physics](https://en.wikipedia.org/wiki/Physics)), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence)/[machine learning](https://en.wikipedia.org/wiki/Machine_learning), [computer science](https://en.wikipedia.org/wiki/Computer_science), game theory, and [philosophy](https://en.wikipedia.org/wiki/Philosophy) to, for example, draw inferences about the expected frequency of events. Probability theory is also used to describe the underlying mechanics and regularities of [complex systems](https://en.wikipedia.org/wiki/Complex_systems).

**Mathematical Representation:**

A probability is a [way of assigning](https://en.wikipedia.org/wiki/Function_(mathematics)) every event a value between zero and one, with the requirement that the event made up of all possible results (in our example, the event {1,2,3,4,5,6}) is assigned a value of one. To qualify as a probability, the assignment of values must satisfy the requirement that if you look at a collection of mutually exclusive events (events with no common results, e.g., the events {1, 6}, {3}, and {2, 4} are all mutually exclusive), the probability that at least one of the events will occur is given by the sum of the probabilities of all the individual events.

The probability of an [event](https://en.wikipedia.org/wiki/Event_(probability_theory)) A is written asDescription: P(A),Description: p(A), or Description: \text{Pr}(A). This mathematical definition of probability can extend to infinite sample spaces, and even uncountable sample spaces, using the concept of a measure.

The opposite or complement of an event A is the event [not A] (that is, the event of A not occurring), often denoted asDescription: \overline{A}, A^C, \neg A, orDescription: \sim A; its probability is given by P(not A) = 1 − P(A).As an example, the chance of not rolling a six on a six-sided die is 1 – (chance of rolling a six) Description: = 1 - \tfrac{1}{6} = \tfrac{5}{6}. See [Complementary event](https://en.wikipedia.org/wiki/Complementary_event) for a more complete treatment.

If two events A and B occur on a single performance of an experiment, this is called the intersection or [joint probability](https://en.wikipedia.org/wiki/Joint_distribution) of A and B, denoted asDescription: P(A \cap B).

* If two events, A and B are [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)) then the joint probability is

Description: P(A \mbox{ and }B) =  P(A \cap B) = P(A) P(B),\,

For example, if two coins are flipped the chance of both being heads is Description: \tfrac{1}{2}\times\tfrac{1}{2} = \tfrac{1}{4}.

* If either event A or event B occurs on a single performance of an experiment this is called the union of the events A and B denoted as Description: P(A \cup B). If two events are [mutually exclusive](https://en.wikipedia.org/wiki/Mutually_exclusive_events) then the probability of either occurring is

Description: P(A\mbox{ or }B) =  P(A \cup B)= P(A) + P(B).

* If the events are not mutually exclusive then

Description: P\left(A \hbox{ or } B\right)=P\left(A\right)+P\left(B\right)-P\left(A \mbox{ and } B\right).

For example, when drawing a single card at random from a regular deck of cards, the chance of getting a heart or a face card (J,Q,K) (or one that is both) is Description: \tfrac{13}{52} + \tfrac{12}{52} - \tfrac{3}{52} = \tfrac{11}{26}, because of the 52 cards of a deck 13 are hearts, 12 are face cards, and 3 are both: here the possibilities included in the "3 that are both" are included in each of the "13 hearts" and the "12 face cards" but should only be counted once.

## 3.1.7 Transition Probabilities

The one-step transition probability is the probability of transitioning from one state to another in a single step. The Markov chain is said to be time homogeneous if the transition probabilities from one state to another are independent of time index Description: $n$.

Description: \begin{displaymath}
p_{ij} = Pr\{X_{n}=j \vert X_{n-1}=i \}
\end{displaymath}

The transition probability matrix, Description: $P$, is the matrix consisting of the one-step transition probabilities, Description: $p_{ij}$.

The Description: $m$-step transition probability is the probability of transitioning from state Description: $i$ to state j  in Description: $m$ steps.

Description: \begin{displaymath}
p^{(m)}_{ij} = Pr\{X_{n+m}=j \vert X_{n}=i \}
\end{displaymath}

The Description: $m$-step transition matrix whose elements are the Description: $m$-step transition probabilities  is denoted as .

The Description: $m$-step transition probabilities can be found from the single-step transition probabilities as follows.

To transition from i to j in Description: $m$ steps, the process can first transition from i to r in m-k steps, and then transition from r to j in Description: $k$ steps, where 0<k<m.

Description: \begin{displaymath}
p^{(m)}_{ij} = \sum_{r} p^{m-k}_{ir} p^{k}_{rj}
\end{displaymath}

In matrix form, this becomes:

Description: \begin{displaymath}
P^{(m)} = P^{(m-k)}P^{(k)}
\end{displaymath}

Setting K=M-1 yields:

Description: \begin{displaymath}
P^{(m)} = P \cdot P^{(m-1)}
\end{displaymath}

From this equation we can see that:

Description: \begin{displaymath}
P^{(m-1)} = P \cdot P^{(m-2)}
\end{displaymath}

Substituting this back into the previous equation yields:

Description: \begin{displaymath}
P^{(m)} = P \cdot P \cdot P^{(m-2)}
\end{displaymath}

Continuing these substitutions, eventually we have:

Description: \begin{displaymath}
P^{(m)} = P \cdot P \cdot P \cdots P = P^{m}
\end{displaymath}

Therefore, the Description: $m$-step transition probability matrix can be found by multiplying the single-step probability matrix by itself Description: $m$ times.

The state vector at time can also be found in terms of the transition probability matrix and the initial state vector. We first observe that:

Description: \begin{displaymath}
\pi_{j}(m) = \sum_{i} \pi_{i}(m-1) p_{ij}
\end{displaymath}

In vector and matrix form, this becomes:

Description: \begin{displaymath}
\Pi(m) = \Pi(m-1) P
\end{displaymath}

We also find that, through substitution:

Description: \begin{displaymath}
\Pi(m-1) = \Pi(m-2) P
\end{displaymath}

or,

Description: \begin{displaymath}
\Pi(m) = \Pi(m-2) P \cdot P
\end{displaymath}

Continuing the substitution yields:

Description: \begin{displaymath}
\Pi(m) = \Pi(0) P^{m}
\end{displaymath}

Where is the vector containing the initial probabilities of being in each state at time 0.

**3.1.8 Conditional Probability**

 A conditional probability measures the [probability](https://en.wikipedia.org/wiki/Probability) of an [event](https://en.wikipedia.org/wiki/Event_(probability_theory)) given that (by assumption, presumption, assertion or evidence) another event has occurred. If the event of interest is A and the event B is known or assumed to have occurred, "the conditional probability of A given B", or "the probability of A under the condition B", is usually written as P(A|B), or sometimes PB(A). For example, the probability that any given person has a cough on any given day may be only 5%. But if we know or assume that the person has a [cold](https://en.wikipedia.org/wiki/Common_cold), then they are much more likely to be coughing. The conditional probability of coughing given that you have a cold might be a much higher 75%.

The concept of conditional probability is one of the most fundamental and one of the most important concepts in probability theory. But conditional probabilities can be quite slippery and require careful interpretation For example, there need not be a causal or temporal relationship between A and B.

P(A|B) may or may not be equal to P(A) (the unconditional probability of A).

If P (A|B) = P(A), then A and B are said to be independent. Also, in general, P(A|B) (the conditional probability of A given B) is not equal to P(B|A). For example, if you have cancer you might have a 90% chance of testing positive for cancer, but if you test positive for cancer you might have only a 10% chance of actually having cancer because cancer is very rare. Falsely equating the two probabilities causes various errors of reasoning such as the [base rate fallacy](https://en.wikipedia.org/wiki/Base_rate_fallacy). Conditional probabilities can be correctly reversed using [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem).

### Conditioning on an event:

#### Kolmogorov definition:

Given two [events](https://en.wikipedia.org/wiki/Event_(probability_theory)) A and B from the sigma-field of a probability space with P(B) > 0, the conditional probability of A given B is defined as the [quotient](https://en.wikipedia.org/wiki/Quotient) of the probability of the joint of events A and B, and the [probability](https://en.wikipedia.org/wiki/Probability) of B:

Description: P(A|B) = \frac{P(A \cap B)}{P(B)}

This may be visualized as restricting the sample space to B. The logic behind this equation is that if the outcomes are restricted to B, this set serves as the new sample space.

Note that this is a definition but not a theoretical result. We just denote the quantity Description: P(A\cap B)/P(B) as Description: P(A|B) and call it the conditional probability of A given B.

#### As an axiom of probability:

Some authors, such as [De Finetti](https://en.wikipedia.org/wiki/Bruno_de_Finetti), prefer to introduce conditional probability as an [axiom of probability](https://en.wikipedia.org/wiki/Probability_axioms):

Description: P(A \cap B) = P(A|B)P(B)

Although mathematically equivalent, this may be preferred philosophically; under major [probability interpretations](https://en.wikipedia.org/wiki/Probability_interpretations) such as the [subjective theory](https://en.wikipedia.org/wiki/Subjective_probability), conditional probability is considered a primitive entity. Further, this "multiplication axiom" introduces a symmetry with the summation axiom for [mutually exclusive events](https://en.wikipedia.org/wiki/Mutually_exclusive_events):

Description: P(A \cup B) = P(A) + P(B) - \cancelto0{P(A \cap B)}.

**3.1.9 Bayes Theorem**

In [probability theory](https://en.wikipedia.org/wiki/Probability_theory) and [statistics](https://en.wikipedia.org/wiki/Statistics), Bayes' theorem (alternatively Bayes' law or [Bayes' rule](https://en.wikipedia.org/wiki/Bayes%27_rule)) describes the [probability](https://en.wikipedia.org/wiki/Probability) of an [event](https://en.wikipedia.org/wiki/Event_(probability_theory)), based on conditions that might be related to the event. For example, suppose one is interested in whether Addison has cancer, and that she is 65. If cancer is related to age, information about Addison's age can be used to more accurately assess the probability of her having cancer using Bayes' Theorem.

When applied, the probabilities involved in Bayes' theorem may have different [probability interpretations](https://en.wikipedia.org/wiki/Probability_interpretation). In one of these interpretations, the theorem is used directly as part of a particular approach to [statistical inference](https://en.wikipedia.org/wiki/Statistical_inference). With the [Bayesian probability](https://en.wikipedia.org/wiki/Bayesian_probability) interpretation the theorem expresses how a subjective degree of belief should rationally change to account for evidence: this is [Bayesian inference](https://en.wikipedia.org/wiki/Bayesian_inference), which is fundamental to [Bayesian statistics](https://en.wikipedia.org/wiki/Bayesian_statistics). However, Bayes' theorem has applications in a wide range of calculations involving probabilities, not just in Bayesian inference.

**Derivation:**

Bayes' theorem may be derived from the definition of [conditional probability](https://en.wikipedia.org/wiki/Conditional_probability):

Description: P(A\mid B)=\frac{P(A \cap B)}{P(B)}, \text{ if } P(B) \neq 0, \!

Description: P(B\mid A) = \frac{P(A \cap B)}{P(A)}, \text{ if } P(A) \neq 0, \!

Description: \Rightarrow P(A \cap B) = P(A\mid B)\, P(B) = P(B\mid A)\, P(A), \!

Description: \Rightarrow P(A\mid B) = \frac{P(B\mid A)\,P(A)}{P(B)}, \text{ if } P(B) \neq 0.

### For random variables

For two continuous [random variables](https://en.wikipedia.org/wiki/Random_variable) X and Y, Bayes' theorem may be analogously derived from the definition of [conditional density](https://en.wikipedia.org/wiki/Conditional_density):

Description: f_X(x\mid Y=y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} 

Description: f_Y(y\mid X=x) = \frac{f_{X,Y}(x,y)}{f_X(x)} 

Description: \Rightarrow f_X(x\mid Y=y) = \frac{f_Y(y\mid X=x)\,f_X(x)}{f_Y(y)}.

**3.2 HARDWARE REQUIREMENTS**

* 1GB External Memory.
* Java Runtime Environment.
* Keyboard(standard).
* Mouse(standard).
* Monitor(basic).
* 512 MB RAM (Random Access Memory).
* Multiprocessor.
* Greater than 2 GB ROM (Read Only Memory)/Secondary Memory.

**3.2.1 External Memory:**

A hard disk drive (HDD), hard disk, hard drive or fixed diskis a [data storage device](https://en.wikipedia.org/wiki/Data_storage_device) used for storing and retrieving [digital](https://en.wikipedia.org/wiki/Digital_data) information using one or more rigid ("hard") rapidly rotating disks ([platters](https://en.wikipedia.org/wiki/Hard_disk_platter)) coated with magnetic material. The platters are paired with [magnetic heads](https://en.wikipedia.org/wiki/Disk_read-and-write_head) arranged on a moving [actuator](https://en.wikipedia.org/wiki/Actuator) arm, which read and write data to the platter surfaces. Data is accessed in a [random-access](https://en.wikipedia.org/wiki/Random-access) manner, meaning that individual [blocks](https://en.wikipedia.org/wiki/Block_(data_storage)) of data can be stored or retrieved in any order and not only [sequentially](https://en.wikipedia.org/wiki/Sequential_access). HDDs are a type of [non-volatile memory](https://en.wikipedia.org/wiki/Non-volatile_memory), retaining stored data even when powered off.

Introduced by [IBM](https://en.wikipedia.org/wiki/IBM) in 1956, HDDs became the dominant [secondary storage](https://en.wikipedia.org/wiki/Secondary_storage) device for [general-purpose computers](https://en.wikipedia.org/wiki/History_of_general-purpose_CPUs) by the early 1960s. Continuously improved, HDDs have maintained this position into the modern era of [servers](https://en.wikipedia.org/wiki/Server_(computing)) and [personal computers](https://en.wikipedia.org/wiki/Personal_computer). More than 200 companies have produced HDD units, though most current units are manufactured by [Seagate](https://en.wikipedia.org/wiki/Seagate_Technology), [Toshiba](https://en.wikipedia.org/wiki/Toshiba) and [Western Digital](https://en.wikipedia.org/wiki/Western_Digital). As of 2015, HDD production (exabytes per year) and areal density are growing, although unit shipments are declining.

The primary characteristics of an HDD are its capacity and [performance](https://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics). Capacity is specified in [unit prefixes](https://en.wikipedia.org/wiki/Unit_prefix) corresponding to powers of 1000: a 1-[terabyte](https://en.wikipedia.org/wiki/Terabyte) (TB) drive has a capacity of 1,000 [gigabytes](https://en.wikipedia.org/wiki/Gigabyte) (GB; where 1 gigabyte = 1 billion [bytes](https://en.wikipedia.org/wiki/Byte)). Typically, some of an HDD's capacity is unavailable to the user because it is used by the [file system](https://en.wikipedia.org/wiki/File_system) and the computer [operating system](https://en.wikipedia.org/wiki/Operating_system), and possibly inbuilt redundancy for [error correction](https://en.wikipedia.org/wiki/Error_correction) and recovery. Performance is specified by the time required to move the heads to a track or cylinder (average access time) plus the time it takes for the desired sector to move under the head (average [latency](https://en.wikipedia.org/wiki/Latency_(engineering)), which is a function of the physical [rotational speed](https://en.wikipedia.org/wiki/Rotational_speed) in [revolutions per minute](https://en.wikipedia.org/wiki/Revolutions_per_minute)), and finally the speed at which the data is transmitted (data rate).

The two most common [form factors](https://en.wiktionary.org/wiki/form_factor) for modern HDDs are 3.5-[inch](https://en.wikipedia.org/wiki/Inch), for desktop computers, and 2.5-inch, primarily for laptops. HDDs are connected to systems by standard [interface](https://en.wikipedia.org/wiki/Computer_interface) cables such as [PATA](https://en.wikipedia.org/wiki/Parallel_ATA) (Parallel ATA), [SATA](https://en.wikipedia.org/wiki/SATA) (Serial ATA), [USB](https://en.wikipedia.org/wiki/USB) or SAS ([Serial attached SCSI](https://en.wikipedia.org/wiki/Serial_attached_SCSI)) cables.

As of 2016, the primary competing technology for secondary storage is [flash memory](https://en.wikipedia.org/wiki/Flash_memory) in the form of [solid-state drives](https://en.wikipedia.org/wiki/Solid-state_drive) (SSDs), which have higher data transfer rates, better reliability, and significantly lower latency and access times, but HDDs remain the dominant medium for secondary storage due to advantages in price per bit. However, SSDs are replacing HDDs where speed, [power consumption](https://en.wikipedia.org/wiki/Electric_energy_consumption) and durability are more important considerations. [Hybrid drive](https://en.wikipedia.org/wiki/Hybrid_drive) products, also known by the [initialism](https://en.wikipedia.org/wiki/Initialism" \o "Initialism) SSHD, have been available since 2007, made as a combination of HDD and SSD technology in a single device.

**3.2.2 Keyboard Standard:**

In [computing](https://en.wikipedia.org/wiki/Computing), a computer keyboard is typewriter which uses an arrangement of buttons or [keys](https://en.wikipedia.org/wiki/Push-button) to act as a mechanical lever or electronic switch. Following the decline of [punch cards](https://en.wikipedia.org/wiki/Punch_card) and [paper tape](https://en.wikipedia.org/wiki/Paper_tape), interaction via [teleprinter](https://en.wikipedia.org/wiki/Teleprinter" \o "Teleprinter)-style keyboards became the main [input device](https://en.wikipedia.org/wiki/Input_device) for [computers](https://en.wikipedia.org/wiki/Computer).

A keyboard typically has characters [en graved](https://en.wikipedia.org/wiki/Engraving) or [printed](https://en.wikipedia.org/wiki/Printing) on the keys and each press of a key typically corresponds to a single written [symbol](https://en.wikipedia.org/wiki/Symbol). However, to produce some symbols requires pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce [letters](https://en.wikipedia.org/wiki/Letter_(alphabet)), [numbers](https://en.wikipedia.org/wiki/Numerical_digit) or signs ([characters](https://en.wikipedia.org/wiki/Character_(computing))), other keys or simultaneous key presses can produce actions or execute computer commands.

Despite the development of alternative input devices, such as the [mouse](https://en.wikipedia.org/wiki/Mouse_(computing)), [touchscreen](https://en.wikipedia.org/wiki/Touchscreen), [pen devices](https://en.wikipedia.org/wiki/Pen_computing), [character recognition](https://en.wikipedia.org/wiki/Character_recognition) and [voice recognition](https://en.wikipedia.org/wiki/Speech_recognition), the keyboard remains the most commonly used device for direct (human) input of [alphanumeric](https://en.wikipedia.org/wiki/Alphanumeric) data into computers.

In normal usage, the keyboard is used as a [text entry interface](https://en.wikipedia.org/wiki/Text_Entry_Interface) to type text and numbers into a [word processor](https://en.wikipedia.org/wiki/Word_processor), [text editor](https://en.wikipedia.org/wiki/Text_editor) or other programs. In a modern computer, the interpretation of key presses is generally left to the software. A computer keyboard distinguishes each physical key from every other and reports all key presses to the controlling software. Keyboards are also used for computer gaming, either with regular keyboards or by using keyboards with special gaming features, which can expedite frequently used keystroke combinations. A keyboard is also used to give commands to the operating system of a computer, such as [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows)' [Control-Alt-Delete](https://en.wikipedia.org/wiki/Control-Alt-Delete) combination, which brings up a task window or shuts down the machine. A [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) is a type of [user interface](https://en.wikipedia.org/wiki/User_interface) operated entirely through a keyboard, or another device doing the job of one.

**Standard:**

Standard alphanumeric keyboards have keys that are on three-quarter inch centers (0.750 inches, 19.05 mm), and have a key travel of at least 0.150 inches (3.81 mm). Desktop computer keyboards, such as the 101-key US traditional keyboards or the 104-key Windows keyboards, include alphabetic characters, [punctuation](https://en.wikipedia.org/wiki/Punctuation) symbols, numbers and a variety of [function keys](https://en.wikipedia.org/wiki/Function_Keys). The internationally common 102/104 key keyboards have a smaller left shift key and an additional key with some more symbols between that and the letter to its right (usually Z or Y). Also the [enter key](https://en.wikipedia.org/wiki/Enter_key) is usually shaped differently. Computer keyboards are similar to electric-typewriter keyboards but contain additional keys, such as the command or Windows keys.

**3.2.3 Mouse:**

A computer mouse is a [pointing device](https://en.wikipedia.org/wiki/Pointing_device)(hand control) that detects [two-dimensional](https://en.wikipedia.org/wiki/Two-dimensional_space) motion relative to a surface. This motion is typically translated into the motion of a [pointer](https://en.wikipedia.org/wiki/Pointer_(graphical_user_interfaces)) on a [display](https://en.wikipedia.org/wiki/Computer_monitor), which allows a smooth control of the [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface).

Physically, a mouse consists of an object held in one's hand, with one or more buttons. Mice often also feature other elements, such as touch surfaces and "wheels", which enable additional control and dimensional input.

**Operation:**

A mouse typically controls the motion of a [pointer](https://en.wikipedia.org/wiki/Pointer_(graphical_user_interfaces)) in two dimensions in a graphical user interface (GUI). The mouse turns movements of the hand backward and forward, left and right into equivalent electronic signals that in turn are used to move the pointer.

The relative movements of the mouse on the surface are applied to the position of the pointer on the screen, which signals the point where actions of the user take place, so that the hand movements are replicated by the pointer. Clicking or hovering (stopping movement while the cursor is within the bounds of an area) can select files, programs or actions from a list of names, or (in graphical interfaces) through small images called "icons" and other elements. For example, a text file might be represented by a picture of a paper notebook, and clicking while the cursor hovers this icon might cause a text editing program to open the file in a window.

Different ways of operating the mouse cause specific things to happen in the GUI:

* Click: pressing and releasing a button.
  + (left) [Single-click](https://en.wikipedia.org/wiki/Point_and_click): clicking the main button.
  + (left) [Double-click](https://en.wikipedia.org/wiki/Double-click): clicking the button two times in quick succession counts as a different gesture than two separate single clicks.
  + (left) [Triple-click](https://en.wikipedia.org/wiki/Triple-click): clicking the button three times in quick succession.
  + [Right-click](https://en.wikipedia.org/wiki/Right-click): clicking the secondary button.
  + Middle-click: clicking the tertiary button.
* [Drag and drop](https://en.wikipedia.org/wiki/Drag_and_drop): pressing and holding a button, then moving the mouse without releasing. (Using the command "[drag with the right mouse button](https://en.wikipedia.org/wiki/Right-drag)" instead of just "drag" when one instructs a user to drag an object while holding the right mouse button down instead of the more commonly used left mouse button.)
* [Mouse button chording](https://en.wikipedia.org/wiki/Mouse_chording) (a.k.a. Rocker navigation).
  + Combination of right-click then left-click.
  + Combination of left-click then right-click or keyboard letter.
  + Combination of left or right-click and the mouse wheel.
* Clicking while holding down a [modifier key](https://en.wikipedia.org/wiki/Modifier_key).
* Moving the pointer a long distance: When a practical limit of mouse movement is reached, one lifts up the mouse, brings it to the opposite edge of the working area while it is held above the surface, and then replaces it down onto the working surface. This is often not necessary, because acceleration software detects fast movement, and moves the pointer significantly faster in proportion than for slow mouse motion.
* Multi-touch: this method is similar to a multi-touch track pad on a laptop with support for tap input for multiple fingers, the most famous example being the [Apple Magic Mouse](https://en.wikipedia.org/wiki/Apple_Magic_Mouse).

[](https://en.wikipedia.org/wiki/File:Logitechms48.jpg)

* Mechanical mouse, shown with the top cover removed. The scroll wheel is grey, to the right of the ball.

### Mouse Gestures:

Users can also employ mice gestural; meaning that a stylized motion of the mouse cursor itself, called a "[gesture](https://en.wikipedia.org/wiki/Pointing_device_gesture)", can issue a command or map to a specific action. For example, in a drawing program, moving the mouse in a rapid "x" motion over a shape might delete the shape.

Gestural interfaces occur more rarely than plain pointing-and-clicking; and people often find them more difficult to use, because they require finer motor-control from the user. However, a few gestural conventions have become widespread, including the [drag and drop](https://en.wikipedia.org/wiki/Drag_and_drop) gesture, in which:

1. The user presses the mouse button while the mouse cursor hovers over an interface object
2. The user moves the cursor to a different location while holding the button down
3. The user releases the mouse button

For example, a user might drag-and-drop a picture representing a file onto a picture of a [trash can](https://en.wikipedia.org/wiki/Trash_(computing)), thus instructing the system to delete the file.

Standard semantic gestures include:

* [Crossing-based goal](https://en.wikipedia.org/wiki/Crossing-based_interface)
* [Drag and drop](https://en.wikipedia.org/wiki/Drag_and_drop)
* [Menu](https://en.wikipedia.org/wiki/Menu_(computing)) traversal
* Pointing
* [Rollover](https://en.wikipedia.org/wiki/Rollover_(web_design)) ([Mouseover](https://en.wikipedia.org/wiki/Mouseover" \o "Mouseover))
* [Selection](https://en.wikipedia.org/wiki/Selection_(user_interface)).

### 3.2.4 Computer Monitor:

A computer monitor or a computer display is an [electronic](https://en.wikipedia.org/wiki/Electronics) [visual display](https://en.wikipedia.org/wiki/Electronic_visual_display) for [computers](https://en.wikipedia.org/wiki/Computer). A monitor usually comprises the [display device](https://en.wikipedia.org/wiki/Display_device), [circuitry](https://en.wikipedia.org/wiki/Electronic_circuit), casing, and power supply. The display device in modern monitors is typically a [thin film transistor liquid crystal display](https://en.wikipedia.org/wiki/Thin_film_transistor_liquid_crystal_display) (TFT-LCD) or a flat panel [LED display](https://en.wikipedia.org/wiki/LED_display), while older monitors used a [cathode ray tubes](https://en.wikipedia.org/wiki/Cathode_ray_tube) (CRT). It can be connected to the computer via [VGA](https://en.wikipedia.org/wiki/VGA_connector), [DVI](https://en.wikipedia.org/wiki/Digital_Visual_Interface),[HDMI](https://en.wikipedia.org/wiki/HDMI), [Display Port](https://en.wikipedia.org/wiki/DisplayPort), [Thunderbolt](https://en.wikipedia.org/wiki/Thunderbolt), [LVDS](https://en.wikipedia.org/wiki/Low-voltage_differential_signaling) (Low-voltage differential signaling) or other proprietary connectors and signals.

Originally, computer monitors were used for [data processing](https://en.wikipedia.org/wiki/Data_processing) while [television receivers](https://en.wikipedia.org/wiki/Television_receiver) were used for entertainment. From the 1980s onwards, computers (and their monitors) have been used for both data processing and entertainment, while televisions have implemented some computer functionality. The common [aspect ratio](https://en.wikipedia.org/wiki/Aspect_ratio) of televisions, and computer monitors, has changed from 4:3 to 16:10, to 16:9.

**Technologies:**

Multiple technologies have been used for computer monitors. Until the 21st century most used cathode ray tubes but they have largely been superseded by [LCD monitors](https://en.wikipedia.org/wiki/LCD_monitors).

### Cathode ray tube:

The first computer monitors used [cathode ray tubes](https://en.wikipedia.org/wiki/Cathode_ray_tube) (CRTs). Prior to the advent of [home computers](https://en.wikipedia.org/wiki/Home_computer) in the late 1970s, it was common for a [video display terminal](https://en.wikipedia.org/wiki/Video_display_terminal) (VDT) using a CRT to be physically integrated with a keyboard and other components of the system in a single large [chassis](https://en.wikipedia.org/wiki/Chassis). The display was [monochrome](https://en.wikipedia.org/wiki/Monochrome) and far less sharp and detailed than on a modern flat-panel monitor, necessitating the use of relatively large text and severely limiting the amount of information that could be displayed at one time. High-resolution CRT displays were developed for specialized military, industrial and scientific applications but they were far too costly for general use.

Some of the earliest home computers (such as the [TRS-80](https://en.wikipedia.org/wiki/TRS-80) and [Commodore PET](https://en.wikipedia.org/wiki/Commodore_PET)) were limited to monochrome CRT displays, but color display capability was already a standard feature of the pioneering [Apple II](https://en.wikipedia.org/wiki/Apple_II), introduced in 1977, and the specialty of the more graphically sophisticated [Atari 800](https://en.wikipedia.org/wiki/Atari_8-bit_family), introduced in 1979. Either computer could be connected to the antenna terminals of an ordinary color TV set or used with a purpose-made CRT color monitor for optimum resolution and color quality. Lagging several years behind, in 1981 IBM introduced the [Color Graphics Adapter](https://en.wikipedia.org/wiki/Color_Graphics_Adapter), which could display four colors with a resolution of 320 x 200 pixels, or it could produce 640 x 200 pixels with two colors. In 1984 IBM introduced the [Enhanced Graphics Adapter](https://en.wikipedia.org/wiki/Enhanced_Graphics_Adapter) which was capable of producing 16 colors and had a resolution of 640 x 350.

By the end of the 1980s color CRT monitors that could clearly display 1024 x 768 pixels were widely available and increasingly affordable. During the following decade maximum display resolutions gradually increased and prices continued to fall. CRT technology remained dominant in the PC monitor market into the new millennium partly because it was cheaper to produce and offered viewing angles close to 180 degrees. CRTs still offer some image quality advantage over LCDs but improvements to the latter have made them much less obvious. The dynamic range of early LCD panels was very poor, and although text and other motionless graphics were sharper than on a CRT, an LCD characteristic known as pixel lag caused moving graphics to appear noticeably smeared and blurry.

### Liquid crystal display:

There are multiple technologies that have been used to implement liquid crystal displays (LCD). Throughout the 1990s, the primary use of LCD technology as computer monitors was in laptops where the lower power consumption, lighter weight, and smaller physical size of LCDs justified the higher price versus a CRT. Commonly, the same laptop would be offered with an assortment of display options at increasing price points: (active or passive) monochrome, passive color, or active matrix color (TFT). As volume and manufacturing capability have improved, the monochrome and passive color technologies were dropped from most product lines.

[TFT-LCD](https://en.wikipedia.org/wiki/TFT-LCD) is a variant of LCD which is now the dominant technology used for computer monitors.

The first standalone LCDs appeared in the mid-1990s selling for high prices. As prices declined over a period of years they became more popular, and by 1997 were competing with CRT monitors. Among the first desktop LCD computer monitors was the Eizo L66 in the mid-1990s, the Apple Studio Display in 1998, and the Apple Cinema Display in 1999. In 2003, TFT-LCDs outsold CRTs for the first time, becoming the primary technology used for computer monitors.[[2]](https://en.wikipedia.org/wiki/Computer_monitor#cite_note-pctech1-2) The main advantages of LCDs over CRT displays are that LCDs consume less power, take up much less space, and are considerably lighter. The now common active matrix TFT-LCD technology also has less flickering than CRTs, which reduces eye strain.[[4]](https://en.wikipedia.org/wiki/Computer_monitor#cite_note-4) On the other hand, CRT monitors have superior contrast, have superior response time, are able to use multiple screen resolutions natively, and there is no discernible flicker if the refresh rate is set to a sufficiently high value. LCD monitors have now very high temporal accuracy and can be used for vision research.

**3.2.5 Random Access Memory (RAM):**

Random-access memory  is a form of [computer data storage](https://en.wikipedia.org/wiki/Computer_data_storage). A random-access memory device allows [data](https://en.wikipedia.org/wiki/Data) items to be accessed ([read](https://en.wikipedia.org/wiki/Read_(computer)) or written) in almost the same amount of time irrespective of the physical location of data inside the memory. In contrast, with other direct-access data storage media such as [hard disks](https://en.wikipedia.org/wiki/Hard_disk), [CD-RWs](https://en.wikipedia.org/wiki/CD-RW), [DVD-RWs](https://en.wikipedia.org/wiki/DVD-RW) and the older [drum memory](https://en.wikipedia.org/wiki/Drum_memory), the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement.

RAM contains [multiplexing](https://en.wikipedia.org/wiki/Multiplexor) and [de-multiplexing](https://en.wikipedia.org/wiki/Demultiplexing) circuitry to connect the data lines to the addressed storage for reading or writing the entry. Usually more than one bit of storage is accessed by the same address, and RAM devices often have multiple data lines and are said to be '8-bit' or '16-bit' etc. devices.

Today, random-access memory takes the form of [integrated circuits](https://en.wikipedia.org/wiki/Integrated_circuit). RAM is normally associated with [volatile](https://en.wikipedia.org/wiki/Volatile_memory) types of memory (such as [DRAM](https://en.wikipedia.org/wiki/DRAM) [memory modules](https://en.wikipedia.org/wiki/DIMM)), where stored information is lost if power is removed, although many efforts have been made to develop non-volatile RAM chips. Other types of [non-volatile memories](https://en.wikipedia.org/wiki/Non-volatile_memory) exist that allow random access for read operations, but either do not allow write operations or have limitations on them. These include most types of [ROM](https://en.wikipedia.org/wiki/Read_only_memory) and a type of [flash memory](https://en.wikipedia.org/wiki/Flash_memory) called [NOR-Flash](https://en.wikipedia.org/wiki/Flash_memory#NOR_flash).

**3.2.6 Multiprocessor:**

Multiprocessing is the use of two or more [central processing units](https://en.wikipedia.org/wiki/CPU) (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined ([multiple cores](https://en.wikipedia.org/wiki/Multi-core_(computing)) on one [die](https://en.wikipedia.org/wiki/Die_(integrated_circuit)), multiple dies in one [package](https://en.wikipedia.org/wiki/Chip_carrier), multiple packages in one [system unit](https://en.wikipedia.org/wiki/System_unit), etc.).

According to some on-line dictionaries, a multiprocessor is a computer system having two or more processing units (multiple processors) each sharing [main memory](https://en.wikipedia.org/wiki/Main_memory) and peripherals, in order to simultaneously process programs. A 2009 textbook defined multiprocessor system similarly, but noting that the processors may share "some or all of the system’s memory and I/O facilities"; it also gave tightly coupled system as a synonymous term.

At the [operating system](https://en.wikipedia.org/wiki/Operating_system) level, multiprocessing is sometimes used to refer to the execution of multiple concurrent [processes](https://en.wikipedia.org/wiki/Process_(computing)) in a system, with each process running on a separate CPU or core, as opposed to a single process at any one instant. When used with this definition, multiprocessing is sometimes contrasted with [multitasking](https://en.wikipedia.org/wiki/Multitasking), which may use just a single processor but switch it in time slices between tasks (i.e. a [time-sharing system](https://en.wikipedia.org/wiki/Time-sharing_system)). Multiprocessing however means true parallel execution of multiple processes using more than one processor. Multiprocessing doesn't necessarily mean that a single process or task uses more than one processor simultaneously; the term [parallel processing](https://en.wikipedia.org/wiki/Parallel_computing) is generally used to denote that scenario. Other authors prefer to refer to the operating system techniques as [multiprogramming](https://en.wikipedia.org/wiki/Multiprogramming) and reserve the term multiprocessing for the hardware aspect of having more than one processor. The remainder of this article discusses multiprocessing only in this hardware sense.

In [Flynn's taxonomy](https://en.wikipedia.org/wiki/Flynn%27s_taxonomy), multiprocessors as defined above are [MIMD](https://en.wikipedia.org/wiki/MIMD) machines. As they are normally construed to be tightly coupled (share memory), multiprocessors are not the entire class of MIMD machines, which also contains [message passing](https://en.wikipedia.org/wiki/Message_passing) multicomputer systems.

**3.2.7 Read Only Memory (ROM):**

Read-only memory (ROM) is a type of [non-volatile memory](https://en.wikipedia.org/wiki/Non-volatile_memory) used in [computers](https://en.wikipedia.org/wiki/Computer) and other electronic devices. Data stored in ROM can only be modified slowly, with difficulty, or not at all, so it is mainly used to store [firmware](https://en.wikipedia.org/wiki/Firmware) ([software](https://en.wikipedia.org/wiki/Software) that is closely tied to specific [hardware](https://en.wikipedia.org/wiki/Computer_hardware) and unlikely to need frequent updates) or application software in plug-in cartridges.

Strictly, read-only memory refers to memory that is hard-wired, such as [diode matrix](https://en.wikipedia.org/wiki/Diode_matrix) and the later [mask ROM](https://en.wikipedia.org/wiki/Mask_ROM) (MROM) which cannot be changed after manufacture. Although discrete circuits can be altered in principle, [integrated circuits](https://en.wikipedia.org/wiki/Integrated_circuit) (ICs) cannot, and are useless if the data is bad or requires an update. That such memory can never be changed is a disadvantage in many applications, as bugs and security issues cannot be fixed, and new features cannot be added.

More recently, ROM has come to include memory that is read-only in normal operation, but can still be reprogrammed in some way. [Erasable programmable read-only memory](https://en.wikipedia.org/wiki/EPROM) (EPROM) and [electrically erasable programmable read-only memory](https://en.wikipedia.org/wiki/EEPROM)(EEPROM) can be erased and re-programmed, but usually this can only be done at relatively slow speeds, may require special equipment to achieve, and is typically only possible a certain number of times.

The simplest type of [solid-state](https://en.wikipedia.org/wiki/Solid_state_(electronics)) ROM is as old as the [semiconductor technology](https://en.wikipedia.org/wiki/Transistor) itself. [Combinational](https://en.wikipedia.org/wiki/Combinational_logic) [logic gates](https://en.wikipedia.org/wiki/Logic_gate) can be joined manually to map n-bit address input onto arbitrary values of m-bit data output (a [look-up table](https://en.wikipedia.org/wiki/Look-up_table)). With the invention of the [integrated circuit](https://en.wikipedia.org/wiki/Integrated_circuit) came [mask ROM](https://en.wikipedia.org/wiki/Mask_ROM). Mask ROM consists of a grid of [word](https://en.wikipedia.org/wiki/Word_(data_type)) lines (the address input) and bit lines (the data output), selectively joined together with transistor switches, and can represent an arbitrary look-up table with a regular physical layout and predictable [propagation delay](https://en.wikipedia.org/wiki/Propagation_delay).

In mask ROM, the data is physically encoded in the circuit, so it can only be programmed during fabrication. This leads to a number of serious disadvantages:

1. It is only economical to buy mask ROM in large quantities, since users must contract with a [foundry](https://en.wikipedia.org/wiki/Foundry_(electronics)) to produce a custom design.
2. The turnaround time between completing the design for a mask ROM and receiving the finished product is long, for the same reason.
3. Mask ROM is impractical for [R&D](https://en.wikipedia.org/wiki/R%26D) work since designers frequently need to modify the contents of memory as they refine a design.
4. If a product is shipped with faulty mask ROM, the only way to fix it is to [recall](https://en.wikipedia.org/wiki/Product_recall) the product and physically replace the ROM in every unit shipped.

Subsequent developments have addressed these shortcomings.

**4. DESIGN**

**4.1** **INTRODUCTION**

**Markov Chain:**

A Markov chain is a [stochastic process](https://en.wikipedia.org/wiki/Stochastic_process) with the [Markov property](https://en.wikipedia.org/wiki/Markov_property). The term "Markov chain" refers to the sequence of random variables such a process moves through, with the Markov property defining [serial dependence](https://en.wikipedia.org/wiki/Serial_dependence) only between adjacent periods (as in a "chain"). It can thus be used for describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system.

We describe a Markov chain as follows: We have a set of states ,S={s1,s2,,,,sr}. The process starts in one of these states and moves successively from one state to another. Each move is called a step. If the chain is currently in state si then ,it moves to state sj at the next step with a probability denoted by pij, and this probability does not depend upon which states the chain was in before the current.

The probabilities pij are called transition probabilities. The process can remain in the state it is in, and this occurs with probability pii. An initial probability distribution, defined on S, specifies the starting state. Usually this is done by specifying a particular state as the starting state. A practical example of the markov chain is a frog jumping on a set of lily pads. The frog starts on one of the pads and then jumps from lily pad to lily pad with the appropriate transition probabilities.

**HMM:**

The Hidden Markov Model (HMM) is a popular statistical tool for modelling a wide range of time series data. In the context of natural language processing (NLP), HMMs have been applied with great success to problems such as part-of-speech tagging and noun-phrase chunking. Andrei Markov gave his name to the mathematical theory of Markov processes in the early twentieth century, but it was Baum and his colleagues that developed the theory of HMMs in the 1960s.

We use Expectation-Minimization (EM) Algorithm is our project. The EM algorithm is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set when the data is incomplete or has missing values. There are two main applications of the EM algorithm. The first occurs when the data indeed has missing values, due to problems with or limitations of the observation process. The second occurs when optimizing the likelihood function is analytically intractable but when the likelihood function can be simplified by assuming the existence of and values for additional but missing (or hidden) parameters. The latter application is more common in the computational pattern recognition community.

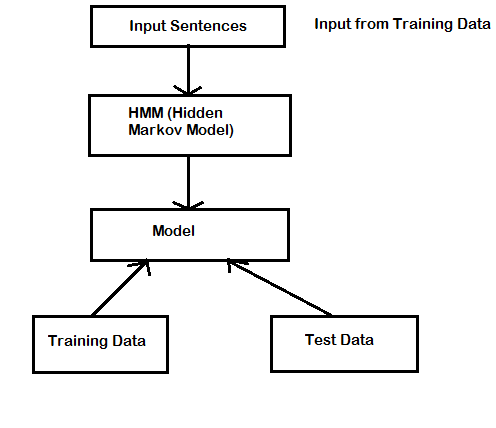
The EM algorithm is an iterative method for approximating the maximum of a likelihood function. The **maximum likelihood estimate**. It is computed in the following way:

1. We make an assumption that our population follows a certain distribution, eg: Normal with unknown parameter(s).
2. We assume that the observed data was collected through a **random sampling process**, and that **these data are independent**.
3. For each of our observed data points, we can compute ’likelihood’, given the unknown parameters. This likelihood is just the value of the corresponding density function, evaluated at that particular data point.
4. We then compute the **joint likelihood function**, which is a combination of the likelihoods for each of the data points we observed. This is computed as **the product of the likelihoods for each individual data point.**

**Eg** : If we have a sample mean of 9.7, then intuitively, a population mean of 9.7 should be 'most likely'. A population mean of 10 or 9 might still be possible, but the 'most likely' population mean, given a sample mean of 9.7 would be 9.7. This is an example of” most likely”.

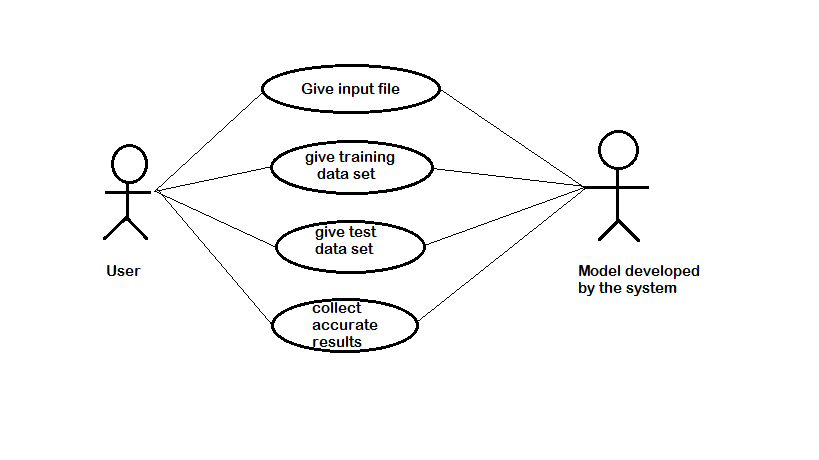
**4.2 UML DIAGRAMS**

**4.2.1 Functional Diagram**

****

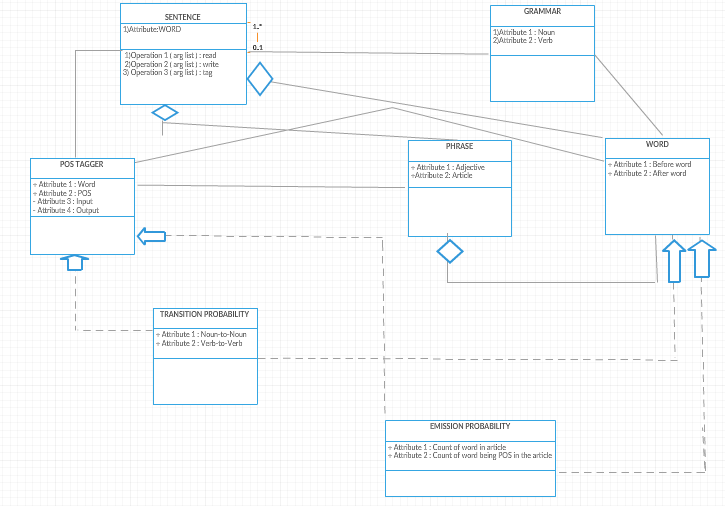
This is a very naive and general diagram for the project. It describes the functionality of the project and the series of events that happen in the project.

**4.2.2 Use Case Diagram**

****

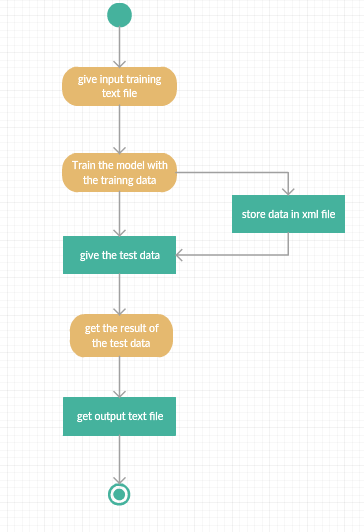
This is the use-case diagram to show that what are actions that be can be performed by the user on the model developed by the system

**4.2.3 Class Diagram**



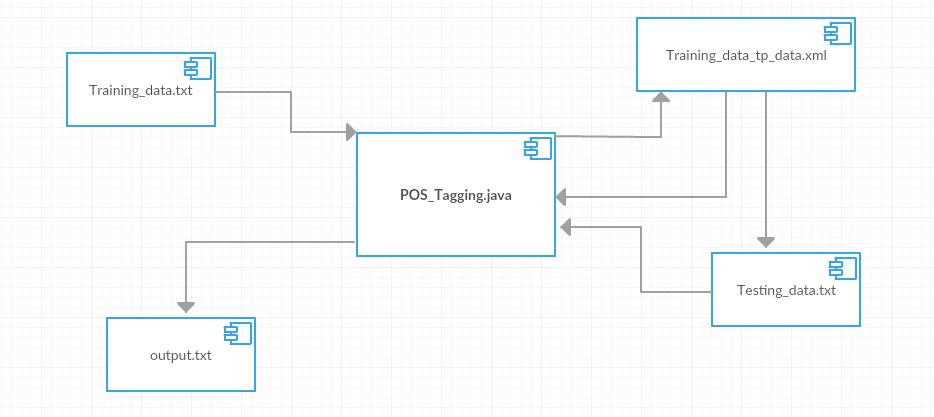
This is the class diagram to show that what are classes and its attributes and operations that be can be performed by the user on the model developed by the system.

**4.2.4 Activity Diagram**



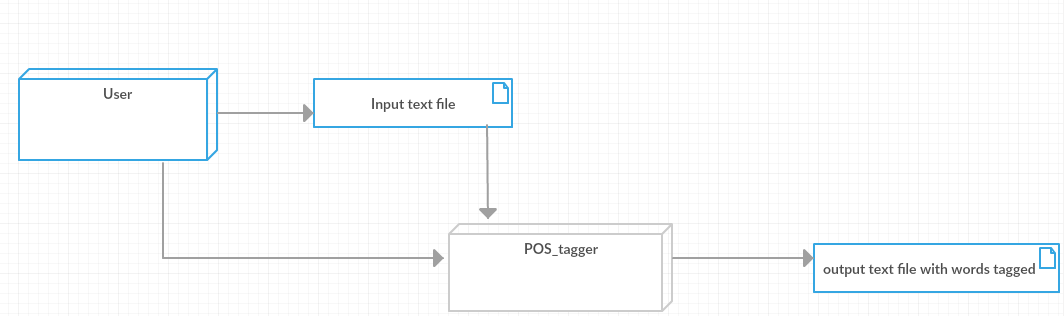
The above is the activity diagram of the project where the activity flow is shown as above.

**4.2.5 Component Diagram**

****

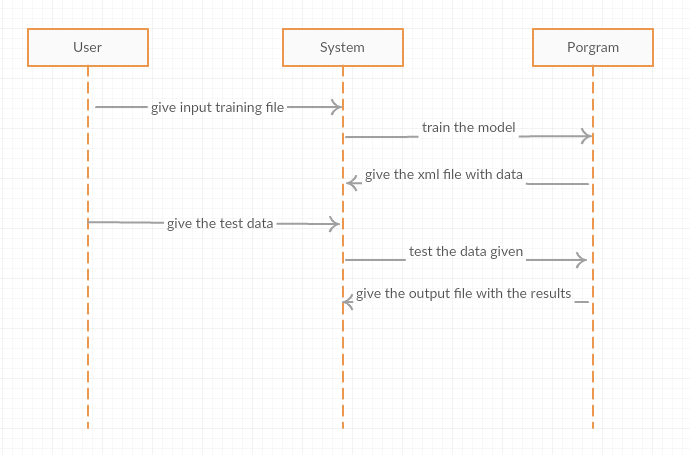
The above is the component diagram of the project where the components and their connections with each other are described and shown as above.

**4.2.6 Deployment Diagram**



The above is the deployment diagram of the project where the components and their connections with each other are described and shown as above.

**4.2.7 Sequence Diagram**



The above is the sequence diagram where the flow of the project is show from different objects at different time scenarios.

**4.3 MODULES:**

In our project we have the following modules.

* One java file which executes pos tagging.
* Two input files i.e; for training and testing data.
* One xml file which stores data about training data.
* And, one output file which gives the result.

**5. IMPLEMENTATION & RESULTS**

**5.1 EXPLANATION OF KEY FUNCTIONS**

The below is the class that holds the transitional probabilities of the word from the training data , which is as follows:

**public** **static** **class** word\_tag

{

/\*

noun-nun,

pronoun-pnu,

adverb-adv,

verb-vrb,

interjection-intj,

preposition-ppt,

adjective-adj,

conjunction-con,

article-art

others-oth

\*/

**public** **float** nun,pnu,adv,adj,con,intj,art,ppt,vrb,oth;

word\_tag()

{

nun=pnu=adv=adj=con=intj=art=ppt=vrb=oth=0;

}

}

The below is the pseudo function that retrieves the data from the xml file for each word in the testing data, which is as follows:

public static word\_tag retrive\_word\_data\_from\_xml(String words)

{

word\_tag result=new word\_tag();

try

{

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

Document doc = docBuilder.parse("C:/Users/Welcome/Desktop/POS- Tagging/Training\_Data\_tp\_data.xml");

doc.getDocumentElement().normalize();

if((int)doc.getElementsByTagName(words).getLength()==0||doc.getElementsByTagName(words).item(0).getNodeName().toString()==null)

{

return result;

}

NodeList nl1=(NodeList)doc.getElementsByTagName(words);

if(nl1.item(0).getNodeName()==null)

{

return result;

}

Element el=(Element)nl1.item(0);

NodeList nl=el.getChildNodes();

int i=0;

Node node = nl.item(i++);

result.nun=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.pnu=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.adj=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.adv=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.art=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.con=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.intj=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.ppt=Float.parseFloat(node.getTextContent());

node = nl.item(i++);

result.vrb=Float.parseFloat(node.getTextContent());

node = nl.item(i);

result.oth=Float.parseFloat(node.getTextContent());

}

catch(Exception ee)

{

System.out.println("exceprion is retrive\_word\_data\_from\_xml: "+ee);

}

return result;

}

The below function is the main function used to calculate the transition probability of the word from the training data, which is given as follows:

**public** **static** **void** calculate\_count\_n\_tp(**int** n)

{

**int** i;

**for**(i=0;i<n;i++)

{

write\_word\_in\_xml(data\_block[i]);

**if**(i<n-1)

Transitional\_probability(data\_block[i].value,data\_block[i+1].value);

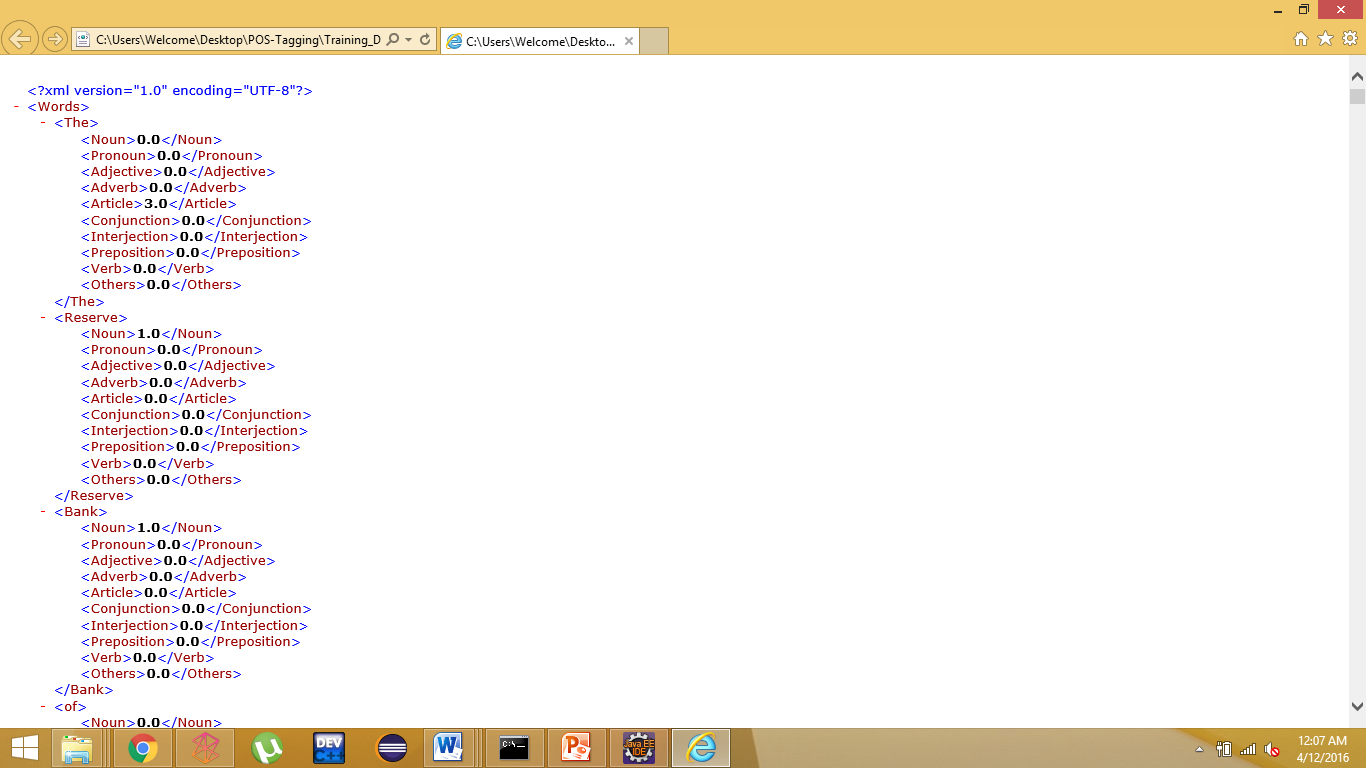
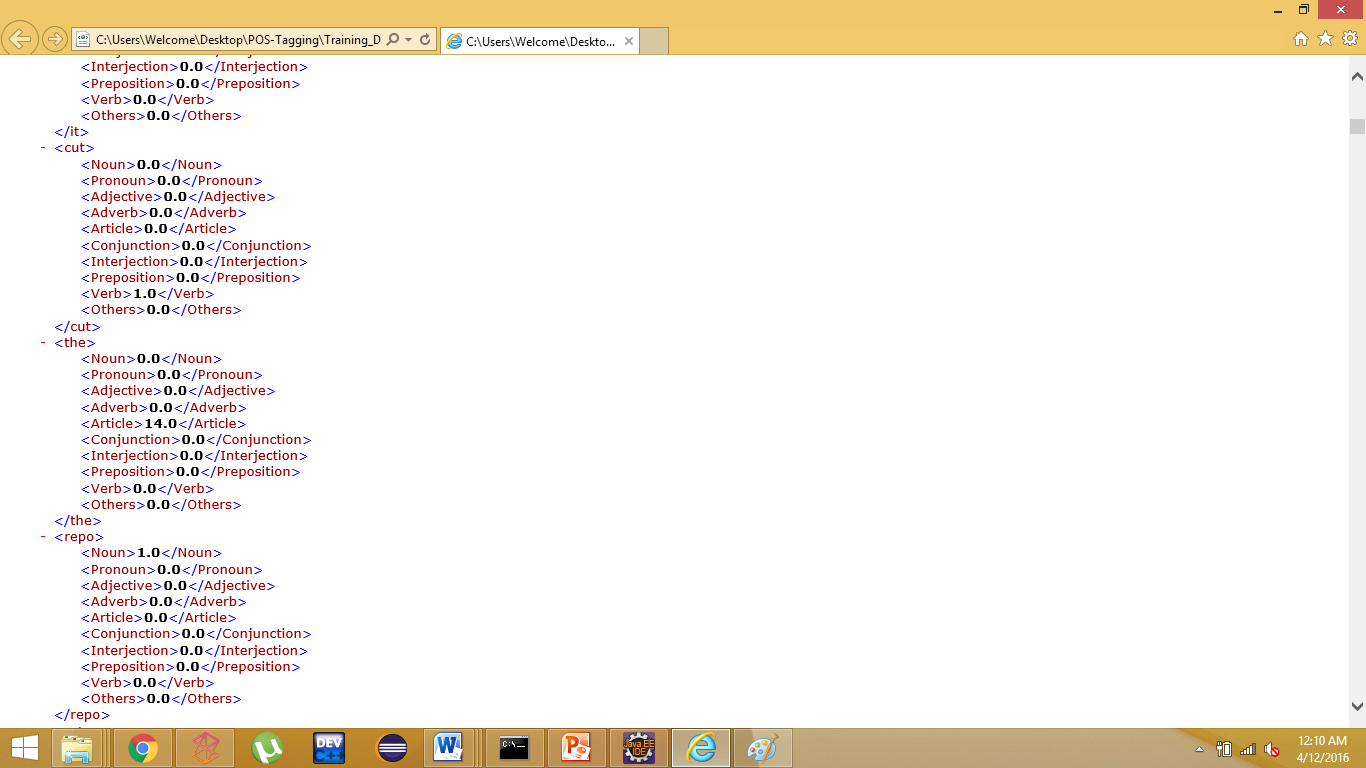
}

}

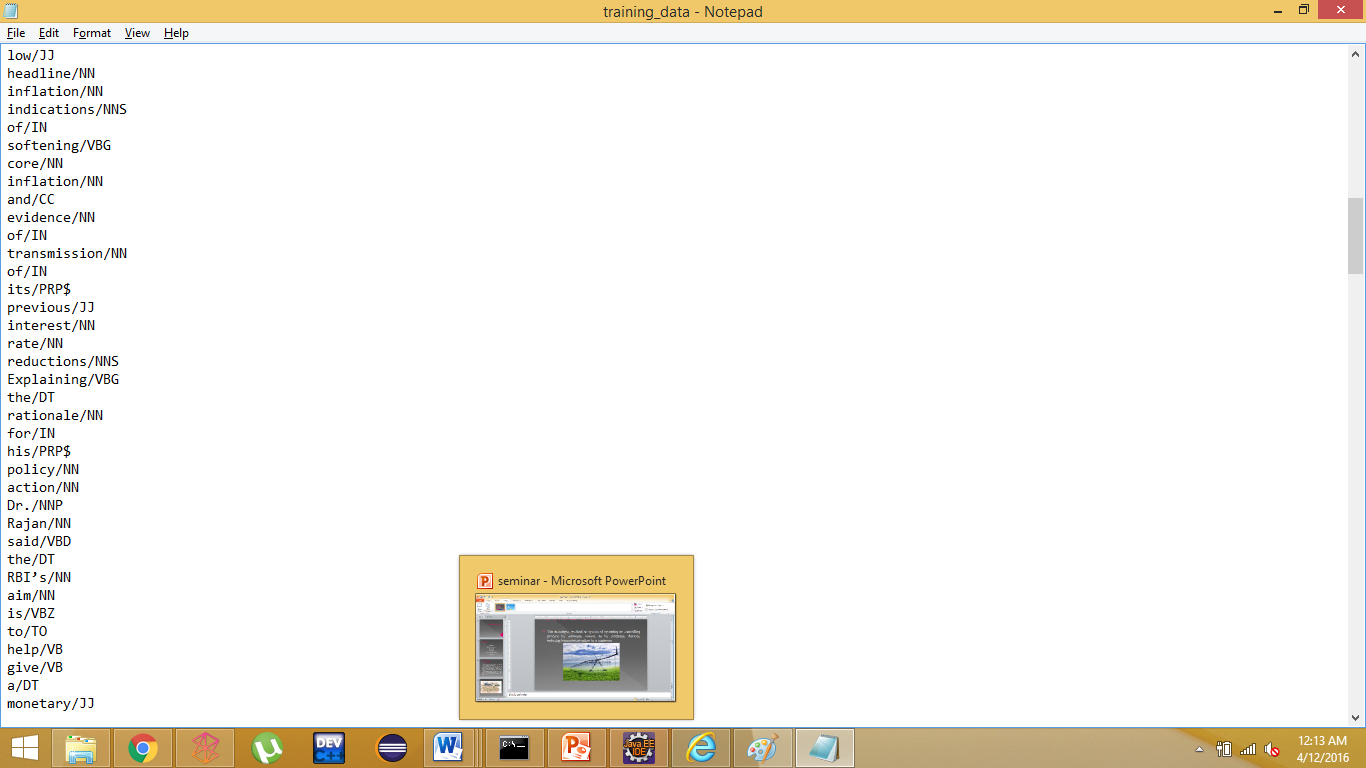
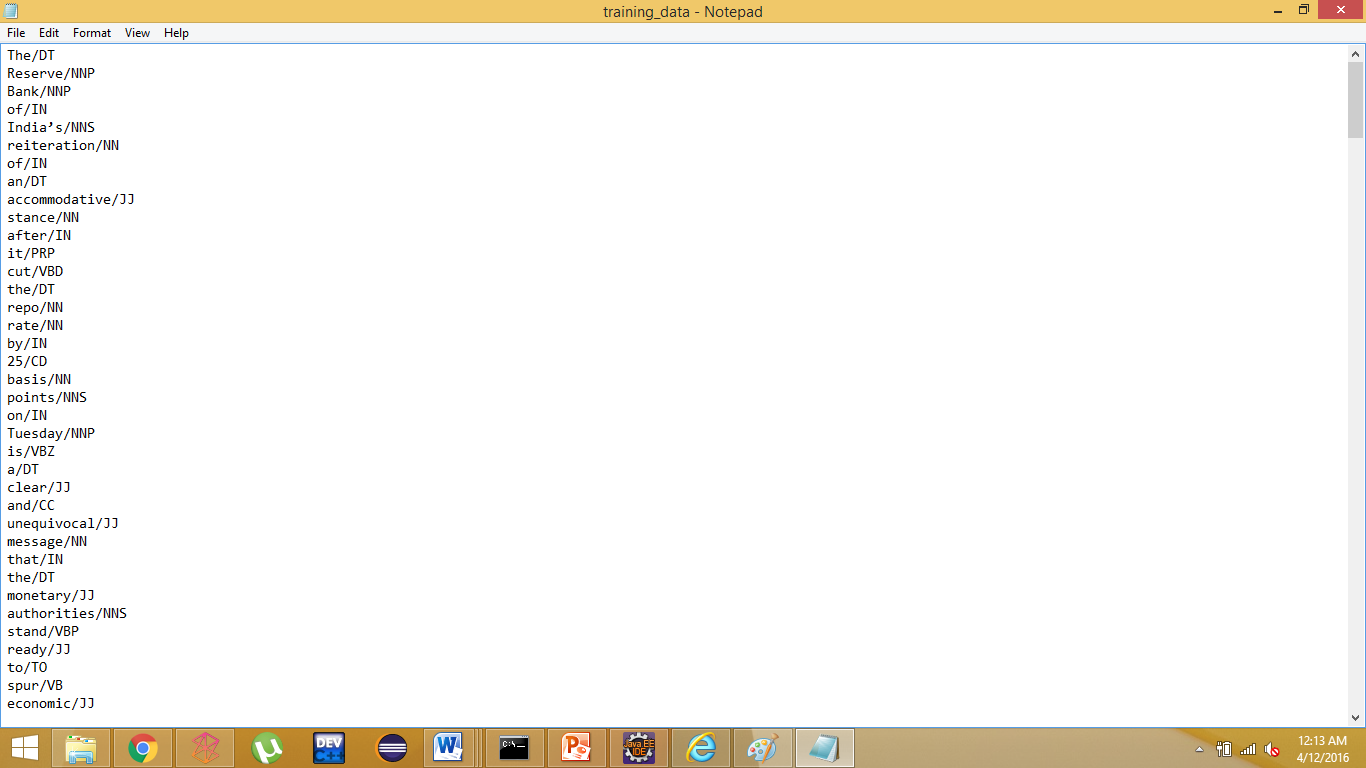
**5.2 METHOD OF IMPLEMENTATION**

**5.2.1 Output Screens**

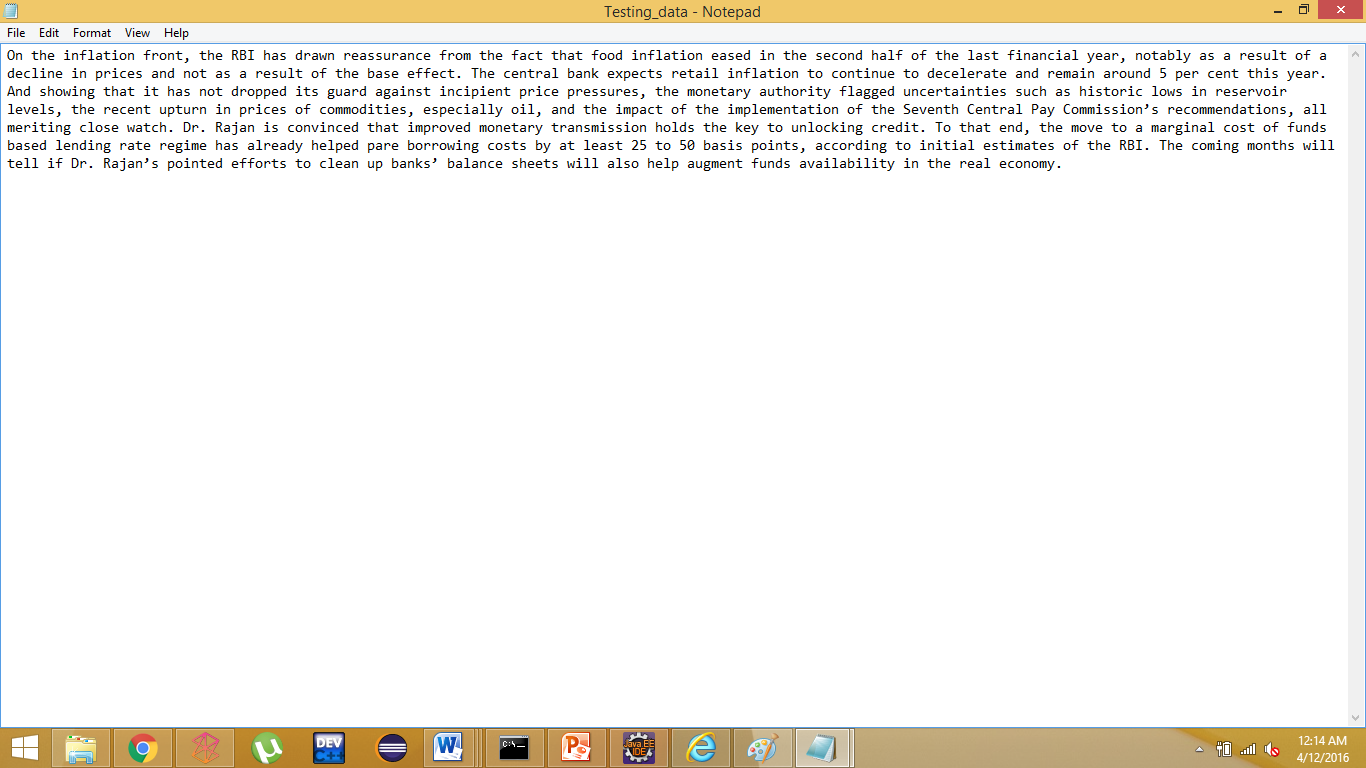
The below is the screen shot of the xml file generated after the training data is read and used to train the model.



The below are the screen shots of the training data that is given to the model to train.

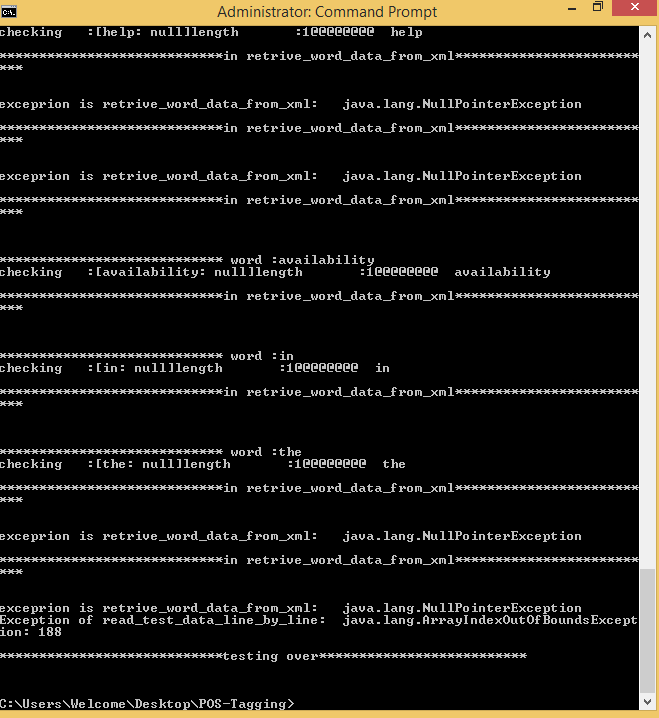


The below are the screen shots of the testing data that is given to the model to test.

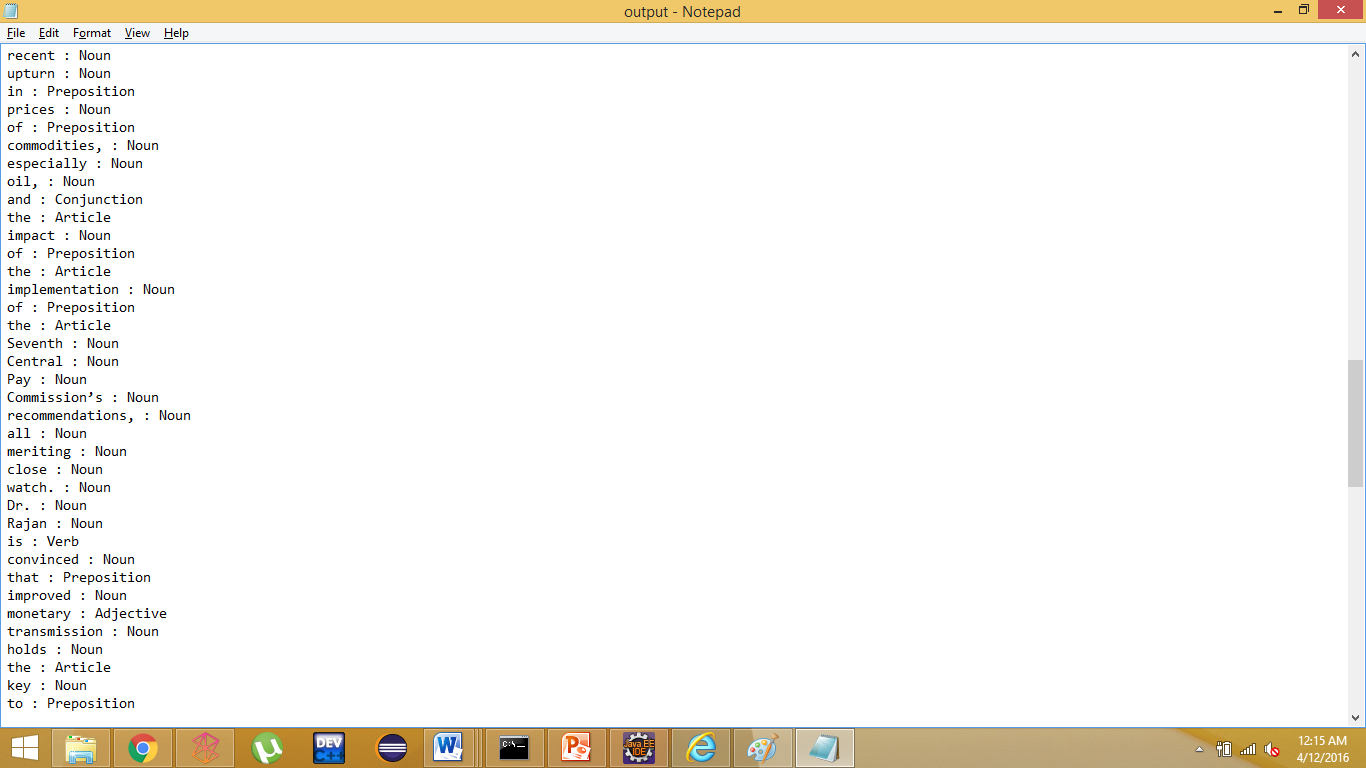
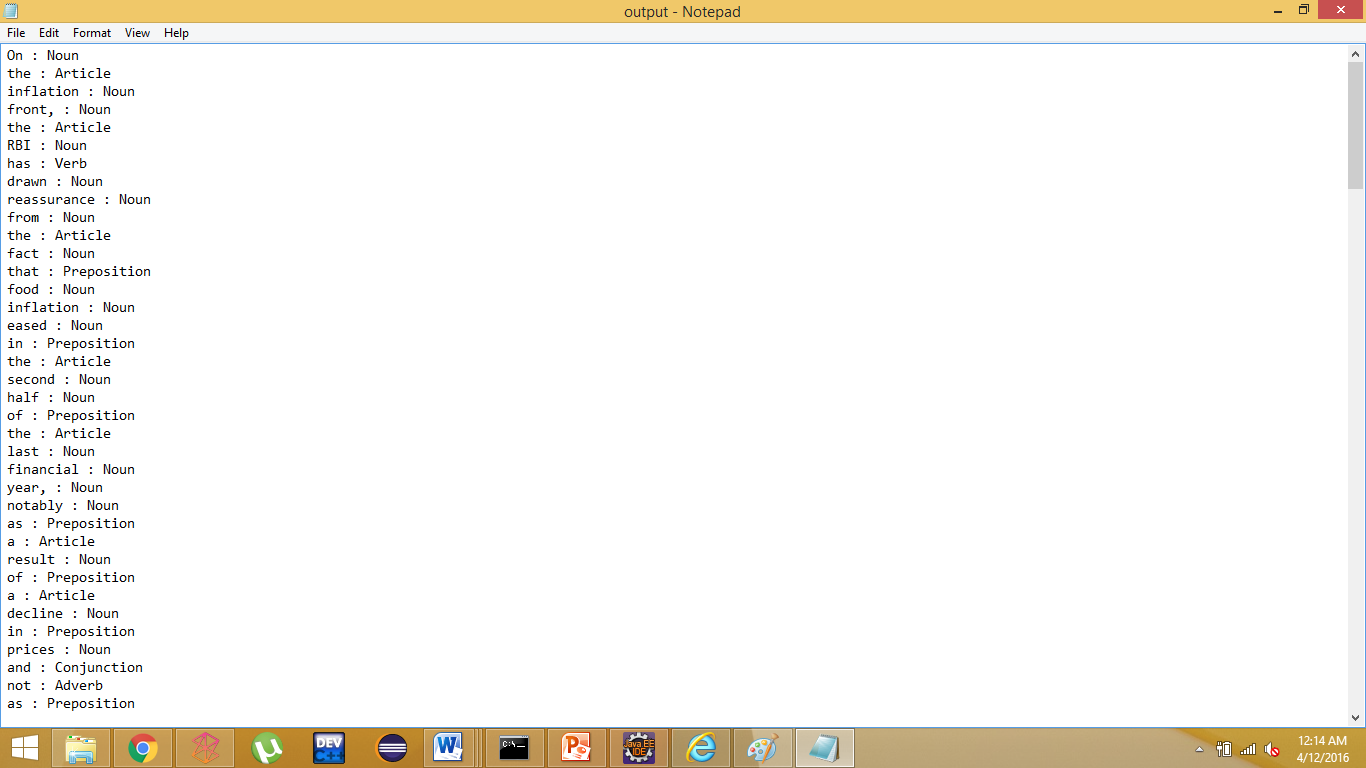


The below are the screen shots of the cmd output screen after the program is completely compiled and executed.

.



The final output file/text document with the parts of speech tags are shown in the following screen shots:



**6. TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the [software](https://en.wikipedia.org/wiki/Software) to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding [software bugs](https://en.wikipedia.org/wiki/Software_bug) (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* meets the requirements that guided its design and development,
* responds correctly to all kinds of inputs,
* performs its functions within an acceptable time,
* is sufficiently usable,
* can be installed and run in its intended [environments](https://en.wikipedia.org/wiki/Operating_environment), and
* Achieves the general result its stake holders desire.

**6.1 TYPES OF SOFTWARE TESTING:**

**Black Box Testing**

Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

**White Box Testing**

This testing is based on knowledge of the internal logic of an application’s code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**Unit Testing**

Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. May require developing test driver modules or test harnesses.

**Incremental Integration Testing**

Bottom up approach for testing i.e continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.

**Integration Testing**

Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

**Functional Testing**

This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

**System Testing**

Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

**End-to-end Testing**

Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

**Performance Testing**

Term often used interchangeably with ‘stress’ and ‘load’ testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

**Usability Testing**

User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

|  |  |  |
| --- | --- | --- |
| **TEST CASE** | **ERROR** | **RESULT** |
| When the word in the training data is also present in the testing data | NO ERROR | PASS |
| Number Input is given | EXCEPTION | PASS |
| Words with special character | EXCEPTION | PASS |
| Words which are new and appear in the testing data | OTHERS | PASS |
| Bad Word | EXCEPTION | PASS |

**6.2 DESIGN OF TEST CASES AND SCENARIOS**

**7. CONCLUSION AND FUTURE WORK**

We have presented a method for unsupervised parts-of-speech tagging that considers a word type and its allowed POS tags as a primary element of the model. This departure from the traditional token-based tagging approach allows us to explicitly capture typelevel distributional properties of valid POS tag assignments as part of the model. The resulting model is compact, efficiently learnable and linguistically expressive.

Our empirical results demonstrate that the type-based tagger rivals state-of-the-art tag-level taggers which employ more sophisticated learning mechanisms to exploit similar constraints. In this paper, we make a simplifying assumption of one-tag-per-word. This assumption, however, is not inherent to type-based tagging models. A promising direction for future work is to explicitly model a distribution over tags for each word type. We hypothesize that modeling morphological information will greatly constrain the set of possible tags, thereby further refining the representation of the tag lexicon.

In the future, one may try using the Expectation Maximization Algorithm to calculate the optimal weights for the interpolation between higher order and lower order probabilities. One may also try to use the Expectation Maximization Algorithm to evaluate the optimal discounting values for the probabilities. Or one may even try to use a different probability smoothing scheme altogether. Finally, one may try to extend the Hidden Markov Model even further by looking at the previous three tags. Nevertheless, we are not hopeful for the last approach since our Extended Hidden Markov Model performed equally or slightly worse than our Interpolation Hidden Markov Model.

**REFERENCES**

* Wikipedia.com.
* Slideshare.com
* Different sources over the internet.
* Gallagher, Sean. ["Memory that never forgets: non-volatile DIMMs hit the market"](http://arstechnica.com/information-technology/2013/04/memory-that-never-forgets-non-volatile-dimms-hit-the-market/). Ars Technica.
* [Jump up^](https://en.wikipedia.org/wiki/Random-access_memory#cite_ref-2) Bellis, Mary. ["The Invention of the Intel 1103"](http://inventors.about.com/library/weekly/aa100898.htm).
* [Jump up^](https://en.wikipedia.org/wiki/Random-access_memory#cite_ref-3) ["IBM Archives -- FAQ's for Products and Services"](http://www-03.ibm.com/ibm/history/reference/faq_0000000011.html). ibm.com.
* [Jump up^](https://en.wikipedia.org/wiki/Random-access_memory#cite_ref-4) Napper, Brian, [Computer 50: The University of Manchester Celebrates the Birth of the Modern Computer](http://www.computer50.org/), retrieved 26 May 2012
* [Jump up^](https://en.wikipedia.org/wiki/Random-access_memory#cite_ref-5) Williams, F.C.; Kilburn, T. (Sep 1948), "Electronic Digital Computers", Nature 162 (4117): 487, [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1038/162487a0](https://dx.doi.org/10.1038%2F162487a0). Reprinted in The Origins of Digital Computers.
* [Definition of: flash ROM, PC Magazine.](http://www.pcmag.com/encyclopedia_term/0,,t=&i=43279,00.asp)
* [Jump up^](https://en.wikipedia.org/wiki/Read-only_memory#cite_ref-2) ["History of Computation - Babbage, Boole, Hollerith"](http://cgi.csc.liv.ac.uk/~ped/teachadmin/histsci/htmlform/lect4.html). [Paul E. Dunne](https://en.wikipedia.org/w/index.php?title=Paul_E._Dunne&action=edit&redlink=1).
* [Jump up^](https://en.wikipedia.org/wiki/Read-only_memory#cite_ref-3) See page 6 of Toshiba's 1993 [NAND Flash Applications Design Guide](http://www.data-io.com/pdf/NAND/Toshiba/NandDesignGuide.pdf.pdf).