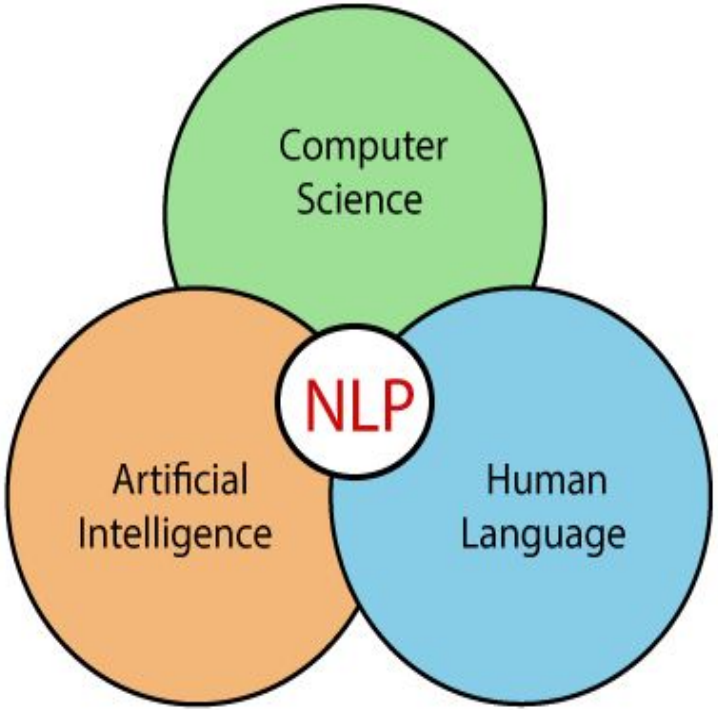
Natural Language Processing

BCA SEM-6 (Gujarat University)

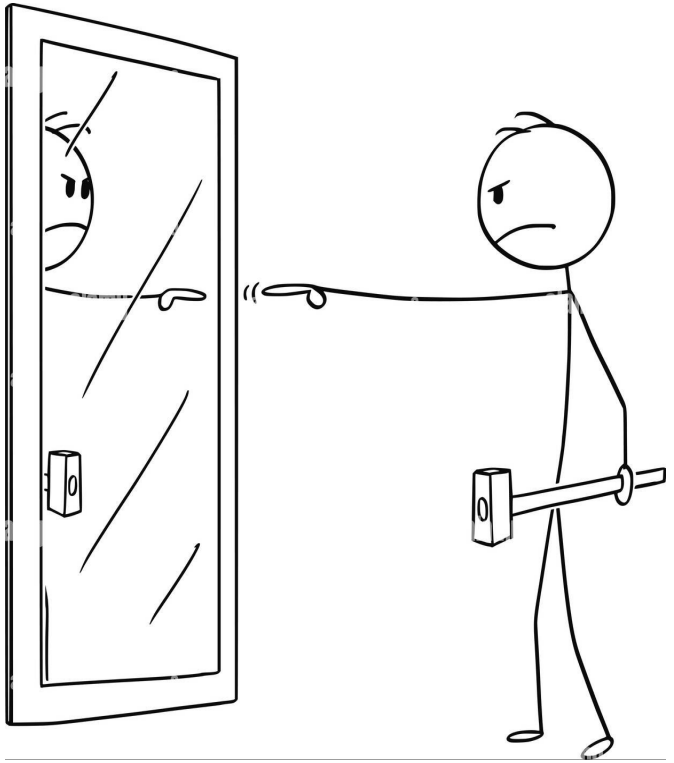
Natural Language Processing

- NLP stands for **Natural Language Processing**, which is a part of **Computer Science, Human language,** and **Artificial Intelligence**.
- It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages.
- It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction,** and **topic segmentation**.



Case Grammar

- Case Grammar was developed by **Linguist Charles J. Fillmore** in the year 1968. **Case Grammar uses languages such as English to express the relationship between nouns and verbs by using the preposition.**
- In Case Grammar, case roles can be defined to link certain kinds of verbs and objects.
- **For example: "Neha broke the mirror with the hammer".** In this example case grammar identify **Neha as an agent**, **mirror as a theme**, and **hammer as an instrument.**



Advantages of NLP

- NLP helps users to **ask questions** about any subject and **get a direct response within seconds.**
- NLP offers **exact answers** to the question means it **does not offer unnecessary and unwanted information.**
- NLP helps **computers to communicate with humans in their languages.**
- **It is very time efficient.**

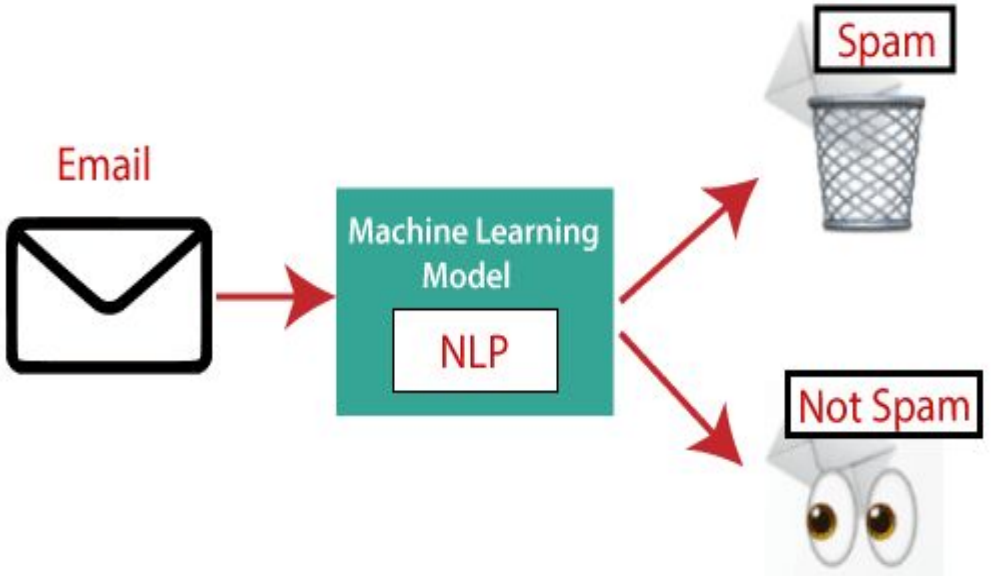
Disadvantages of NLP

- NLP **may not show context.**
- NLP is **unpredictable**
- NLP may **require more keystrokes.**
- NLP **is unable to adapt to the new domain, and it has a limited function** that's why NLP is built for a single and specific task only.

Applications of NLP

- **Question Answering**

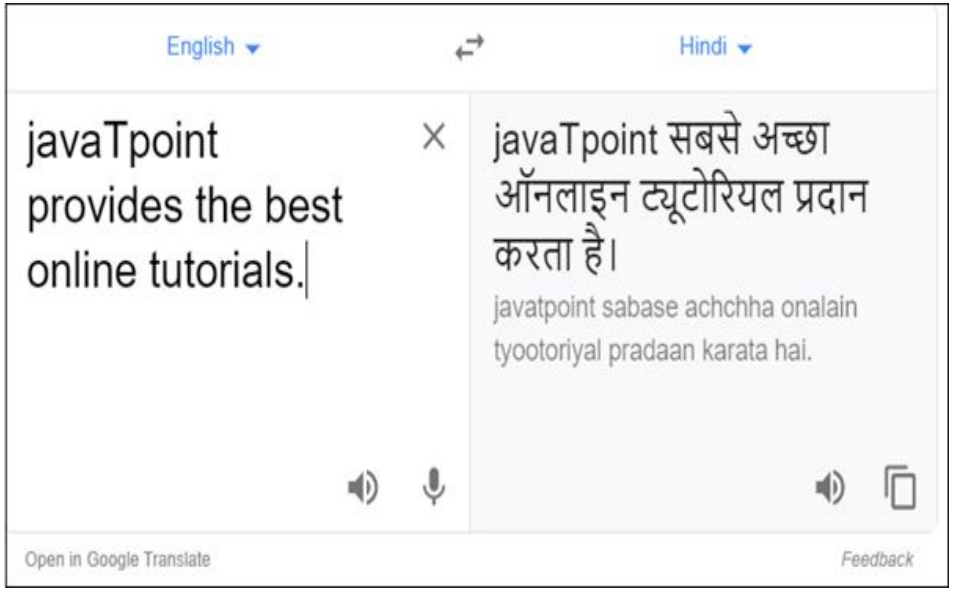
- **Spam Detection**



Applications of NLP

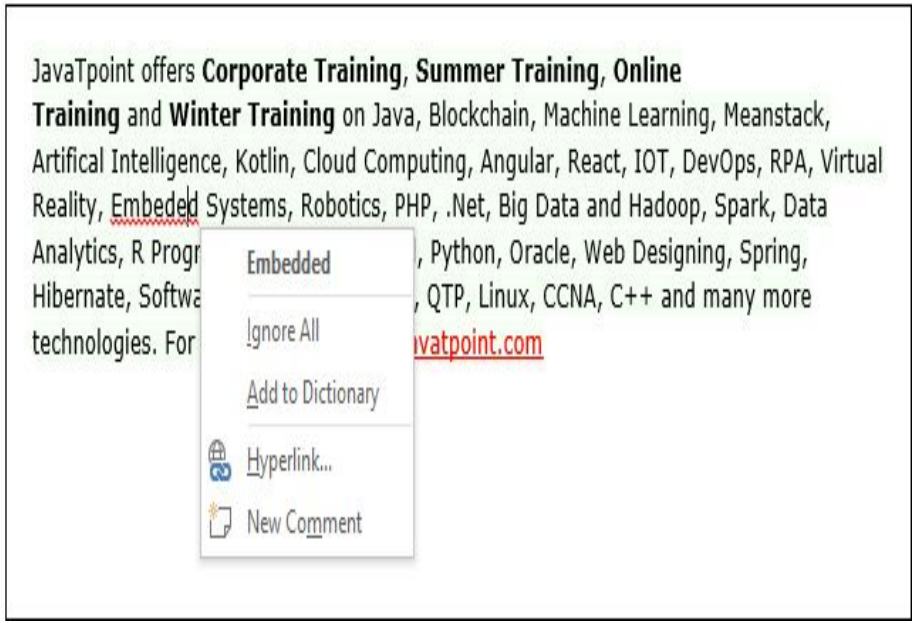
- **Sentiment Analysis**

- **Machine Translation**

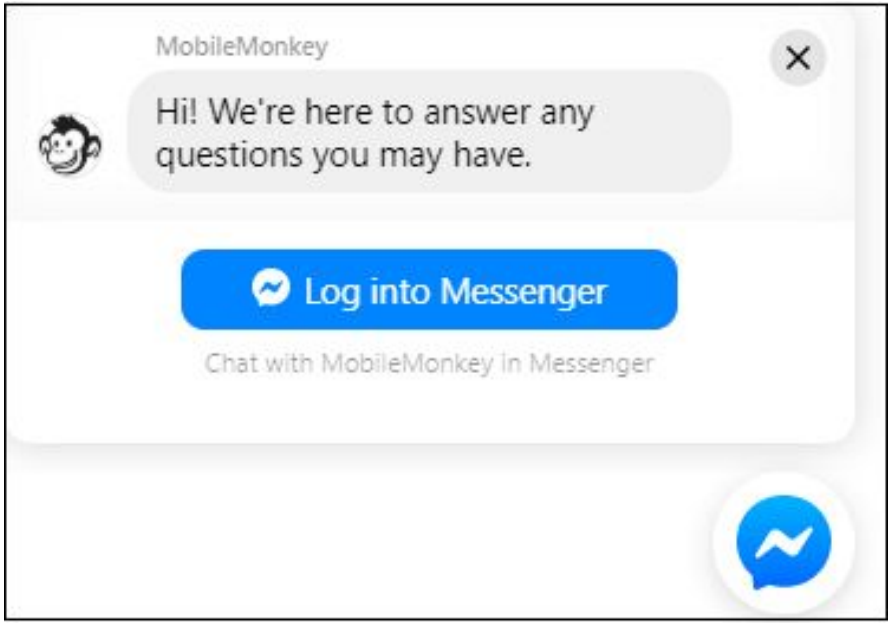


Applications of NLP

- **Spelling correction**



- **Chatbot**



Language Models

- **Formal languages**, such as the programming languages **Java or Python have exactly defined language models.**

- **A language can be defined as a collection of strings;** "**print (2 + 2)**" is a legal program in the language Python, whereas "**2) + (2 print**" is not.

- Since **there are an infinite number of legal programs, they cannot be enumerated; instead they are such that by a collection of rules called a grammar.**

Language Models

- Formal languages **also have rules that define the meaning or semantics of a program;** as an example, the rules say that the **"meaning" of "2+2" is 4**, and the **meaning of "1/0" is that an error** is signaled.

- **Natural languages are also ambiguous. We cannot speak of a single meaning for a sentence, but rather of a probability distribution over possible meanings.**

N Gram Character Models

An n-gram model **is a technique of counting sequences of characters or words that allows us to support rich pattern discovery in text.**

In other words, **it tries to capture patterns of sequences** (characters or words next to each other) **while being sensitive to contextual relations** (characters or words near each other).

Ultimately, **a written text is consist of characters-letters, digits, punctuation, and spaces in English** (and additional exotic characters in some other languages).

Thus, **one of the simplest language models is a probability distribution over sequences of characters.**

N Gram Character Models

unigram

bigram

trigram

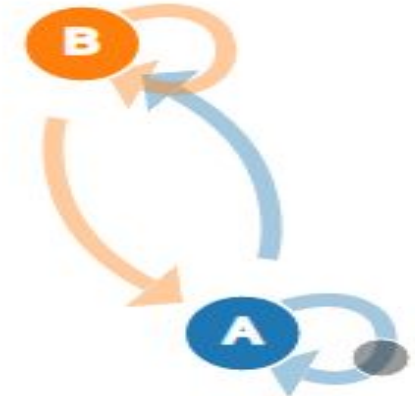
n-gram (n = 4)

N Gram Character Models

A sequence of **written symbols of length n is called an n-gram**, with special case **"unigram" for 1-gram**, **"bigram" for 2-gram**, and **"trigram" for 3-gram**. **A model of the probability distribution of n-letter sequences is thus called an n-gram model.**

A Markov chain or Markov process is **a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event**.

With two states **(A and B) in our state space,**
there are 4 possible transitions
(not 2, because a state can transition back into itself).
If we're at 'A' we could transition to 'B' or stay at 'A'.
If we're at 'B' we could transition to 'A' or stay at 'B'.



N Gram Character Models

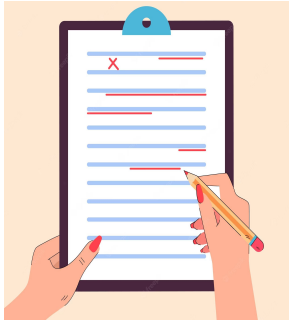
What can we do with n-gram character models?

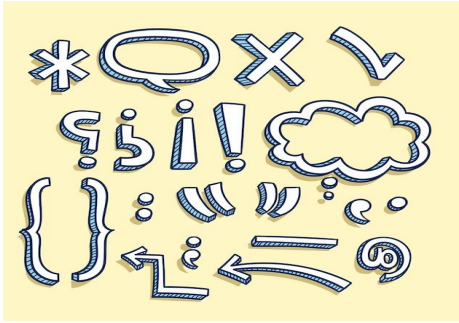
One task for which they are well suited is language identification: **given a text, determine what natural language it is written in.**

This is a relatively easy task; even with short texts such as **"Hello, world"** or **"Wie Geht's Dir,"** it is easy to identify the **first as English** and the **second as German.**

Computer systems **identify languages with greater than 99% accuracy; sometimes, closely related languages, such as Swedish and Norwegian are confused.**

N Gram Character Models



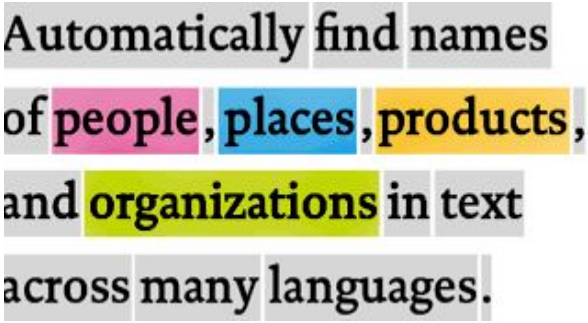


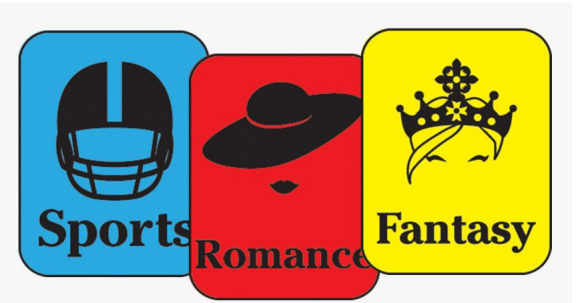
Other tasks for character models **include spelling correction, genre classification,** and named-entity recognition.

Genre classification means deciding **if a text is a news story, a legal document, scientific article, etc.**

While **many features help make this classification,counts of punctuation** and other character n-gram features go a long way.

Named-entity recognition is the task of finding names of things in a document and deciding what class they belong to.

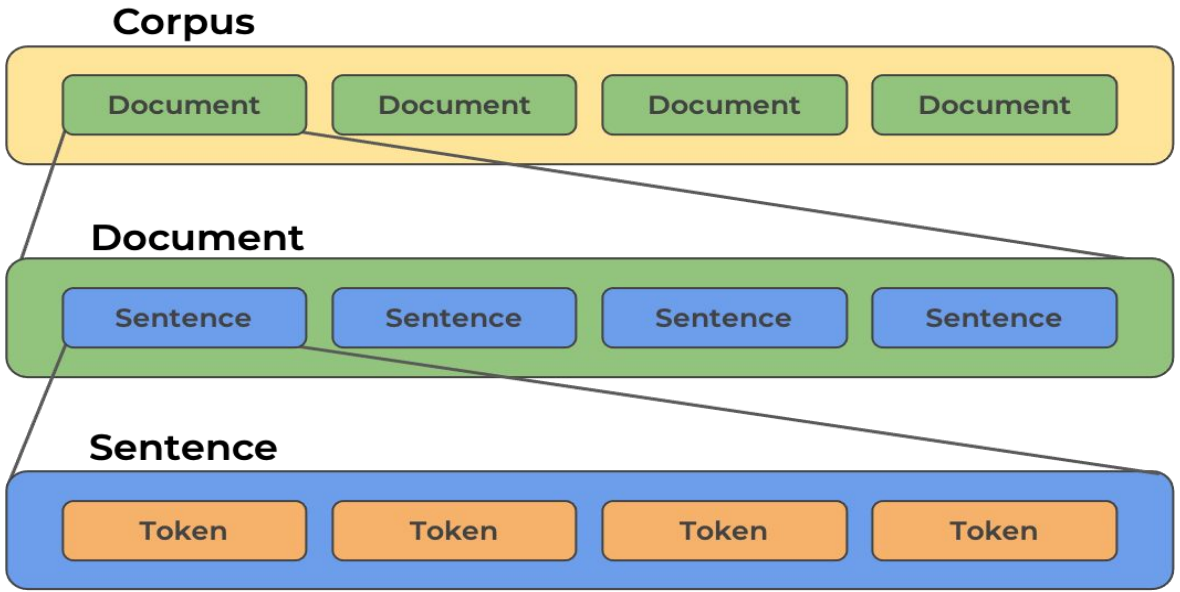




N Gram Character Models

For example, in the text "**Mr. Sopersteen was prescribed aciphex.**" we should recognize that "**Mr. Sopersteen**" is the name of a person and "**aciphex**" is the name of a drug.

A corpus is **a collection of authentic text or audio organized into datasets**. Authentic here means text written or audio spoken by a native of the language or dialect. **A corpus can be made up of everything from newspapers, novels, recipes, radio broadcasts to television shows, movies, and tweets.**



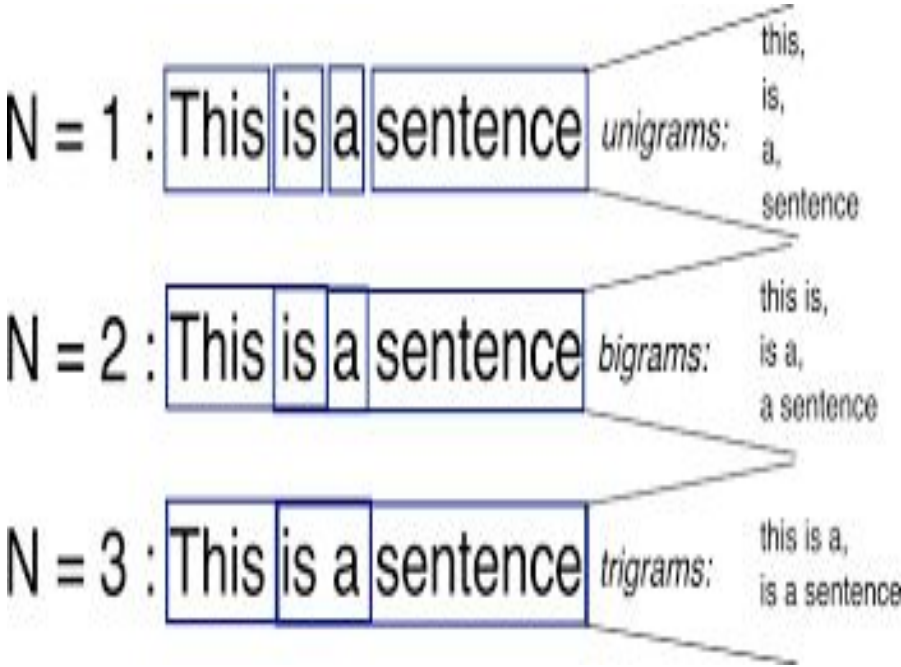
N Gram Word Models

When we have a tendency **to analyze a sentence one word at a time,** then it is referred to as a **unigram.**

The sentence parsed two words at a time is a bigram.

When **the sentence is parsed three words at a time, then it is a trigram.**

Similarly, **n-gram refers to the parsing of n words at a time.**



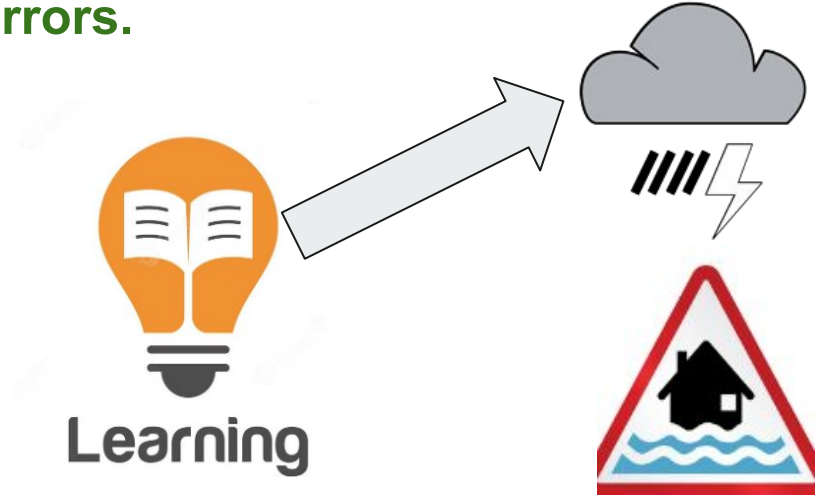
N Gram Word Models

This is Big Data AI Book

Uni-Gram	This	Is	Big	Data	AI	Book
Bi-Gram	This is	Is Big	Big Data	Data AI	AI Book	
Tri-Gram	This is Big	Is Big Data	Big Data AI	Data AI Book		

N Gram Word Models

Parsing allows machines to **understand the individual meaning of a word** in a sentence.
Also, this type of **parsing helps predict the next word and correct spelling errors.**



Consider two sentences:
"There was heavy rain" vs. **"There was heavy flood".**

From experience, we all know that the **previous sentence sounds better.**

An N-gram model will tell us that **"heavy rain" occurs much more often than "heavy flood"** within the training corpus.

Thus, **the first sentence is more probable and can be elected by the model.**

N Gram Word Models

An n-gram model for the above example would calculate the following probability:

P('There was heavy rain') = P('There', 'was', 'heavy', 'rain') = P('There')P('was' "There')P('heavy "There was')P('rain"There was heavy')

Since **it's impractical to calculate these conditional probabilities**, using Markov assumption, we approximate this to a bigram model:

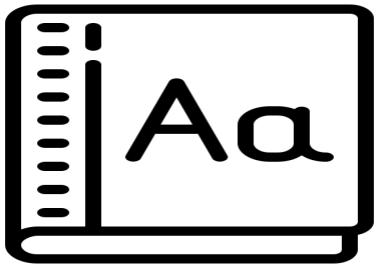
P("There was heavy rain')~P("There')P('was' 'There')P('heavy"was')P('rain" "heavy')

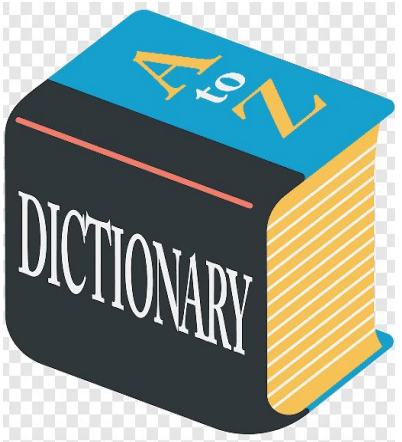
N Gram Word Models vs Character Models

The main difference is that the **vocabulary** the set of symbols that make up the corpus and the model is larger.

There are only about **100 characters in most languages**, and sometimes we build character models that are **even more restrictive**, for example by treating "A" and "a" as the same symbol or by treating all punctuation as the same symbol.

But with word models we have
at least tens of thousands of symbols,
and **sometimes millions.**





N Gram Word Models

The wide range is because it is not clear what constitutes a word.

In **English a sequence of letters enclosed by spaces is a word**, but in some languages,**like Chinese, words are not separated by spaces**, and even in English many decisions must be made to have a clear policy on word boundaries:

How many words are in
"ne'er-do-well"?
Orin "(Tel:1-800-960-5660x123)"?

Text Classification

Text classification **is the process of classifying documents into predefined categories based on their content.**

It is the **machine controlled (automated) assignment of natural language texts to predefined classes (or categories).**

Text classification is the primary requirement of **text retrieval systems** that **retrieve texts in response to a user query**, and **text understanding systems**, which **transform text in some way such as producing summaries, questions-answers or extract data**.

Text Classification

Text classification is that **the method of (process of) classifying documents into predefined categories** by their content.

Language identification and Genre classification are examples of text classification, as is **sentiment analysis(classifying a movie or product review as positive or negative)** and **spam detection (classifying an email message as spam or not-spam).**

Since **"not-spam"** is awkward, **researchers have contained the term ham for not-spam.**

Text Classification

We can treat **spam detection as a problem in supervised learning.**

A training set is readily available: **the positive (spam) examples are in my spam folder**, **the negative (ham) examples are in my inbox.**

Here is an example:

Spam: Wholesale Fashion Watches -57% today. Designer watches **for cheap...**
Spam: **You can buy** ParacetamolFr$1.85 All Medications at unbeatable prices!...
Spam: **WE CAN TREAT ANYTHING YOU SUFFER FROM JUST TRUST US...**
Spam: **Sta.rt earn*ing the salary yo,u d-eserve by o'btaining the prope,rcrede'ntials!**

Text Classification

Ham: The practical significance of hyper tree width in identifying more...
Ham: Abstract: We will motivate the problem of social identity clustering: ...
Ham: Good to see you my friend. Hey Peter, It was good to hear from you. ...
Ham: PDS implies convexity of the resulting optimization problem (Kernel Ridge …

A character model should detect this.
We could either create a full character n gram model of spam and ham, or **we could handcraft features such as "number of punctuation marks embedded in words."**

Text Classification

Note that we have two complementary ways in which of talking about classification. In the language-modeling approach, we define **One n-gram language model** for **P(Message | spam)**by training on the spam folder, and one model for **P(Message | ham)** by training on the inbox.

Then we can classify a new message with an **Application of Bayes' rule:**

$$\underset{c \in \{spam, ham\}}{\operatorname{argmax}} P(c \mid message) = \underset{c \in \{spam, ham\}}{\operatorname{argmax}} P(message \mid c) P(c) .$$

Text Classification

Where,**P(c) is estimated just by counting the total number of spam and ham messages. This approach works well for spam detection, just as it did for language identification.**

The data can accurately determine if it is good or not. It is necessary to constantly update features, because spam detection is an adversarial task; the spammers modify their spam in response to the spam detector changes.

It can be expensive to run algorithms on a very large feature vector, **so often a process of feature selection is used to keep only the features that best discriminate between spam and ham.**

Classification by Data Compression

Another way to think about **classification loss less Compression algorithm takes a sequence of symbols, detects repeated patterns in it**, **and writes a description of the sequence that is lot of compact than the original.**

For example, the text **"0.142857142857142857"** might be compressed to **"0.[142857]*3."**

Compression algorithms **work by dictionaries of subsequences** of the text, and **then one dictionary entry, referring to entries within the dictionary.**

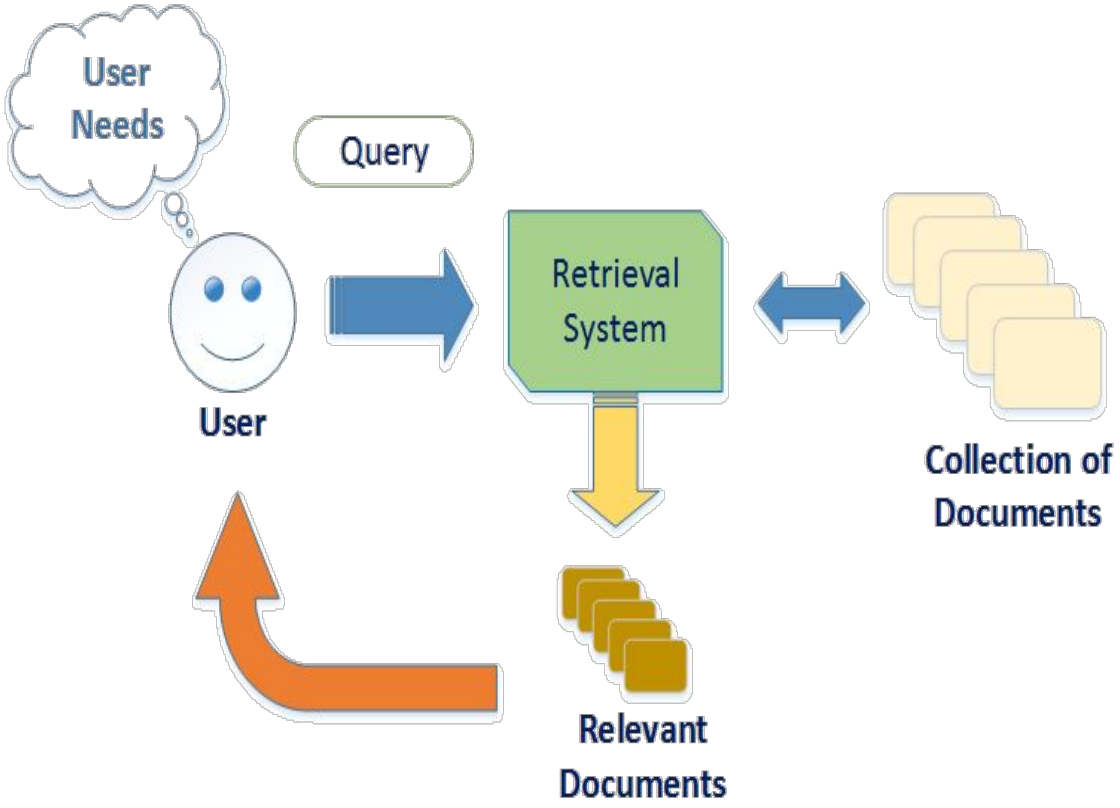
The example here had only one "142857."

Information Retrieval

Information retrieval **is the task of finding documents that are relevant to a user's need for information.**

The well-known examples of information retrieval systems **are search engines on the World Wide Web.**

A Web user can type a query such as: **AI book** into a search engine and see a list of relevant pages.



Information Retrieval

An information retrieval system can be characterized by

1. A corpus of documents. Every system should decide what it needs to treat as a document: **A paragraph, a page, or a multi page text.**

2. Queries posed in a query language. **A query specifies what the user needs to understand.**

The query language will be just a list of words, such as **[AI book]**; or it can specify a phrase of words that has to be adjacent, as in **["AI book"];** it will contain Booleanoperators as in **[AI AND book];** it can include non-Boolean operators such as **[AI NEAR book]** or **[AI book site: www.aaai.org].**

Information Retrieval

3. A result set. **This is the subset of documents that the IR system judges to be relevant to the query.** By relevant, **we tend to mean likely to be of use to the person who posed the query, for the particular information need expressed in the query.**

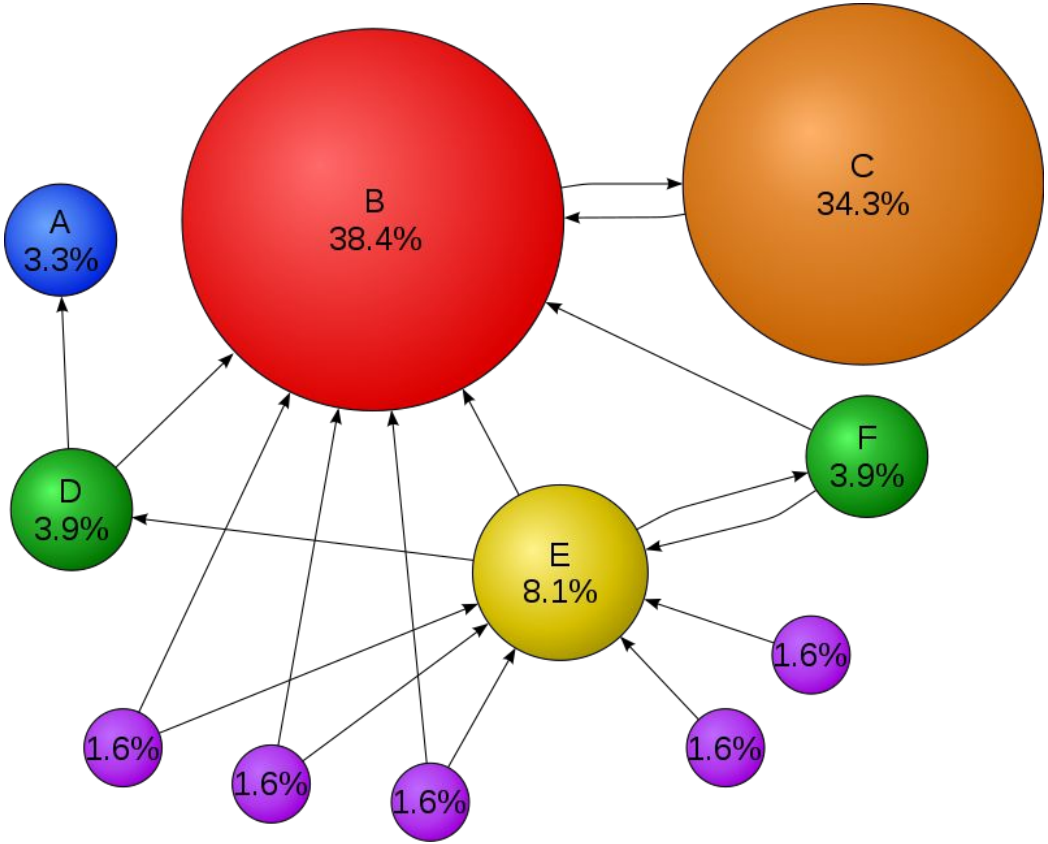
4. A presentation of the result set. **This will be as simple as a ranked list of document titles** or **as advanced as a rotating color map of the result set projected onto a three dimensional space, rendered as a two-dimensional display.**

The Page Rank Algorithm

It is a **scoring measure based only on the link structure of web pages or website.**

A web page **is important if it is pointed to by other important web pages.**

Our **first technique for link analysis assigns to every node in the web graph a numerical score between 0 and 1 known as it page rank.**



The Page Rank Algorithm

PageRank was one of the two original ideas that set **Google's search except from other Web Search engines when it was introduced in 1997.**

Larry Page

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, **with the purpose of "measuring" its relative importance within the set.**

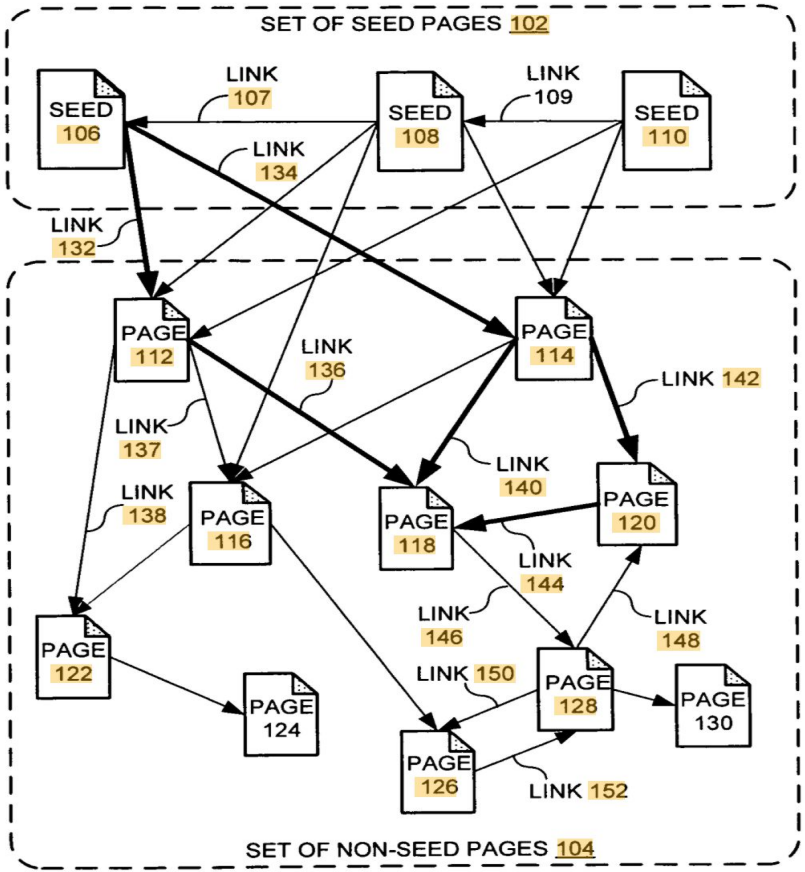
The algorithm may be applied to any collection of entities with reciprocal quotations and references.

The numerical weight that it assigns to any given element *E* is referred to as the *PageRank of E* and denoted by

$$PR(E).$$

The Page Rank Algorithm

- **A PageRank results from a mathematical algorithm based on the webgraph,** created by all World Wide **Web pages as nodes** and **hyperlinks as edges**, taking into consideration **authority hubs** such as cnn.com or mayoclinic.org.
- **The rank value indicates an importance of a particular page.** A hyperlink to a page counts as a vote of support.
- The **PageRank of a page is defined recursively** and **depends on the number and PageRank metric of all pages that link to it ("incoming links").**
- **A page that is linked to by many pages with high PageRank receives a high rank itself.**



The Page Rank Algorithm

- When calculating PageRank, pages with **no outbound links are assumed to link out to all other pages in the collection.**
- **Their PageRank scores are therefore divided evenly among all other pages.**
- In other words, **to be fair with pages that are not sinks, these random transitions are added to all nodes in the Web.**

- **This residual probability, *d*, is usually set to 0.85, estimated from the frequency that an average surfer uses his or her browser's bookmark feature.** So, the equation is as follows:

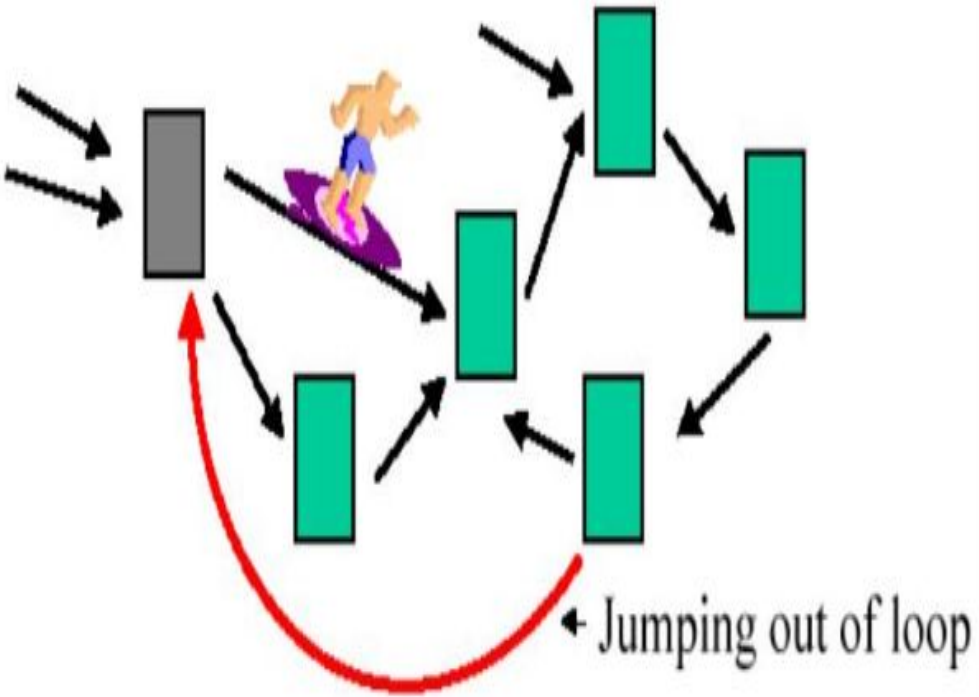
$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_2, \ldots, p_N$ are the pages under consideration, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outbound links on page p_j, and N is the total number of pages.

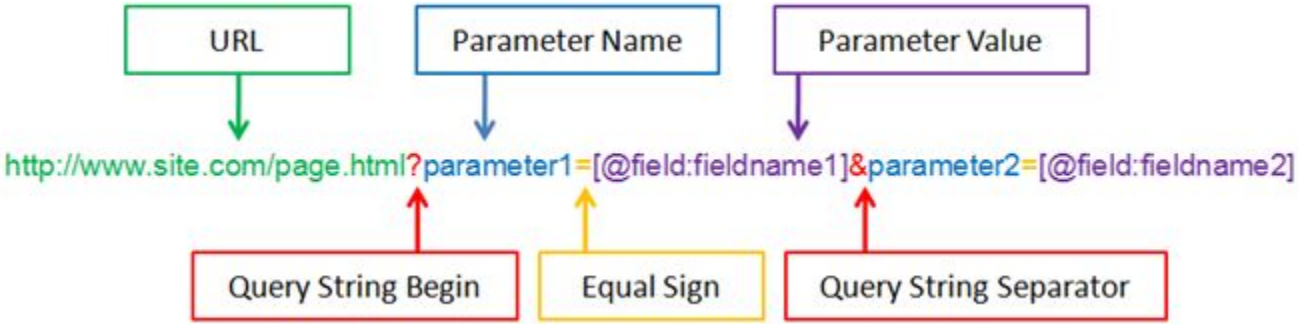
Random Surfing Model

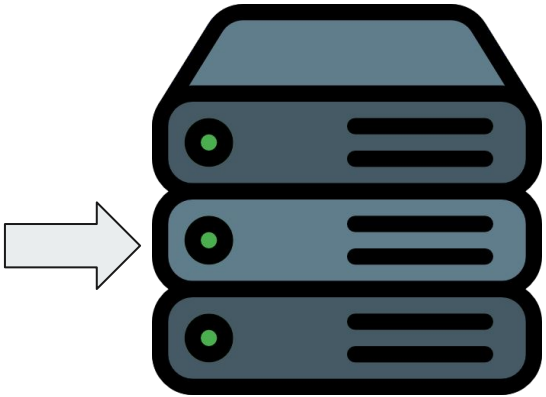
- The **random surfing model** is a graph model **which describes the probability of a random user visiting a web page.**

- The model attempts to predict the chance that a **random internet surfer will arrive at a page by either clicking a link** or **by accessing the site directly, for example by directly entering the website's URL in the address bar.**



The HITS Algorithm

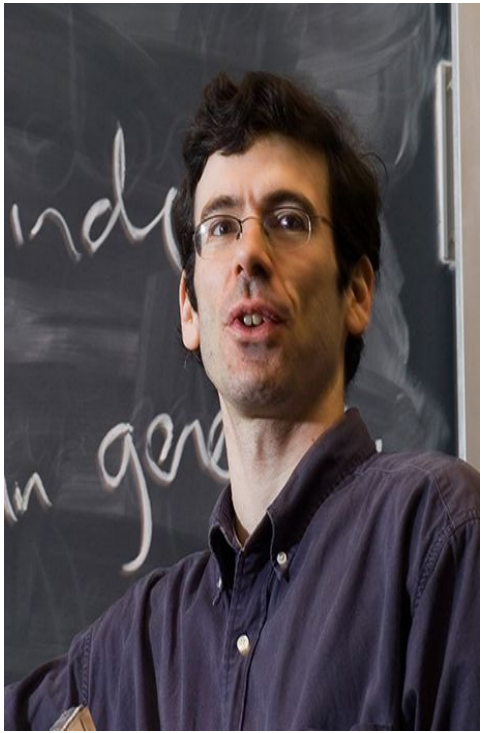




SERVER

QUERY STRING

The HITS Algorithm



HITS - **Hyperlink Induced Topic Search**

It is is a **Link Analysis Algorithm** that **rates Webpages,** developed by **Jon Kleinberg in 1999**. This algorithm is used to the **web link-structures to discover and rank the WebPages relevant for a particular search.**

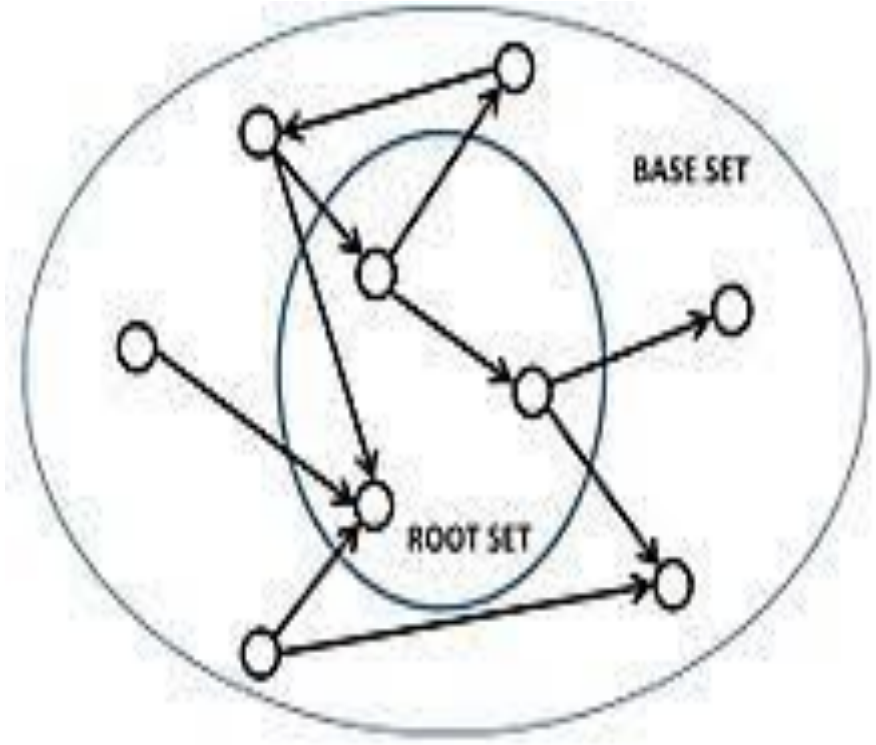
HITS use hubs and authorities to define a recursive relationship between web pages. To get knowledge of HITS Algorithm,

we first have to get knowledge about Hubs and Authorities.

The HITS Algorithm

Given a query to a Search Engine, the **set of highly relevant web pages are called Roots**. They are potential **Authorities**.

Pages that are not very relevant but point to pages in the Root are called Hubs. Thus, an **Authority is a page that many hubs link** and a **Hub is a page that links to many authorities.**



The HITS Algorithm

HITS Algorithm -

-> **Let number of iterations be k.**

-> Each node is assigned a **Hub score = 1** and an **Authority score = 1.**

-> **Repeat k times:**

Hub update: Each node's Hub score = **(Authority score of each node it points to).**

Authority update: Each node's Authority score= **(Hub score of each node pointing to it).**

Normalize the scores by **dividing each Hub score by square root of the sum of the squares of all Hub scores,** and **dividing each Authority score by square root of the sum of the squares of all Authority scores.**

Pagerank vs HITS Algorithm

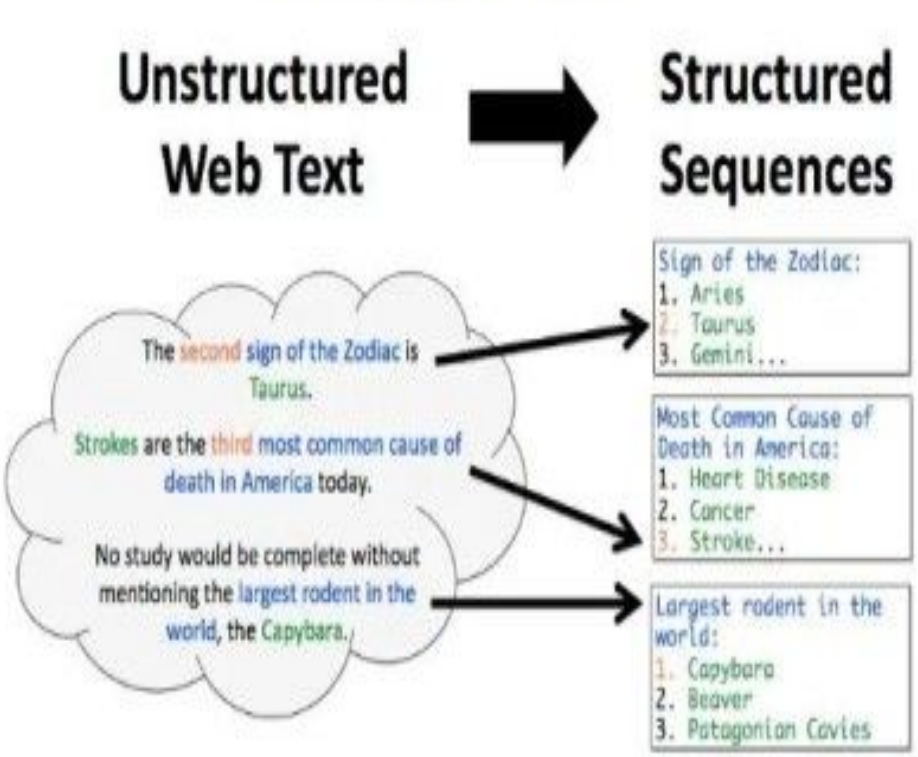
- **PageRank**
  (Google)
  - computed for all web pages stored in the database prior to the query
  - computes authorities only
  - Trivial and fast to compute

- **HITS**
  (CLEVER)
  - performed on the set of retrieved web pages for each query
  - computes authorities and hubs
  - easy to compute, but real-time execution is hard

Information Retrieval vs Information Extraction

S. No	IR	IE
1.	Task of finding text documents which are relevant to a user's information need	Goal is to extract pre-specified features from documents or display information.
2.	Document retrieval	Feature retrieval
3.	Actual information buried inside document	Extract information from within the document.
4.	Long listing of documents.	Aggregate over entire collection.
5.	Describes details of Google which is best IR system for the web	Extracted features are usually entered into a database automatically.

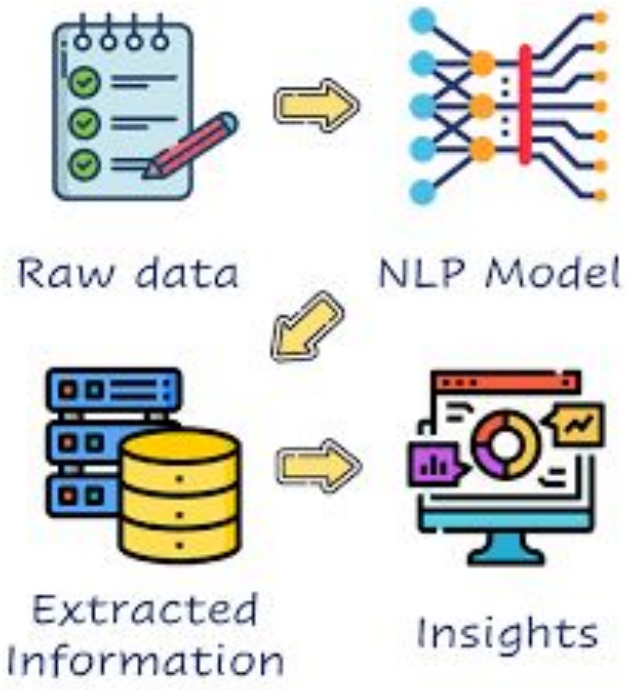
Information Extraction



Information extraction is **the process of extracting information from unstructured textual sources to enable finding entities as well as classifying and storing them in a database**.

In most of the cases **this activity concerns processing human language texts by means of natural language processing(NLP).**

Information Extraction



Identify phrases in language **that refer to specific types of entities and relations in text.**

Named entity recognition is the task of identifying names of people, places, organizations, etc. in text.

Information extraction is the process of **acquiring knowledge by skimming a text and looking for occurrences of a particular class of object and for relationships among objects. A difficult task is to extract instances of addresses from Web pages, with database fields for street, and zip code.**

Finite State Automata for Information Extraction

The **simplest type of information extraction system** is an **attribute-based extraction system** that assumes that the **entire text refers to a single object and the task is to extract attributes of that object.**

For example, the problem of extracting from the text **"IBM ThinkBook 970.Our price: Rs.399.00"**,
the set of attributes
{
Manufacturer - IBM,
Model - ThinkBook 970,
Price - Rs.399.00
}

Finite State Automata for Information Extraction

We can address this problem **by defining a template (also known as a pattern) for each attribute we would like to extract.**

The template is defined by a finite state automaton, **the simplest example of which is the regular expression, or regex.**

Regular expressions are used in **UNIX commands such as grep** and in **word processors such as Microsoft Word.**

Finite State Automata for Information Extraction

Templates are always defined in three parts: **a prefix regex**, **a target regex**, and **a postfix regex.**

For prices, the target regex is as above, **the prefix would look for strings such as "price:" and the postfix could be empty.**

The idea is that **some clues about an attribute come from the attribute value** itself **and some come from the surrounding text.**

Finite State Automata for Information Extraction

If a **regular expression for an attribute matches the text exactly once**, **then we can pull out the portion of the text that is the value of the attribute.**

If there are miss-match, it means that was a default value or given attribute missing; but **if there are several matches**, **we should a process to choose among them.** One strategy is to have several templates for each attribute, ordered by priority.

Finite State Automata for Information Extraction

For example, the top-priority example for price might look for the prefix **"our price:"** if that is not found, we look for the prefix **"price:"** and if that is not found, **the empty prefix.**

Another strategy **is to require all the matches and find some way to choose among them.**

One step up from **attribute-based extraction systems are relational extraction** systems, which deal with multiple objects and also the relations among them.

Thus, **when these systems see the text "Rs.249.99," they need to determine not just that it is a price, but also which object has that price.**

A relational extraction system can be built as a series of cascaded finite-state transducers.

Finite State Automata for Information Extraction

So the system consists of a series of small, **efficient finite-state automata (FSAs)**, where **each automaton receives text as input**, **transfers the text into a different format, and passes it along with the next automaton.**

FASTUS - Cascaded Finite-State Transducer for Extracting Information

FASTUS consists of five stages:

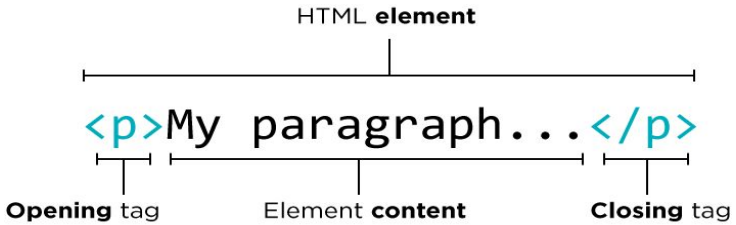
1. Tokenization
2. Complex-word handling
3. Basic-group handling
4. Complex-phrase handling
5. Structure merging

Finite State Automata for Information Extraction

TOKENIZATION :

Which segments the stream of characters into tokens (words, numbers, and punctuation). **For English, tokenization can be simple; just separating characters at white space or punctuation does a fairly good job.**

Some tokenizes also deal with markup languages such as **HTML, SGML, and XML.**



COMPLEX WORDS :

Names such as **"Bridgestone Sports Co."** These are recognized by a **combination of lexical entries and finite-state grammar rules.**

For example, a company name might be recognized by the rule Capitalized Word+ **("Company" / "Co" | "Inc" | "Ltd").**

BRIDGESTONE

Finite State Automata for Information Extraction

BASIC GROUP HANDLING:

The third stage handles **basic groups,** meaning **noun groups** and **verb groups**.

The example sentence :
1 NG: Bridgestone Sports Co. **2 VG:** said **11 CJ:** and
3 NG: Friday **12 NG:** a Japanese trading house
4 NG: it **13 VG:** to produce

Here NG stands for noun group, VG stands for verb group, PR stands for preposition, and CJ stands for conjunction.

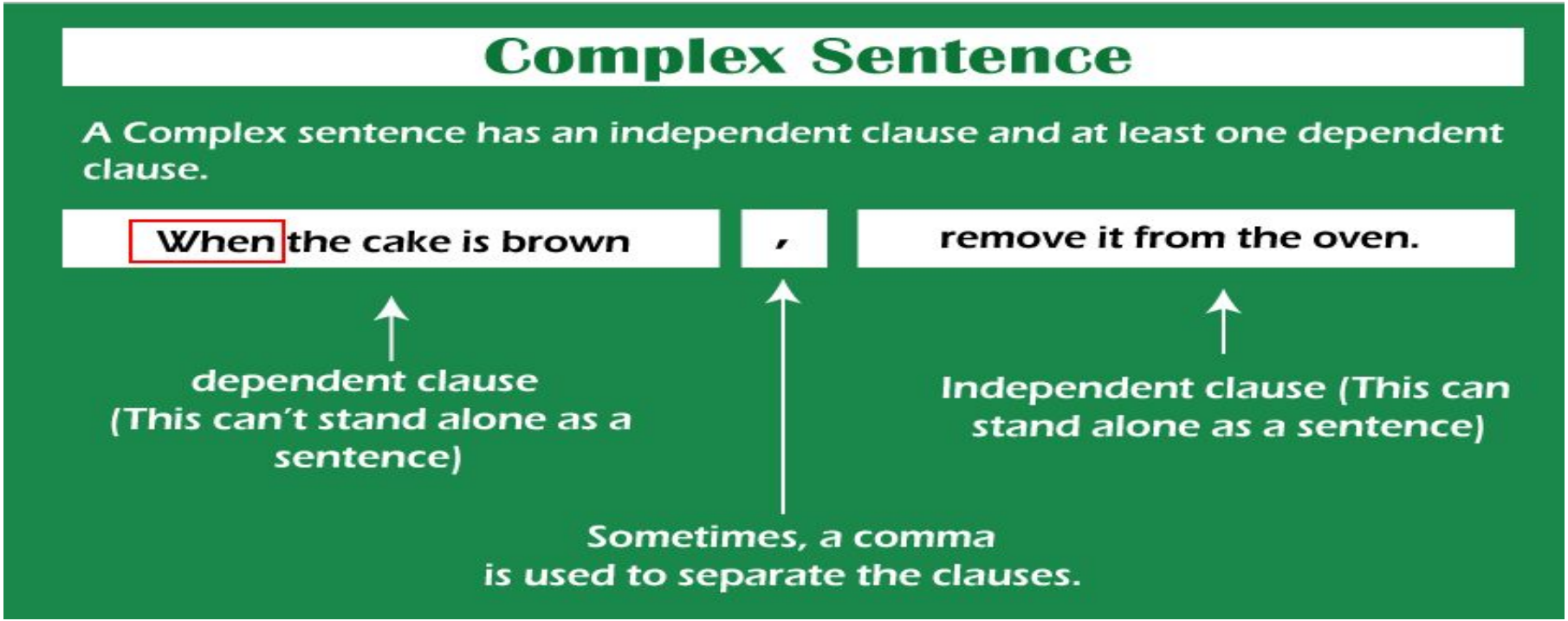
COMPLEX PHRASE HANDLING:

Again, **the aim is to have rules that are finite-state** and therefore it can be processed quickly, and will result in unambiguous (or nearly unambiguous) output phrases. **One type of combination rule deals with domain-specific events.**

A sequence of two or more words arranged in a grammatical unit and lacking a finite verb

Finite State Automata for Information Extraction

COMPLEX PHRASE HANDLING:



Probabilistic Model for Information Extraction

Hidden Markov models (HMMs), named after the Russian mathematician Andrey Andreyevich Markov, who developed much of relevant statistical theory, are introduced and studied in the early 1970s.

They were first used in speech recognition and have been successfully applied to the analysis of **biological sequences** since late 1980s.

Nowadays, they are considered as a specific form of dynamic Bayesian networks, **which are based on the theory of Bayes.**

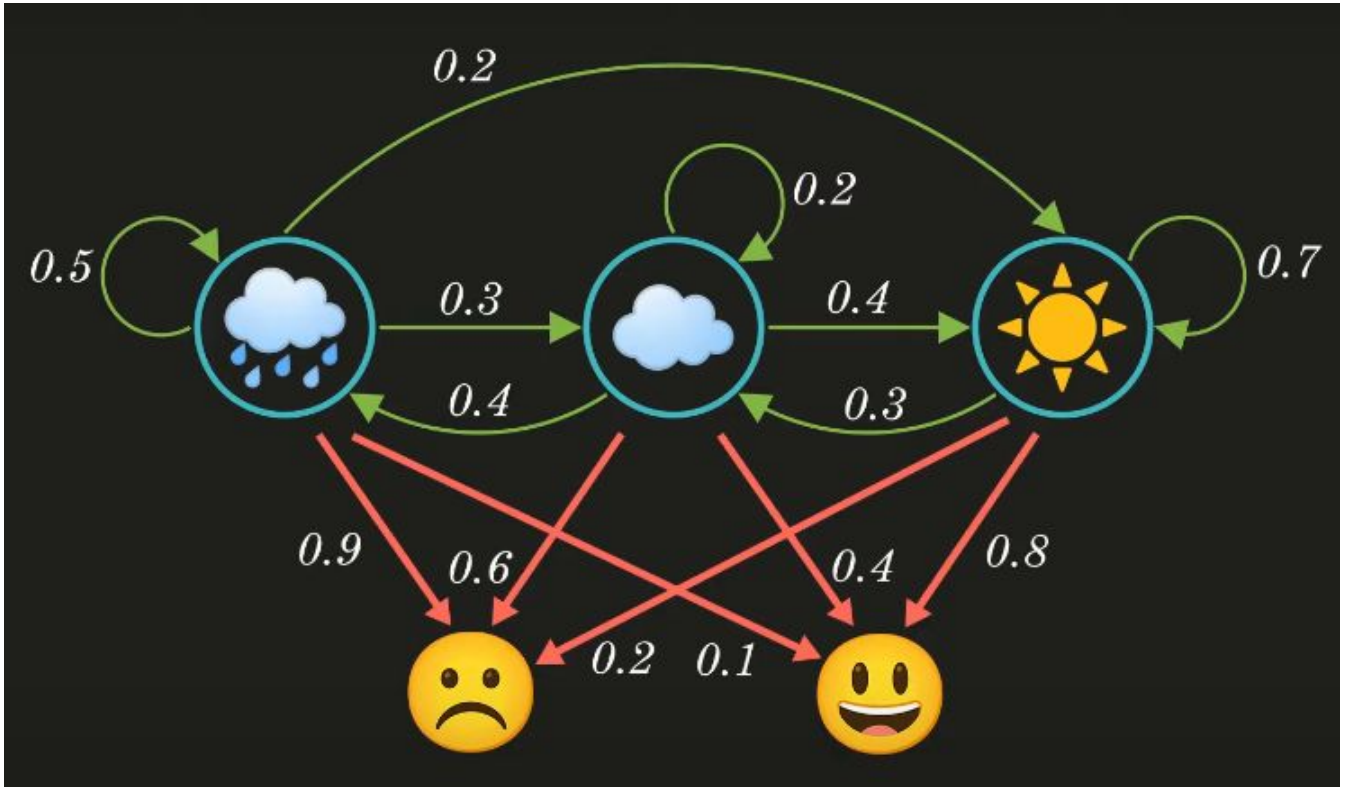
HMMs are statistical models to capture hidden information from observable sequential symbols

Probabilistic Model for Information Extraction

Weather

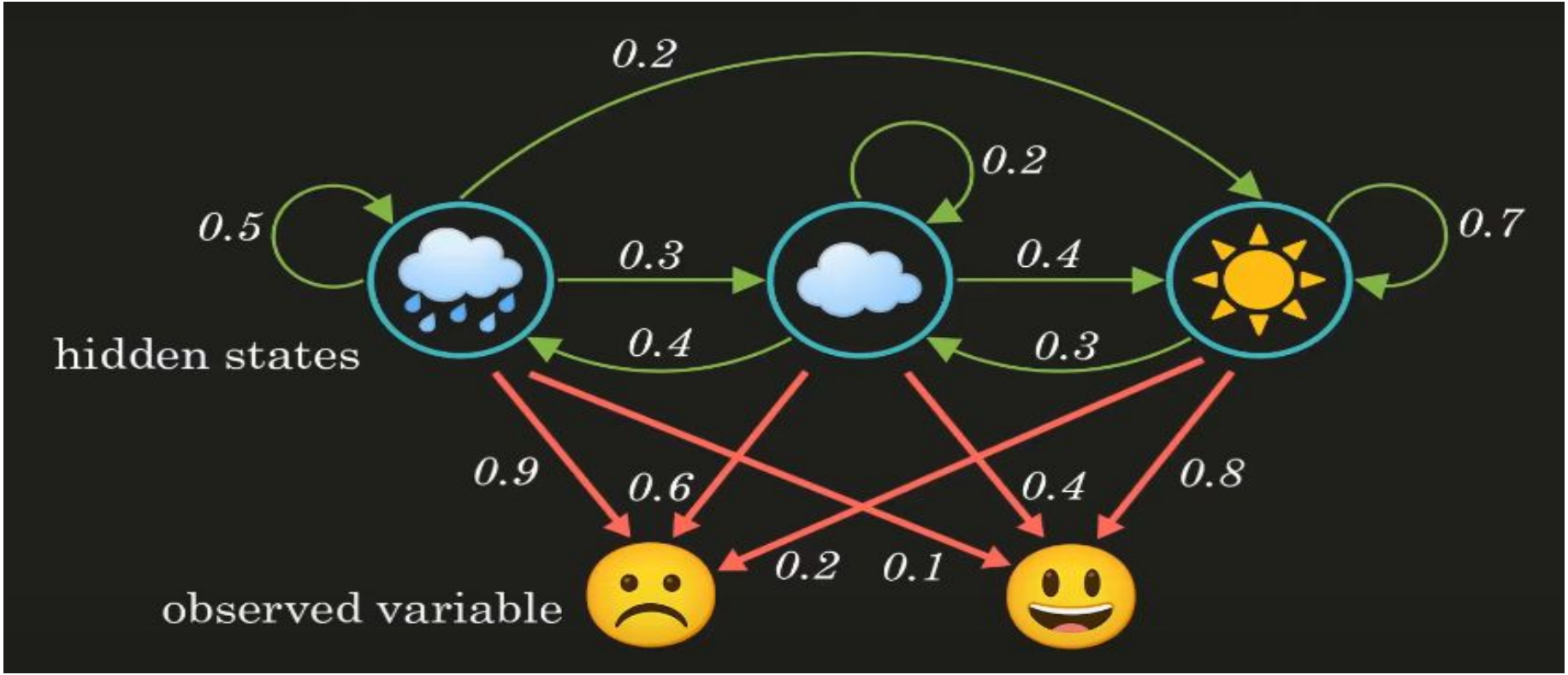
Probability

Probabilistic Model for Information Extraction

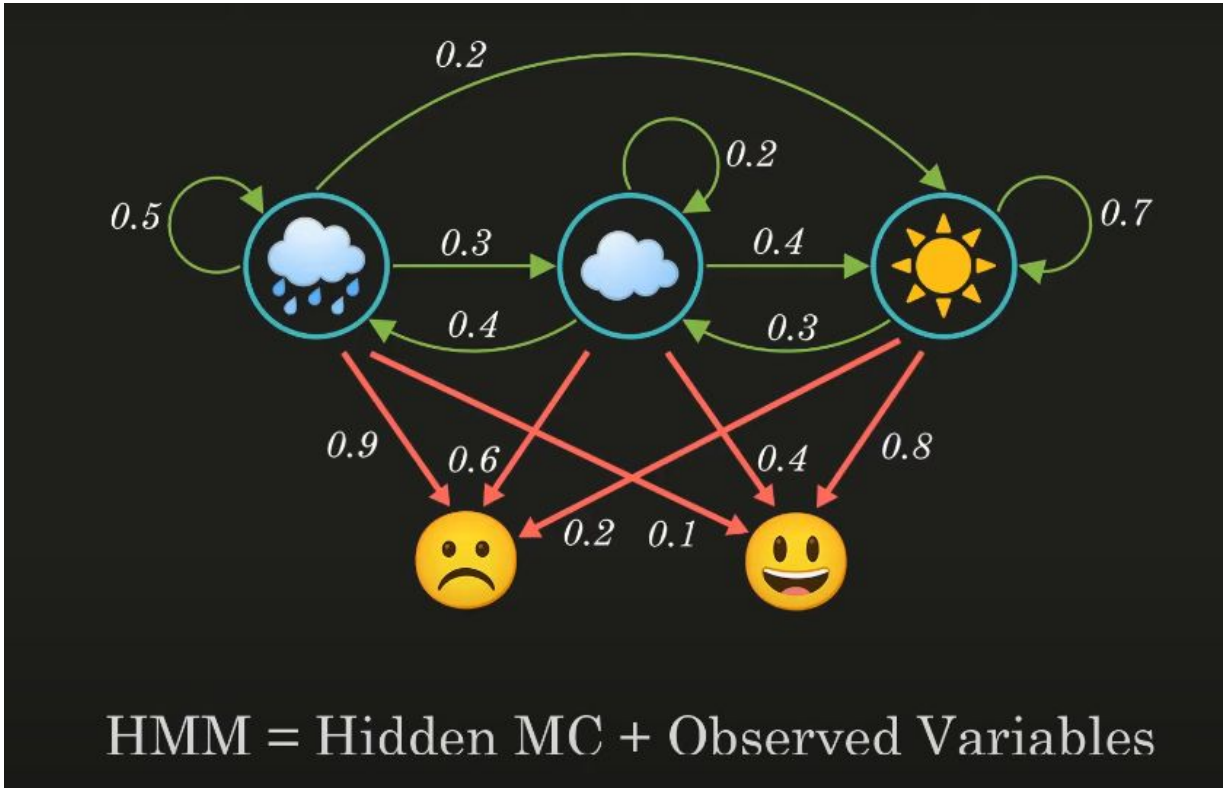


Probability of Mood Based on Weather

Probabilistic Model for Information Extraction



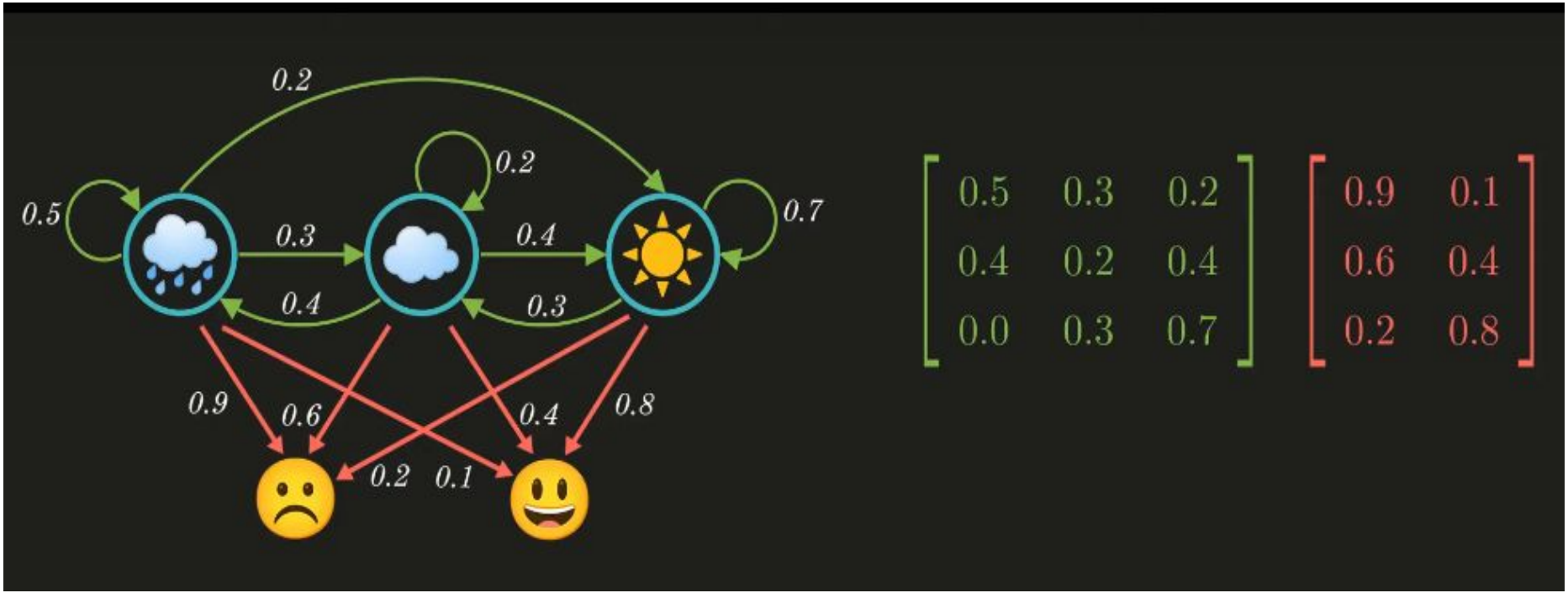
Probabilistic Model for Information Extraction



Hidden Markov Model = Hidden MC + Observed Variables

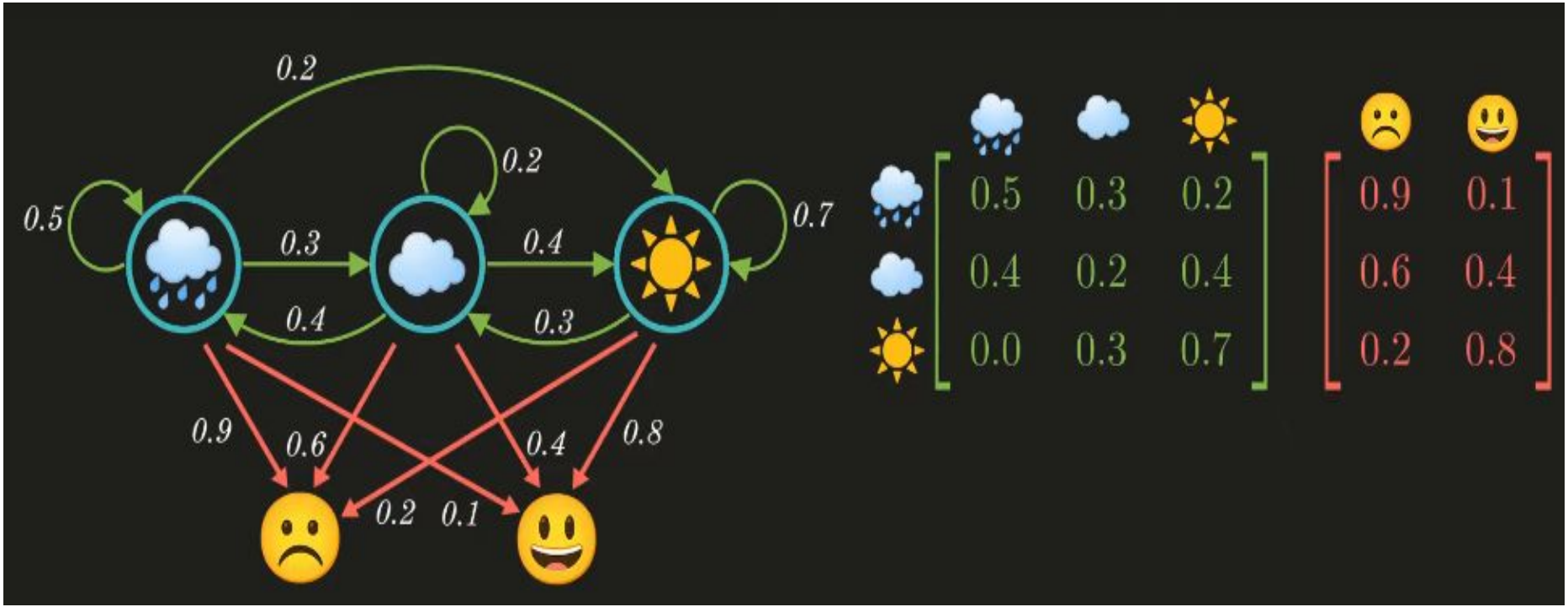
Probabilistic Model for Information Extraction

Matrix Representation of Scenario



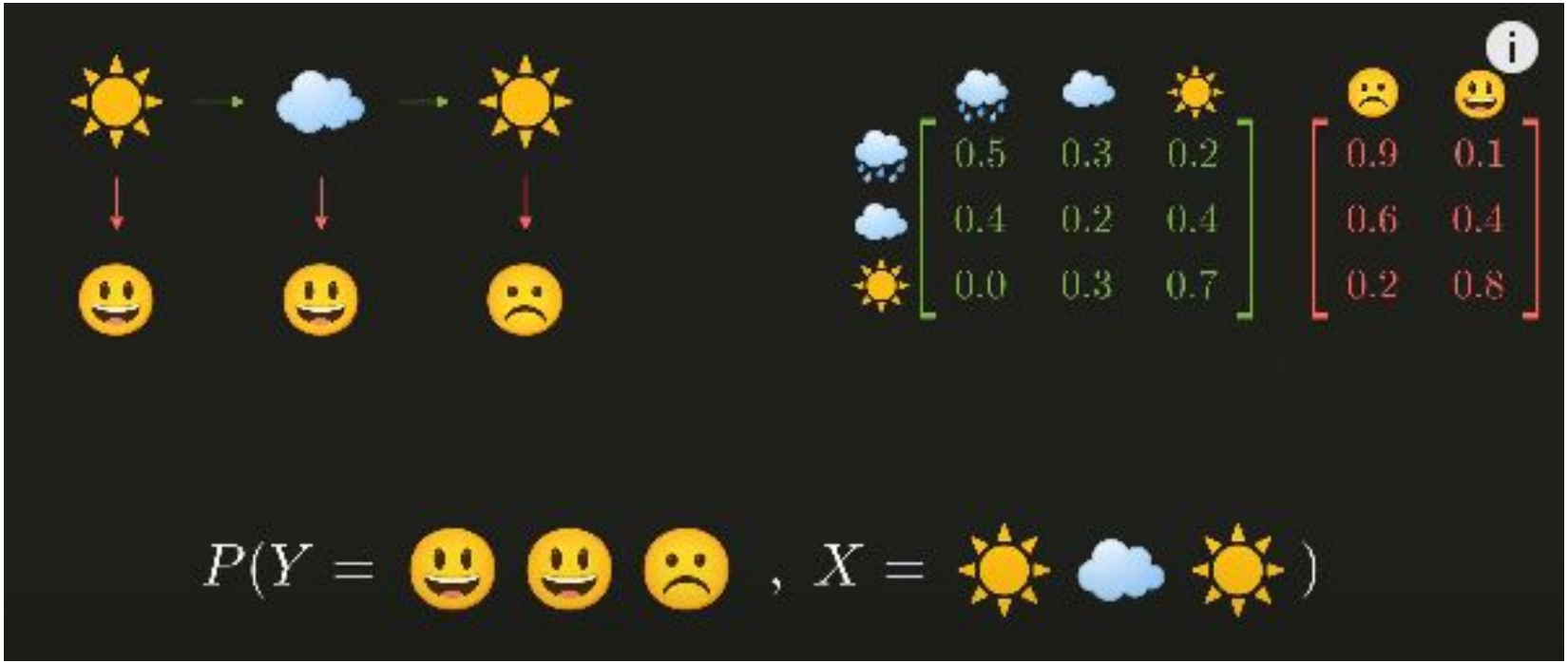
Probabilistic Model for Information Extraction

Matrix Representation of Scenario

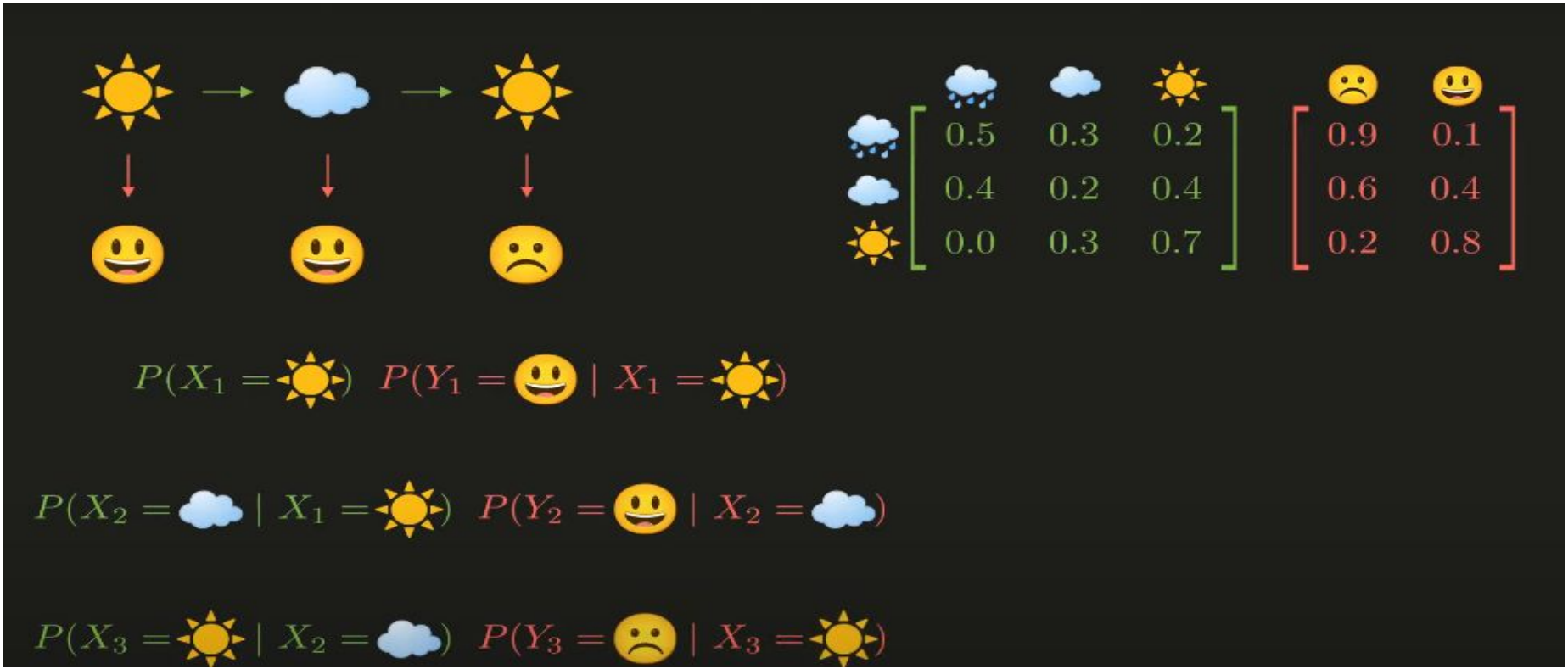


Probabilistic Model for Information Extraction

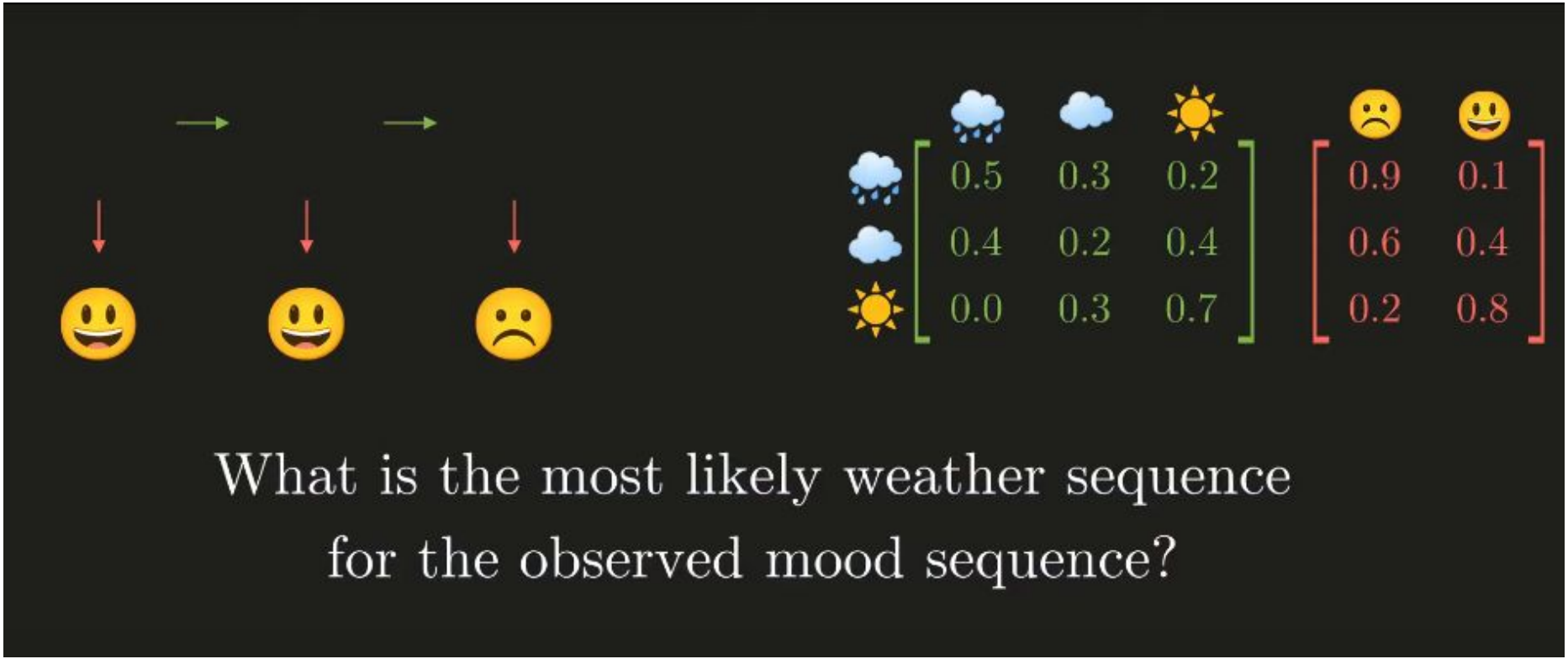
Mathematical Representation



Probabilistic Model for Information Extraction

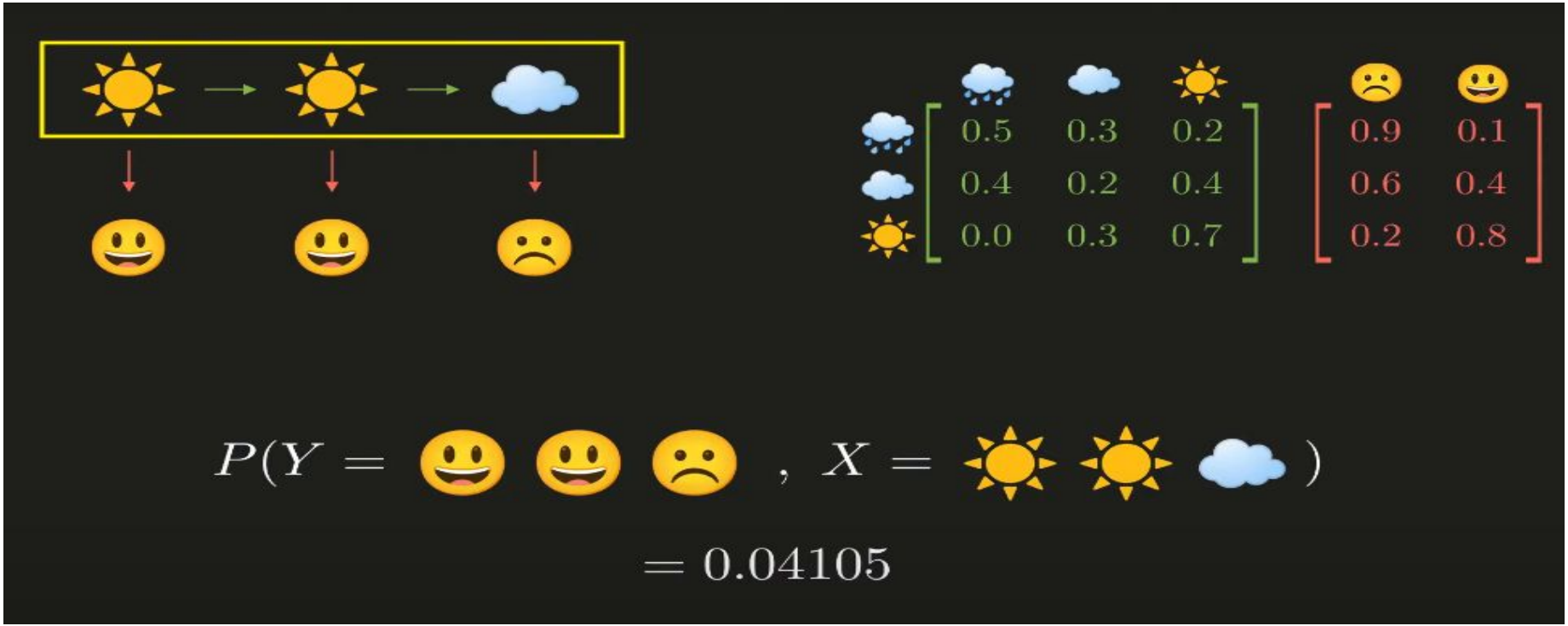


Probabilistic Model for Information Extraction



Probabilistic Model for Information Extraction

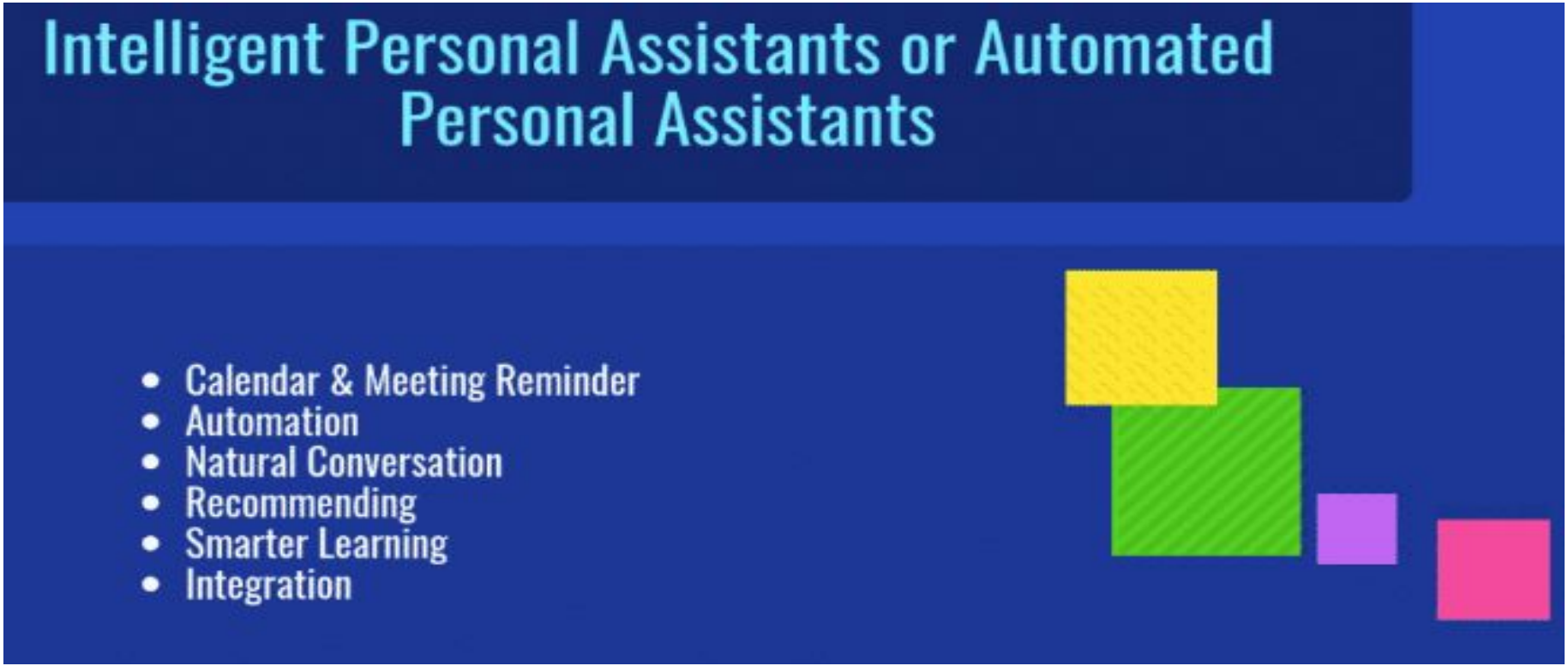
Calculated by Hidden Markov Model



Case Study : Automated Voice Assistants, Chat Bots

Intelligent Personal Assistants, answer the queries and perform actions via voice commands using natural language user interface.

Case Study : Automated Voice Assistants, Chat Bots



Case Study : Automated Voice Assistants, Chat Bots

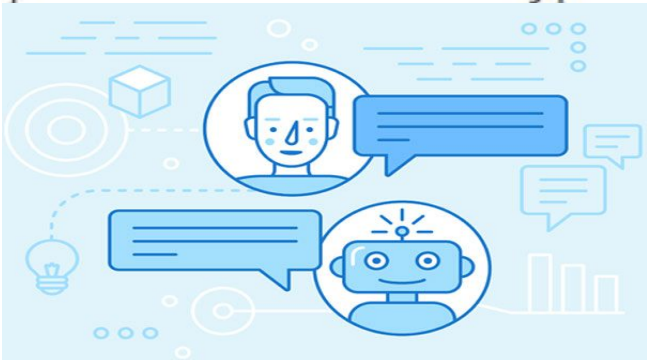
Calendar & Meeting Reminder: Intelligent Personal Assistant schedule meetings & appointments reminders instantly on behalf of the user. IPAs also helps the user remember everything they have set it to remind them and send the user signals, photos, links and more via SMS, emails or other means. This software can set alarms to tell the user of an upcoming event or task.

Case Study : Automated Voice Assistants, Chat Bots

Automation: Help to automate most of the essential functions that the user wants. The user can utilize IPAs to do research, identify landmarks, shop, and translate foreign languages among other tasks.

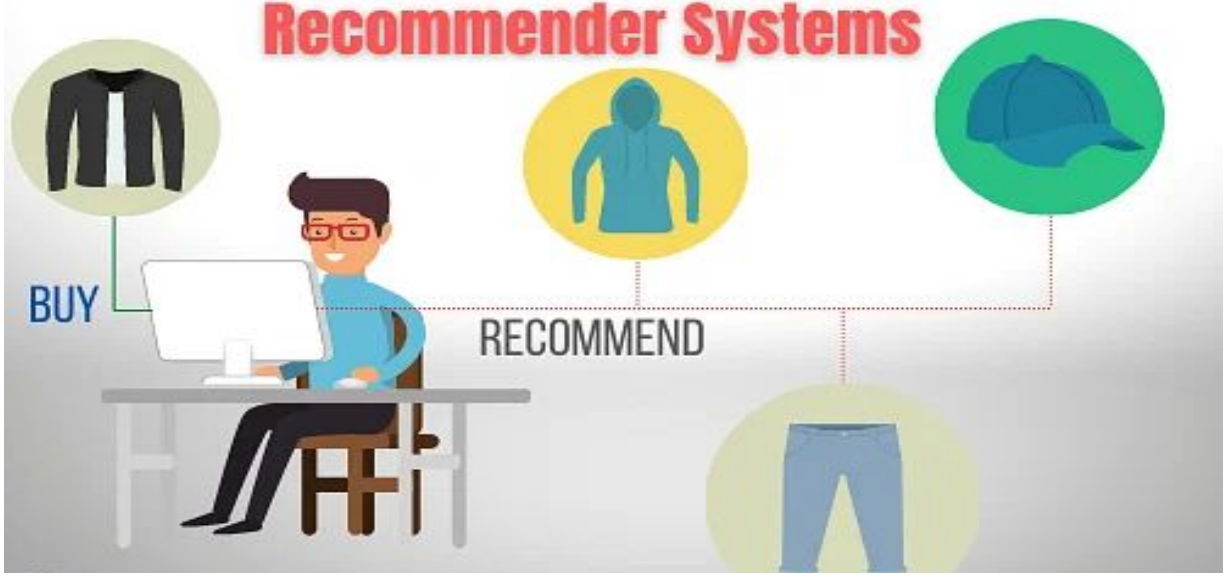
Case Study : Automated Voice Assistants, Chat Bots

Natural Conversation: Intelligent Personal Assistant can understand and respond to complex questions. It recognizes the intent of the user's inquiry, personalizes the responses based on context, and troubleshoots the problem using conversational strategies when answering social questions, reacting to customer frustrations and even becomes a live chat agent when need be. The user can create reminders, ask questions and even type anything they want by speaking to the program.



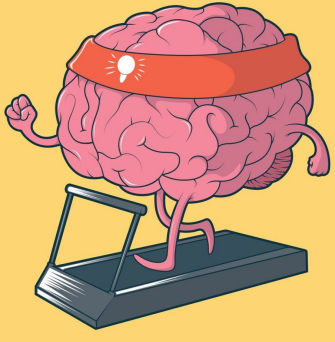
Case Study : Automated Voice Assistants, Chat Bots

Recommending: They can recommends things, places and items to the user. The user can find whatever they need including shops, hospitals and more with a swipe or tap.



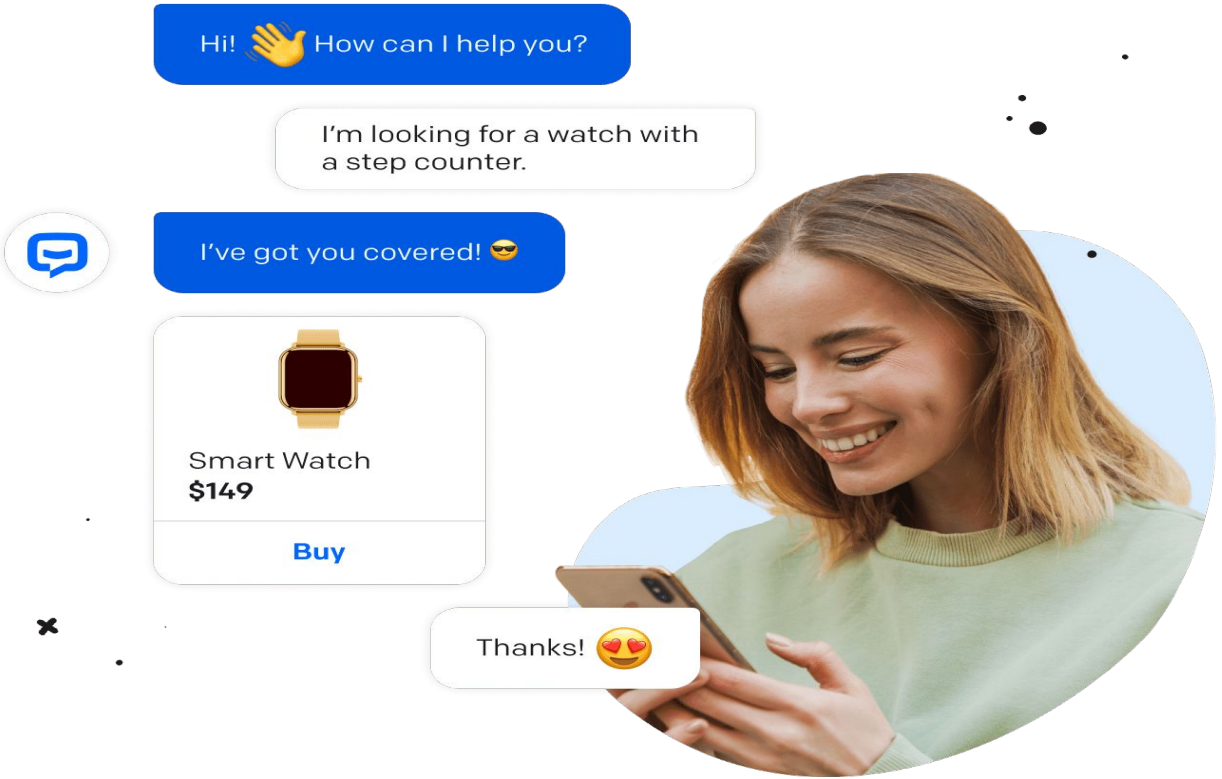
Case Study : Automated Voice Assistants, Chat Bots

Smarter Learning: Artificial intelligence technology uses both machine learning and natural language understanding which allows it to obtain industry-specific knowledge and unique business data and thus can do marketing for an enterprise.



Case Study : Automated Voice Assistants, Chat Bots

A chatbot is software that simulates human-like conversations with users via **chat**. Its key task is to answer user questions with instant messages.



Thank You !!