# Introduction to Text Mining

Yogesh Kulkarni

February 26, 2021

# Information Extraction

*"The task of Information Extraction (IE) involves extracting meaningful information from unstructured text data and presenting it in a structured format."*

# Example

We can extract the following information from the text:

> Indian captain Virat Kohli was dismissed cheaply for just 2 in Wellington on Friday by debutant Kyle Jamieson extending a rare lull in the batsman's stellar career. Throughout the ongoing New Zealand tour, Kohli has managed to score just a single fifty across 8 innings in all 3 international formats.

- Country – India, Captain – Virat Kohli
- Batsman – Virat Kohli, Runs – 2
- Bowler – Kyle Jamieson
- Match venue – Wellington
- Match series – New Zealand
- Series highlight – single fifty, 8 innings, 3 formats

Makes text structured meaning with relationships (key value).

(Ref: Hands-on NLP Project: A Comprehensive Guide to Information Extraction using Python - Aniruddha Bhandari)

# Information Extraction Use case

# As a Task

**As a task:** **Filling slots in a database from unstructured text.**

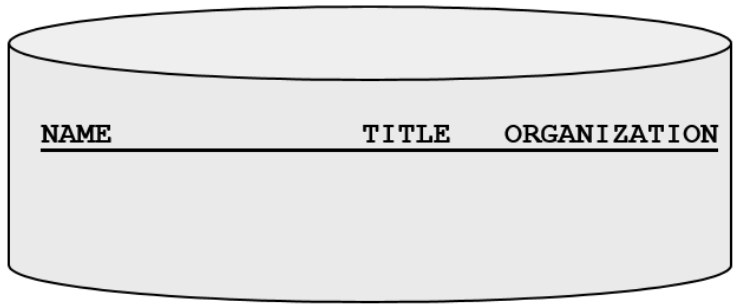
October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is [..].

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying...



# Task

**Task:** **Extract structured data, such as tables, from unstructured text**

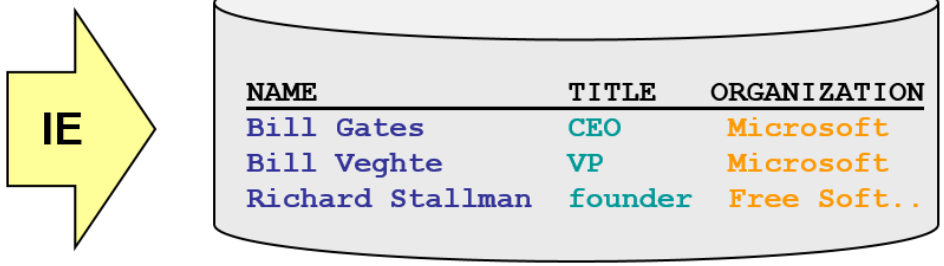
October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is [..].

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying...



# Extraction

**Information Extraction =**
**segmentation + classification + association**

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying…

Microsoft Corporation
CEO
Bill Gates
Microsoft
Gates
Microsoft
Bill Veghte
Microsoft
VP
Richard Stallman
founder
Free Software Foundation

aka "named entity extraction/detection"

Adapted from slide by William Cohen

# Techniques

**A family of techniques:**

**Information Extraction = segmentation + classification + association**

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying...

Microsoft Corporation
CEO
Bill Gates
Microsoft
Gates
Microsoft
Bill Veghte
Microsoft
VP
Richard Stallman
founder
Free Software Foundation

# Example

**Information Extraction =**
**segmentation + classification + association**

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying...

Microsoft Corporation
CEO
Bill Gates

Microsoft
Gates

Microsoft

Bill Veghte
Microsoft
VP

Richard Stallman
founder
Free Software Foundation

Adapted from slide by William Cohen

# Landscape of Information Extraction

## Web site specific

Formatting

Amazon.com Book Pages


## Genre specific

Layout

Resumes


## Wide, non-specific

Language

University Names


Adapted from slide by William Cohen

# Complexity of Information Extraction

**Closed set**

U.S. states

He was born in Alabama…

The big Wyoming sky…

**Regular set**

U.S. phone numbers

Phone: (413) 545-1323

The CALD main office can be reached at 412-268-1299

**Complex pattern**

U.S. postal addresses

University of Arkansas
P.O. Box 140
Hope, AR 71802

Headquarters:
1128 Main Street, 4th Floor
Cincinnati, Ohio 45210

**Ambiguous patterns, needing context and many sources of evidence**

Person names

…was among the six houses sold by Hope Feldman that year.

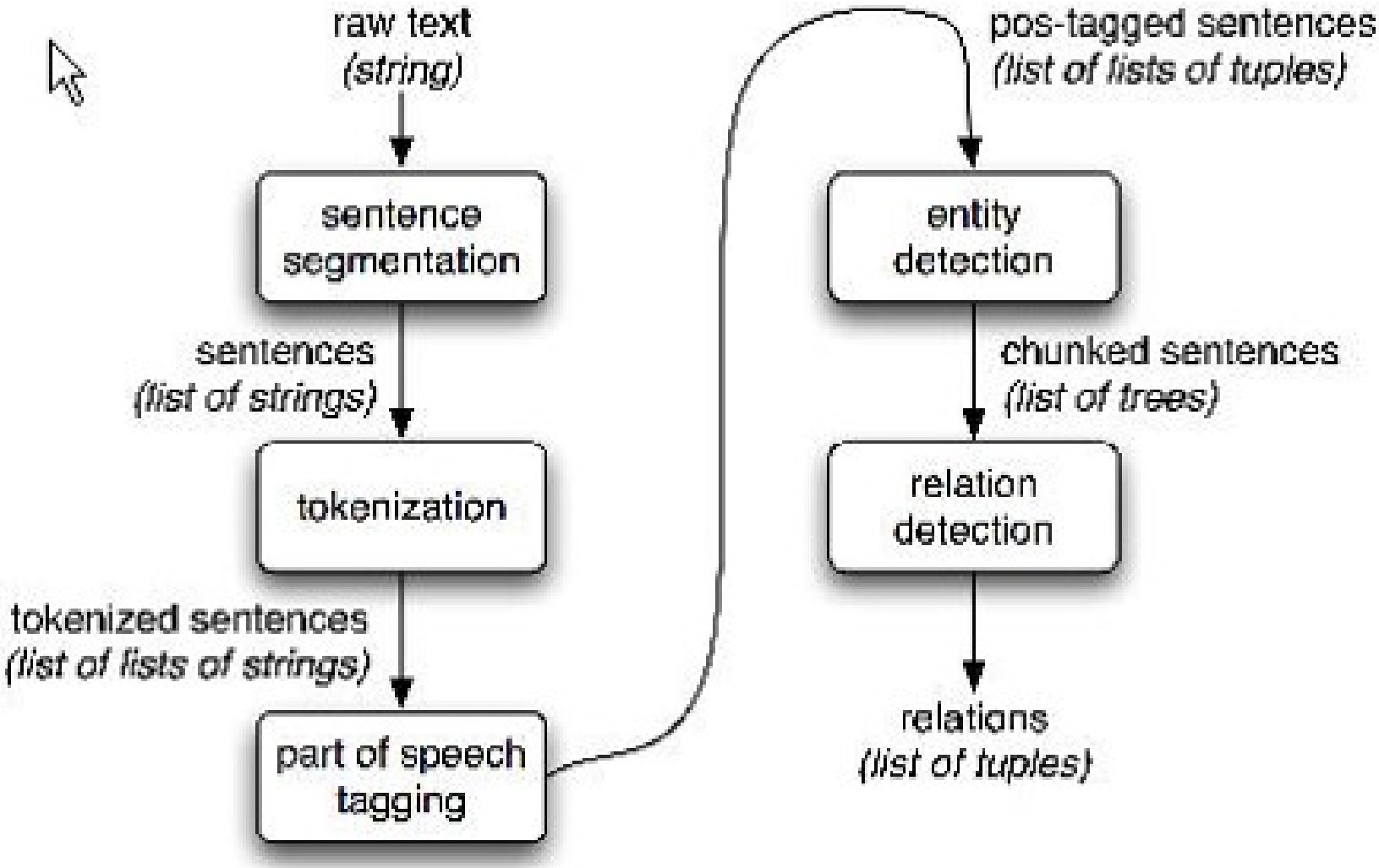
Pawel Opalinski, Software Engineer at WhizBang Labs.

Adapted from slide by William Cohen

# Applications of Information Extraction

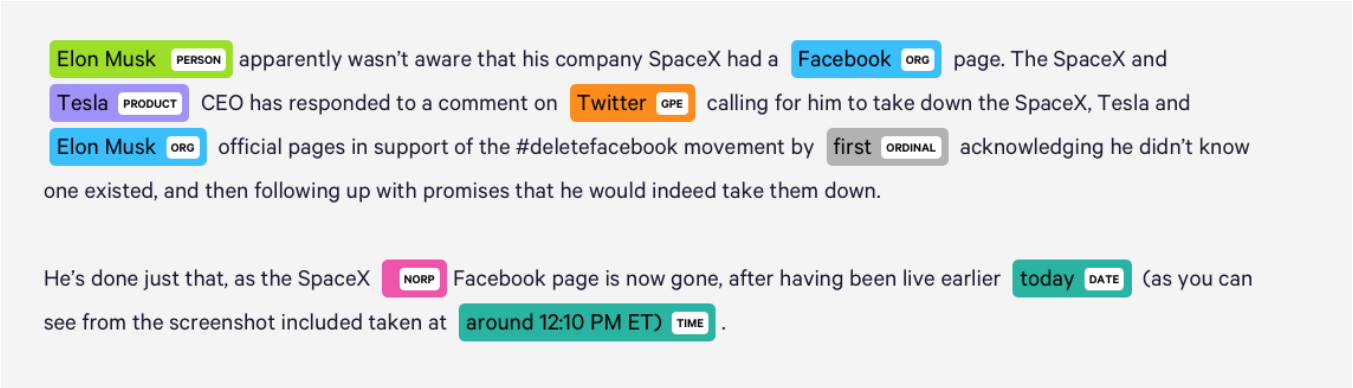
- Extract structured data out of electronically-available scientific literature, especially in the domain of biology and medicine
- Legal documents
- Business intelligence
- Resume harvesting
- Media analysis
- Sentiment detection
- Patent search
- Email scanning

# Architecture of Information Extraction



# Main tasks of Information Extraction

- Named Entity Recognition (Chunking, Parsing)
- Relation Extraction
- Relations like subject are syntactic, relations like person, location, agent or message are semantic

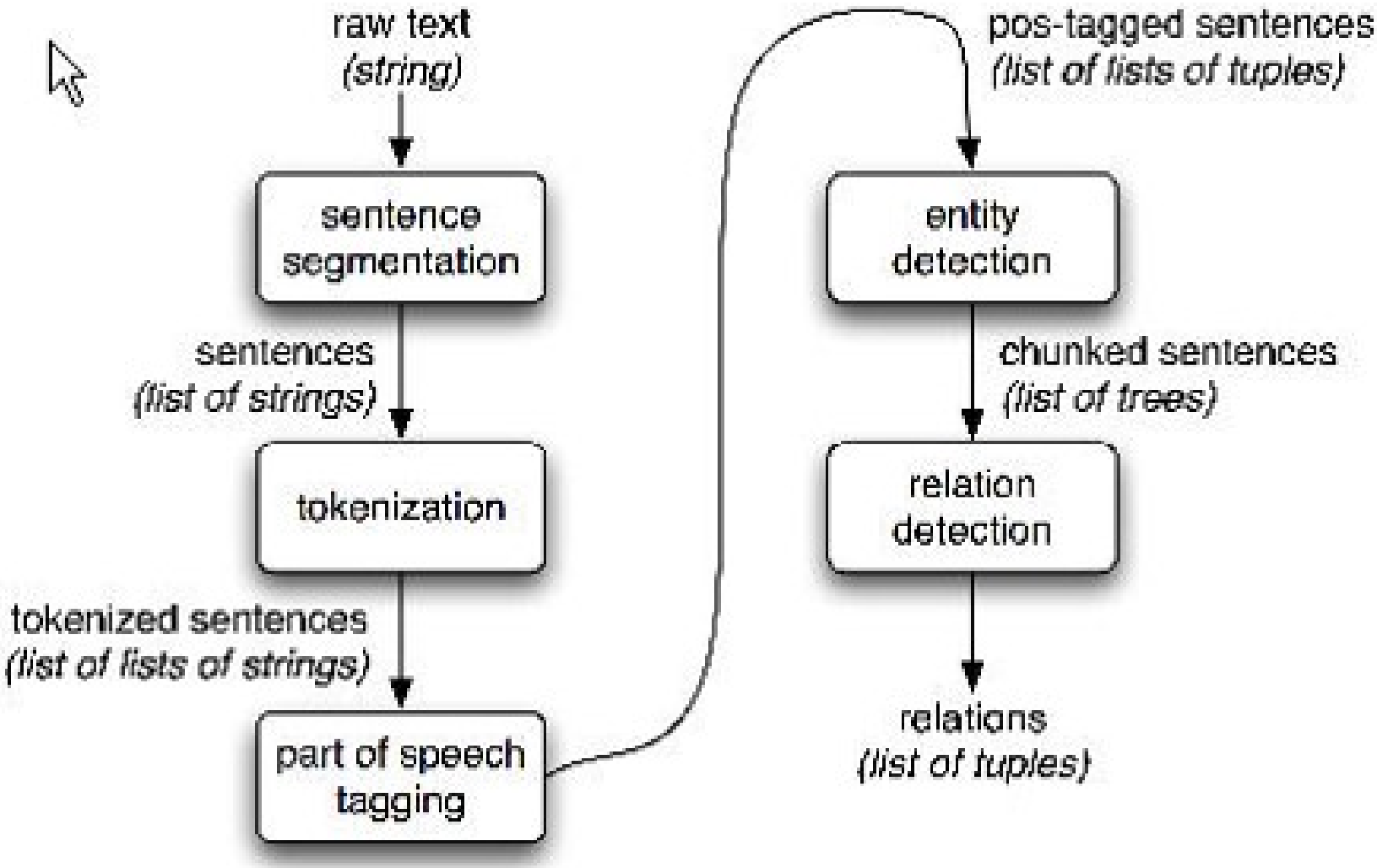


# Example of Information Extraction

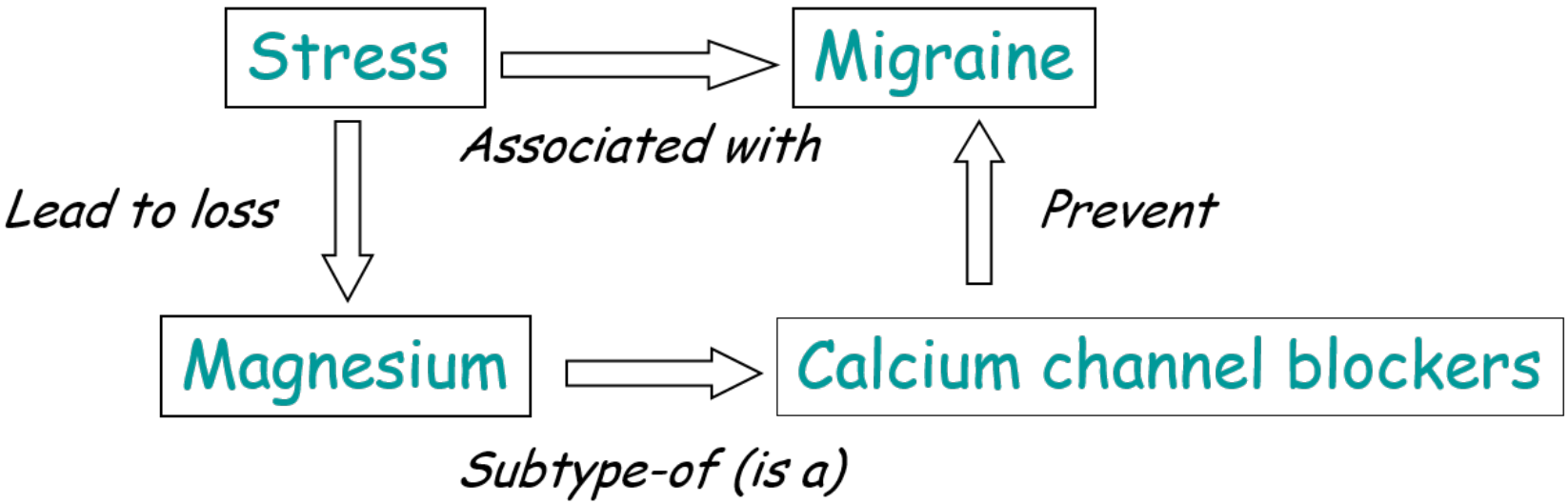
Text:

- Stress is associated with migraines
- Stress can lead to loss of magnesium
- Calcium channel blockers prevent some migraines
- Magnesium is a natural calcium channel blocker

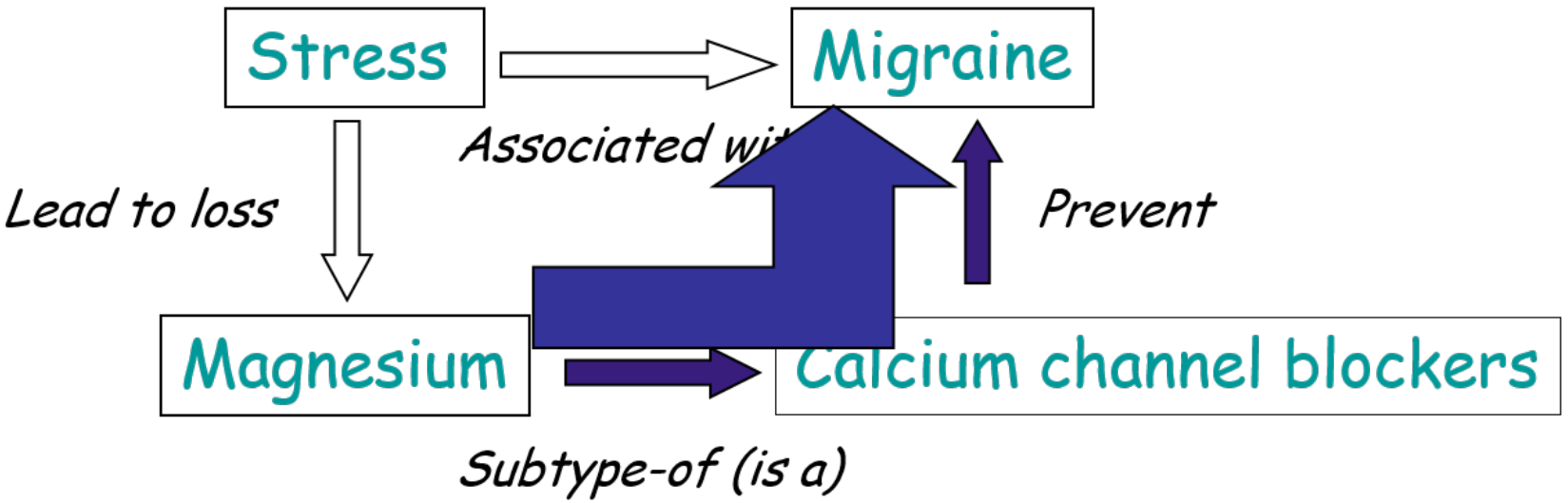
# Extract semantic entities from text



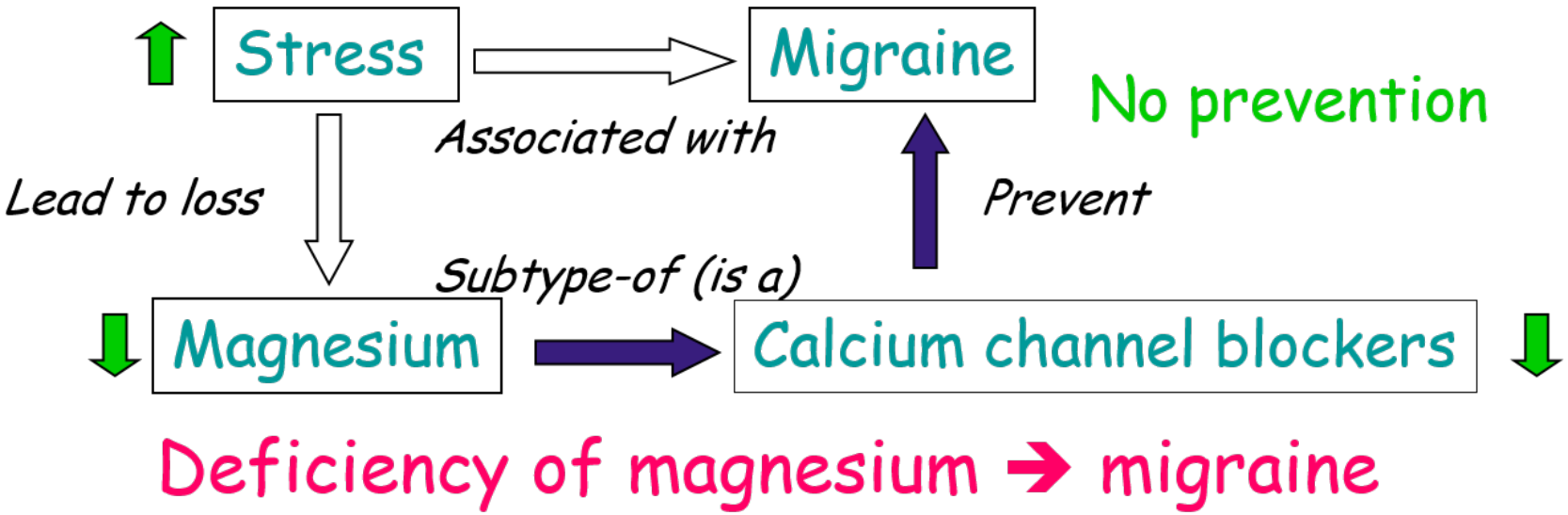
# Classify relations between entities



# Do reasoning: find new correlations



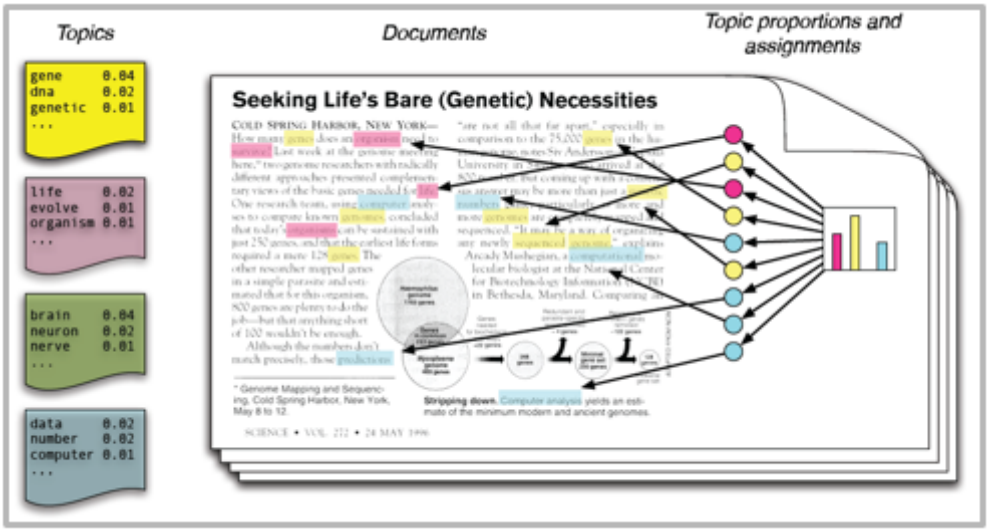
# Do reasoning: infer causality



# Topic Modeling

# What is Topic Modeling?

- Topic modeling is a form of text mining, a way of identifying patterns in a corpus.
- You take your corpus and run it through a tool which groups words across the corpus into 'topics' ( clusters of words) by a process of similarity.
- In a good topic model, the words in topic make sense, for example "navy, ship, captain" and "tobacco, farm, crops."



# Core Concept: Similarity

Are these statements similar?

- "A seven-year quest to collect samples from the solar system's formation ended in triumph in a dark and wet Utah desert this weekend."
- "For a month, a huge storm with massive lightning has been raging on Jupiter under the watchful eye of an orbiting spacecraft."
- "One of Saturn's moons is spewing a giant plume of water vapour that is feeding the planet's rings, scientists say."

How would you quantify their similarity? How would you decide that two are more similar to each other than to the third?

## Semantic Similarity

- A study done in Australia in 2005 : what a sample of Australian college students think is similar.
- Students read hundreds of paired short excerpts from news articles and ranked the pairwise similarity on a scale.
- Then examined all the classifications, and found that they had a correlation of 0.6.
- That's obviously a positive correlation, but not terribly high.
- So, humans doesn't necessarily agree with each other about semantic similarity
- It's kind of a fuzzy notion

## Semantic Similarity: Any number?

- The study mentioned found that a particular topic modelling technique called latent semantic analysis (LSA) could achieve also a 0.6 correlation with the human ratings
- Correlating with the study participant's choice about as well as they correlated with each other.

# Who cares about semantic similarity?

Some use cases:

- Query large collections of text: Traditionally used on large document collections. Legal discovery. Answer questions or aid searching huge corpora like government regulations, manuals, patent databases, etc.
- Automatic metadata: system can intelligently suggest tags and categories for documents based on other documents they're similar too.
- Recommendations: plagiarism detection, exam scoring. And there are a number of other use cases.
- Better human-computer interaction: Matching on similarity rather than words or with regexes allows us to accept broader ranges of input, in theory.

# What is Topic Modeling?

- Topic modelling attempts to uncover the underlying semantic structure of text (or other data) by using statistical techniques to identify abstract, recurring patterns of terms in a set of data.
- These patterns are called topics. They may or may not correspond to our intuitive notion of a topic.
- Topic modelling models documents as collections of features, representing the documents as long vectors that indicate the presence/absence of important features, for example, the presence or absence of words in a document.
- We can use those vectors to create spaces and plot the locations of documents in those spaces and use that as a kind of proxy for their meaning.

# What is Topic Modeling?

- Topic modeling is an unsupervised algorithm.
- One of the key ideas behind it is that every topic is present to varying degrees within each document.
- The typical output of running a corpus through TM is a set of distinctive words for each topic and for each document a percentage value indicating the presence of each topic within that document.
- One of the popular algorithms underlying TM is called Latent Dirichlet Allocation (LDA) and was described in a paper published by Blei, Ng and Jordan in 2003.

# What Topic Modeling is NOT?

Topic modelling:

- Does not parse sentences.
- In fact, knows nothing about word order.
- Makes no attempt to "understand" grammar or language syntax.

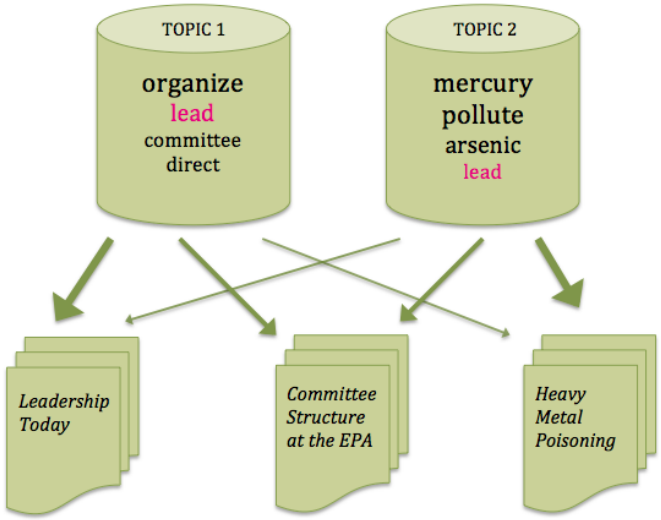
# How does it work?

Manual mode:

- Imagine working through an article with a set of highlighters
- A different color for the key words of themes within the paper as you come across them
- Once done, you could copy out the words as grouped by the color you assigned them. That list of words is a topic, and each color represents a different topic.
- The popular LDA which is used to extract topics is based on "Dirichlet Distribution".
- Trying to understand LDA (and its proof) is not trivial (so leave it!!)

# Still, how does it work?

- Each document contains a mixture of different topics.
- "topic" can be understood as a collection of words that have different probabilities of appearance.
- One topic might contain many occurrences of "organize," "committee," "direct," and "lead."
- Another might contain a lot of "mercury" and "arsenic," with a few occurrences of "lead."



# Still, how does it work?

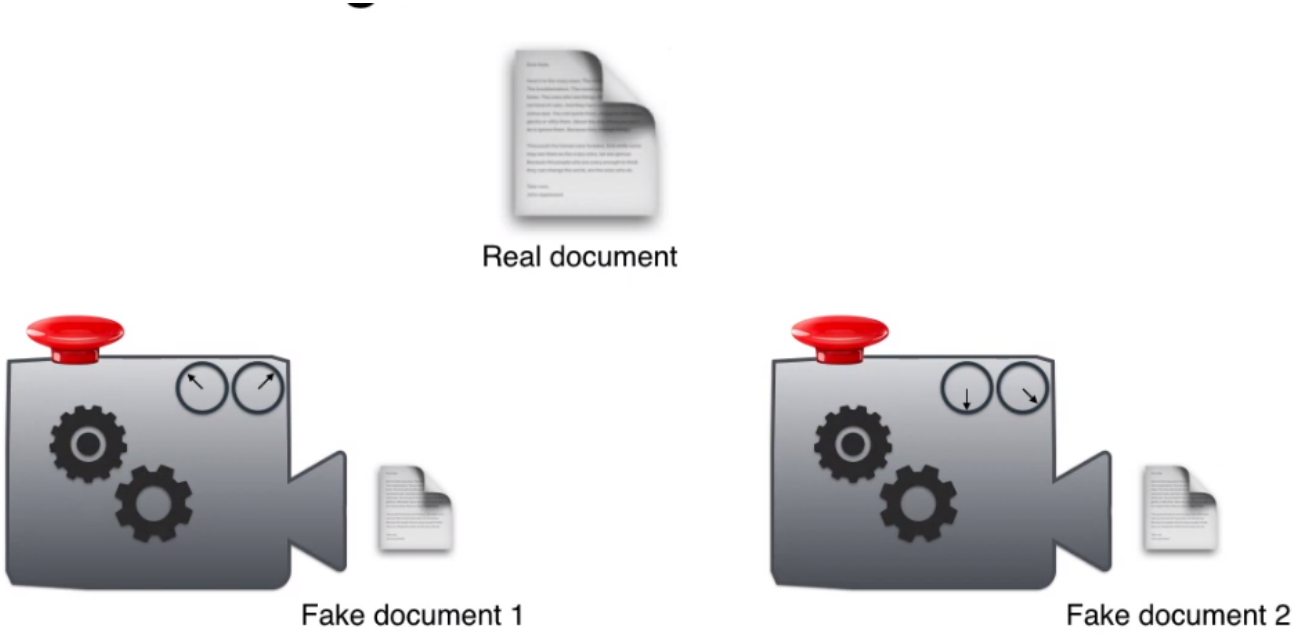
- Can't directly observe topics; what we have are documents
- Topic modeling is a way of extrapolating backward to infer the discourses ("topics") that could have generated them.
- How are we going to decide whether this occurrence of W belongs to topic Z?
- But one way to guess is to consider two questions.
  - How often does "lead" appear in topic Z?
  - How common is topic Z in the rest of this document?
- We will find probability of W present in Z for this document D, as follows:

$$P(Z|W,D) = \frac{\#\ of\ word\ W\ in\ topic\ Z + \beta_w}{total\ tokens\ in\ Z + \beta} * (\#\ words\ in\ D\ that\ belong\ to\ Z + \alpha)$$

- Its is calculated by multiplying the frequency of this word type W in Z by the number of other words in document D that already belong to Z.

# Document generation

Document is generated using some settings. Which set of settings (machine 1 or machine 2) are more similar to the real document?

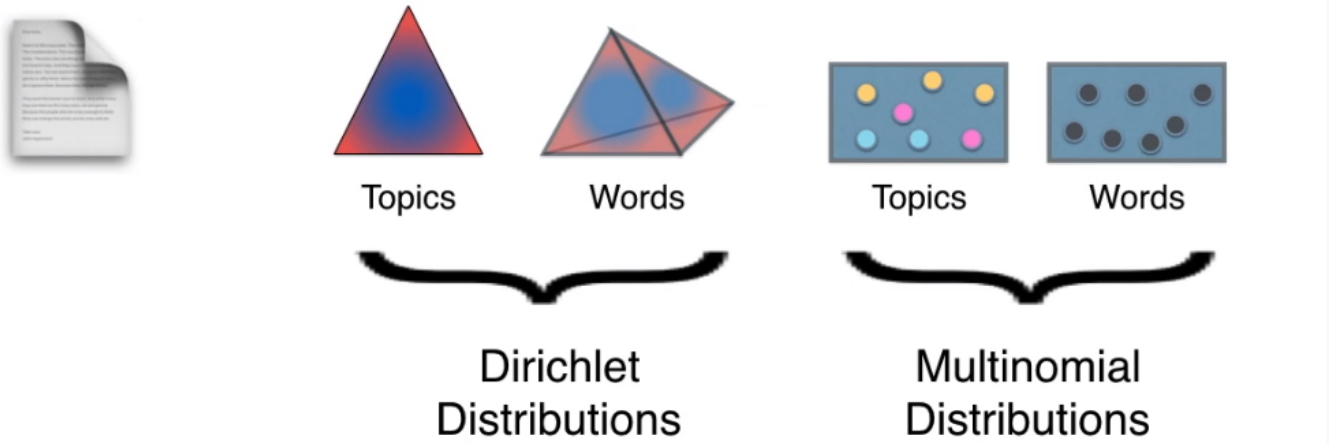



(Ref: Natural Language Processing - Luis Serrano)

# Settings in generator machines

Settings are nothing but two Dirichlet Distributions (doc to topic and topic to words) and then two multinomial distributions (doc to topic and topic to words).

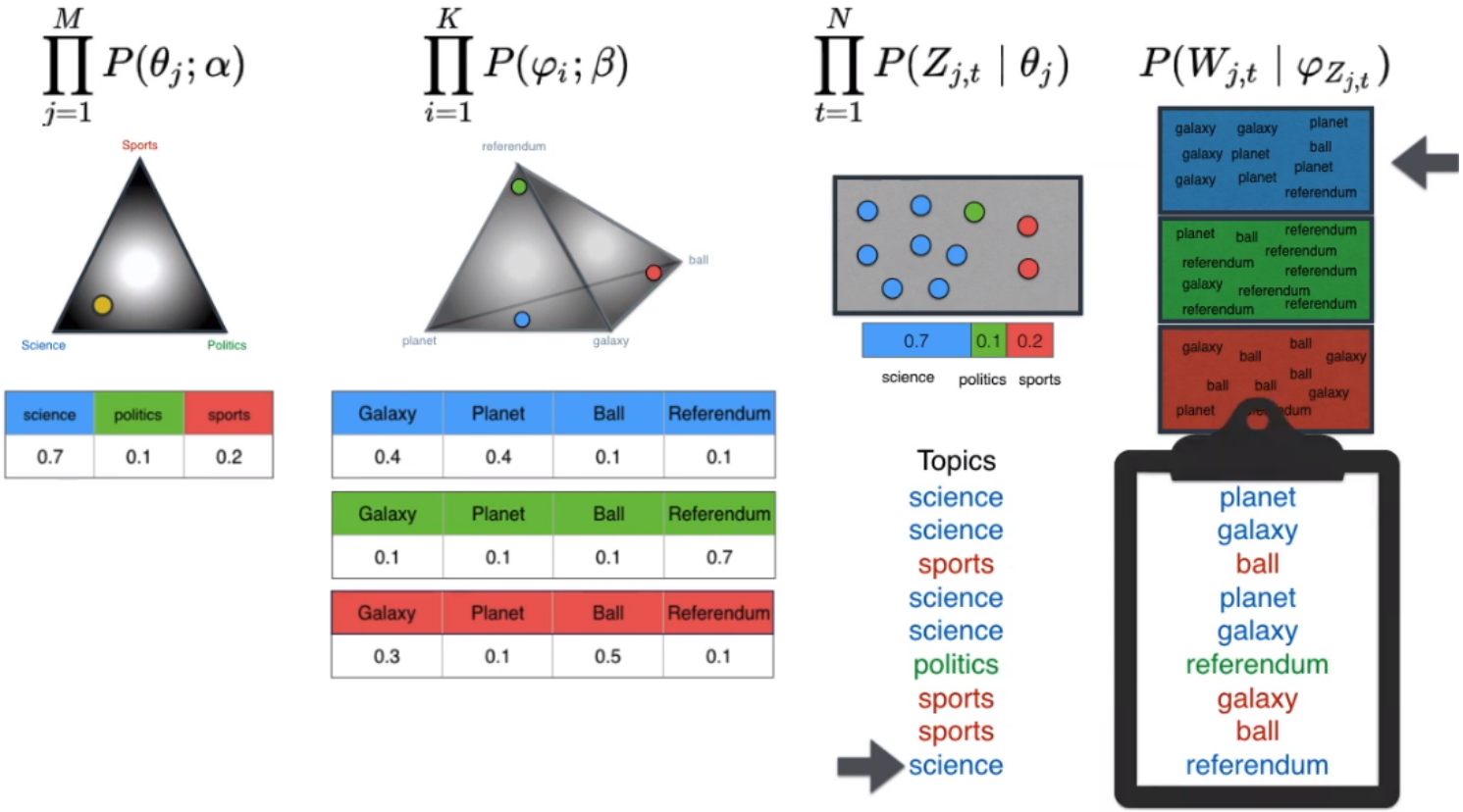
$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$




(Note: for 4 words, square is not chosen for Dirichlet distribution as its diagonal points are not equidistant, so tetrahedron is chosen)

# Example generation

Example document is generated.




(Ref: Natural Language Processing - Luis Serrano)

# Example generation

Multiple sample documents are generated and checked against the real ones.

$$\prod_{j=1}^{M} P(\theta_j;\alpha) \qquad \prod_{i=1}^{K} P(\varphi_i;\beta) \qquad \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) \qquad P(W_{j,t} \mid \varphi_{Z_{j,t}})$$





P(same articles) = low


The settings which give original artical back, ie with better probability are chosen.

(Ref: Natural Language Processing - Luis Serrano)

# Practically

- Topic modeling gives us a way to infer the latent structure behind a collection of documents.
- David Blei invented LDA, and writes well, so if you want to understand why this technique has "Dirichlet" in its name, his works are the next things to read

# How Topics look?

```
>>> lsi_model.show_topics()
'-0.203*"smith" + 0.166*"jan" + 0.132*"soccer" + 0.132*"software" +
     0.119*"fort" + -0.119*"nov" + 0.116*"miss" + -0.114*"opera" +
     -0.112*"oct" + -0.105*"water"',

'0.179*"squadron" + 0.158*"smith" + -0.140*"creek" + 0.135*"chess" +
     -0.130*"air" + 0.128*"en" + -0.122*"nov" + -0.120*"fr" + 0.119*"jan" +
     -0.115*"wales"',

'0.373*"jan" + -0.236*"chess" + -0.234*"nov" + -0.208*"oct" + 0.151*"dec" +
     -0.106*"pennsylvania" + 0.096*"view" + -0.092*"fort" + -0.091*"feb" +
     -0.090*"engineering"',
```

- These three abbreviated topics were extracted from a large corpora of texts by gensim.
- Words are not ordered.
- Positive and negative weights for each word, which get smaller in magnitude as the topic goes on.

# Example

Suppose you have the following set of sentences:

- I eat fish and vegetables.
- Fish are pets.
- My kitten eats fish.

Latent Dirichlet allocation (LDA) is a technique that automatically discovers topics that these documents contain.
LDA achieves the above results in 3 steps. Instead of sentences, imagine you have 2 documents with the following words (cleaned docs):

| Document X | Document Y |
|---|---|
| Fish | Fish |
| Fish | Fish |
| Eat | Milk |
| Eat | Kitten |
| Vegetables | Kitten |

# Algorithm

- Step 1: You tell the algorithm how many topics you think there are.Its just a number. Then topics would have ID's like Topic1, Topic2, etc.
- Step 2: Assign every word to a temporary topic.

```
          Topic1    Topic2    Topic3
word1      1           0         36
Word2      6           3         2
:

           Topic1     Topic2      Topic3
doc1     25            2              11
doc2     3             7              34
:
```

- Step 3: In iteration, for each doc, for each word, its topic assignment is updated based on two criteria:
  - How prevalent is that word across topics?
  - How prevalent are topics in the document?

# Example

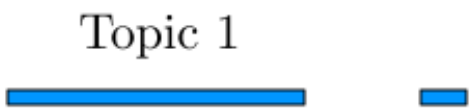
- Following shows two documents X and Y, and number of topics to be be generated is 2.
- Lets call them Topics F (for Food) and P (for Pets)
- Go to every document and for each word write its Topic ID, RANDOMLY.
- Say, starting with first word of Doc Y: "fish" in Doc Y.
- Rest all the topic-word-docs assignments, although RANDOM are ASSUMED to be correct (this method/sampling is called Gibb's sampling)

| | Document X | | Document Y |
|---|---|---|---|
| F | Fish | ? | Fish |
| F | Fish | F | Fish |
| F | Eat | F | Milk |
| F | Eat | P | Kitten |
| F | Vegetables | P | Kitten |

# How prevalent is that word across topics?

- In Doc X, “fish” is with TopicF in both the cases, ie 2/2. It is with TopicP 0 times.
- In (remaining) Doc Y, “fish” is with TopicF in 1 case, ie 1/1. It is with TopicP 0 times.
- Total: “fish” with TopicF 3/3 times and with TopcP 0 times.

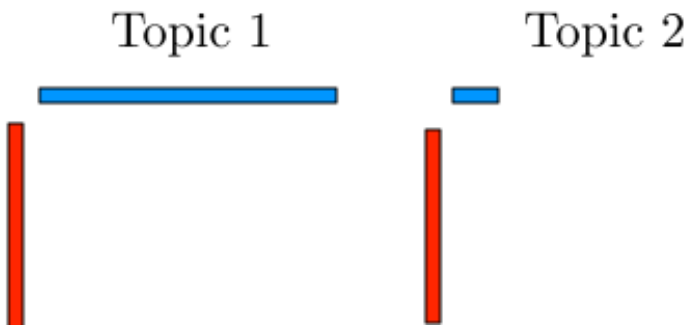
| | Document X | | Document Y |
|---|---|---|---|
| **F** | **Fish** | ? | Fish |
| **F** | **Fish** | **F** | **Fish** |
| F | Eat | F | Milk |
| F | Eat | P | Kitten |
| F | Vegetables | P | Kitten |



# How prevalent are topics in the document?

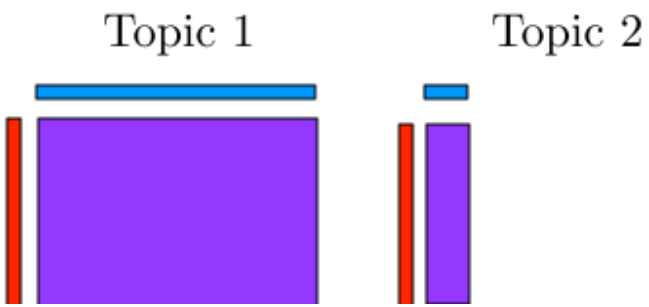
- Now within the single document, ie Doc Y, how are topics distributed?
- 2 TopicFs, 2 TopicPs. 50-50%
- So, their length bars are equal (but vertical!!)

| | Document X | | Document Y |
|---|---|---|---|
| F | Fish | ? | Fish |
| F | Fish | F | Fish |
| F | Eat | F | Milk |
| F | Eat | P | Kitten |
| F | Vegetables | P | Kitten |



# Probability of Assignment

- Multiply horizontal and vertical lengths, get get areas




- Topic1 has more area, so “fish” going to TopcF is proportionally more
- Go back and update the assignment value of “fish” to have more in TopicF.
- Repeating this ‘n’ iterations, words cluster to topics and topics cluster to documents.

# Interesting fact

- Even though value for TopicP was 0, we did not show the bar length to be 0, but some value.
- They are hyper parameters, one in each dimension, alpha and beta. (go back to probability formulation)
- alpha is for value adding to word-topics say 0.8 as smoothing constant
- beta is for value added to doc=topics likings

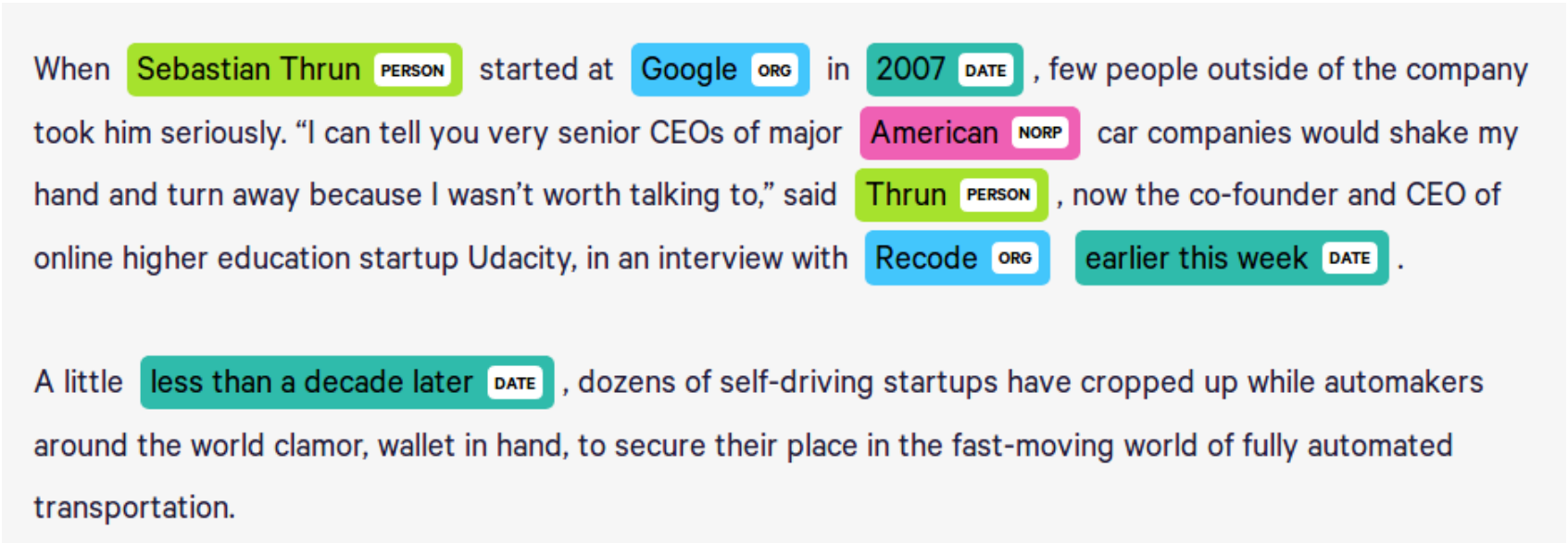
# Results

- In conclusion, we would assign the "fish" word of Doc Y to Topic F. Go to next word (assume remaining assignments are perfect)
- The process of checking topic assignment is repeated for each word in every document, cycling through the entire collection of documents multiple times.
- This iterative updating is the key feature of LDA that generates a final solution with coherent topics.

# Named Entity Recognition

# Introduction

In NLP, NER is a method of extracting the relevant information from a large corpus and classifying those entities into predefined categories such as location, organization, name and so on.




(Ref: Complete Tutorial on Named Entity Recognition (NER) using Python and Keras - Akshay Chavan)

# The who, where, when & how much in a sentence

The task: identify atomic elements of information in text

- Person names
- Company/organization names
- Locations
- Dates & times
- Percentages
- Monetary amounts

The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation

# Example

| Labels | Labels Meaning | Example |
|---|---|---|
| geo | Geographical Entity | Dubai |
| org | Organization | New York Times |
| per | Person | Harry |
| gpe | Geopolitical Entity | European |
| tim | Time Indicator | Friday |
| art | Artifact | Economics |
| eve | Event | Olympic |
| nat | Natural Phenomenon | Hurricane |
| O | Other | of, an, the |

(Ref: Complete Tutorial on Named Entity Recognition (NER) using Python and Keras - Akshay Chavan)

# Difficulties

- Too numerous to include in dictionaries
- Changing constantly: new names invent unknown words
- Appear in many variant forms: e.g. John Smith, Mr Smith, John
- Subsequent occurrences might be abbreviated
- List search/matching does't perform well

## Concept

Whether a phrase is a proper name, and what name class it has, depends on

- Internal structure:“Mr. London”
- Context:“The new location, London, will make a better place.”

# Applications

- Information Extraction: relations are associations between named entities
- Named entities can be indexed, linked off, etc.
- Sentiment can be attributed to companies or products
- Summary generation
- Machine Translation
- Document organization/classification
- Increase accuracy of Internet search results (location Clinton/South Carolina vs. President Clinton)
- For question answering, answers are often named entities

# Standard Approaches

- Hand-written regular expressions: Perhaps stacked
- Using classifiers: Naïve Bayes, Maxent models
- Sequence models:HMMs, CRFs

| Traditional | Statistical | Neural |
|---|---|---|
| • Rules<br>• Regular Expressions<br>• Gazetteers | • Word-based models – PMI, log-likelihood.<br>• Sequence models – Conditional Random Fields | • Bi-LSTM<br>• Bi-LSTM+CRF<br>• Transformer based Models |

Note: Gazetteers are dictionaries or lookup lists.

(Ref: Sujit Pal: Building Named Entity Recognition Models Efficiently Using NERDS — PyData LA 2019)

# The hand-crafted/Rule-based approach

Uses hand-written context-sensitive reduction rules:

- For certain restricted, common types of entities in unstructured text, simple regex patterns also usually work.
- Finding (US) phone numbers
- `(?:\(?[0-9]{3}\)?[ -.])?[0-9]{3}[ -.]?[0-9]{4}`
- Title capitalized word : title person_name compare "Mr. Jones" vs. "Mr. Ten-Percent" : no rule without exceptions
- person_name, "the" adj* "CEO of" organization "Fred Smith, the young dynamic CEO of BlubbCo" : ability to grasp non-local patterns

Plus help from databases of known named entities

## Methods for Sequence Labeling

Typically the following methods are used for NER:

- Hidden Markov Model (HMM)
- Maximum Entropy Classifier (MaxEnt)
- Maximum Entropy Markov Model (MEMM)
- Conditional Random Fields (CRF)

These are all classifiers (i.e., supervised learning) which model sequences (rather than individual random variables)

## Sequence approach

Training

- Collect a set of representative training documents
- Label each token for its entity class or other (O)
- Design feature extractors appropriate to the text and classes 3. Design feature extractors appropriate to the text and classes
- Train a sequence classifier to predict the labels from the data

Testing

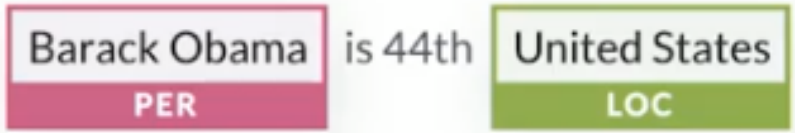
- Receive a set of testing documents
- Run sequence model inference to label each token
- Appropriately output the recognized entities

# Features for sequence labeling

- Words:
  - Current word (essentially like a learned dictionary)
  - Previous/next word (context)
- Other kinds of inferred linguistic classification: Part-of-speech tags
- Label context: Previous (and perhaps next) label
- Word Shapes: Map words to simplified representation that encodes attributes such as length, capitalization, numerals, Greek letters, internal punctuation, etc.

| | |
|---|---|
| Varicella-zoster | Xx-xxx |
| mRNA | xXXX |
| CPA1 | XXXd |

# BIO or IOB format




- **BIO** – **B**egin **I**n **O**ut.
  - Barack/**B-PER** Obama/**I-PER** is/O 44th/O United/**B-LOC** States/**I-LOC** President/O ./O
- **BILOU** – a tagging variant:
  - **U** – **Unit** token (for single token entities)
  - **L** – **Last** token in sequence, ex. Barack/B-PER Obama/L-PER

CoNLL

| | |
|---|---|
| Barack | B-PER |
| Obama | I-PER |
| is | O |
| 44th | O |
| United | B-LOC |
| States | I-LOC |
| President | O |
| . | O |

(Ref: Sujit Pal: Building Named Entity Recognition Models Efficiently Using NERDS — PyData LA 2019)

# Features and Output Data format

| | IO encoding | IOB encoding |
|---|---|---|
| Fred | PER | B-PER |
| showed | O | O |
| Sue | PER | B-PER |
| Mengqiu | PER | B-PER |
| Huang | PER | I-PER |
| 's | O | O |
| new | O | O |
| painting | O | O |

| VBG | NN | IN | DT | NN | IN | NN |
|---|---|---|---|---|---|---|
| Chasing | opportunity | in | an | age | of | upheaval |

POS tagging

| PERS | O | O | O | ORG | ORG |
|---|---|---|---|---|---|
| Murdoch | discusses | future | of | News | Corp. |

Named entity recognition

| B | B | I | I | B | I | B | I | B | B |
|---|---|---|---|---|---|---|---|---|---|
| 而 | 相 | 对 | 于 | 这 | 些 | 品 | 牌 | 的 | 价 |

Word segmentation


Text segmentation

# Dataset example

CoNLL-2003

- CoNLL - Conference on Natural Language Learning by the ACL's Special Interest Group on Natural Language Learning
- Shared Task: Language-Independent Named Entity Recognition
- Goal: Identify boundaries and types of named entities

| Word | POS | Chunk | EntityType |
|---|---|---|---|
| U.N. | NNP | I-NP | I-ORG |
| official | NN | I-NP | O |
| Ekeus | NNP | I-NP | I-PER |
| heads | VBZ | I-VP | O |
| for | IN | I-PP | O |
| Baghdad | NNP | I-NP | I-LOC |
| . | . | O | O |

- Inputs: $x = (x_1, \ldots, x_n)$
- Labels: $y = (y_1, \ldots, y_n)$
- Typical goal: Given $x$, predict $y$

# Algorithms

# Maximum Entropy Markov Model (MEMM)

- (MEMM) classifier makes a single decision at a time, conditioned on evidence from observations and previous decisions.
- Scoring individual labeling decisions is no more complex than standard classification decisions. Using features for classification.

Local Context

Decision Point

| -3 | -2 | -1 | 0 | +1 |
|---|---|---|---|---|
| DT | NNP | VBD | ??? | ??? |
| The | Dow | fell | 22.6 | % |

(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)

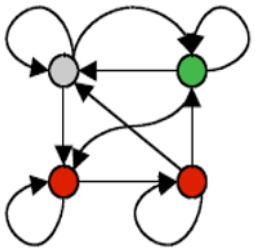
Features

| | |
|---|---|
| $W_0$ | 22.6 |
| $W_{+1}$ | % |
| $W_{-1}$ | fell |
| $T_{-1}$ | VBD |
| $T_{-1}$-$T_{-2}$ | NNP-VBD |
| hasDigit? | true |
| ... | ... |

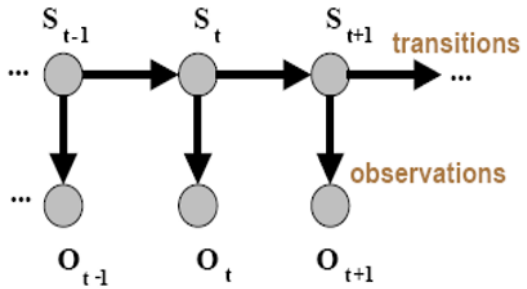
# Hidden Markov Models (HMMs)

**HMMs are the standard sequence modeling tool in genomics, music, speech, NLP, ...**

**Finite state model**



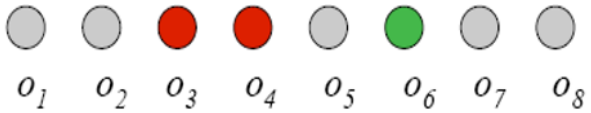
**Graphical model**




Generates:

State sequence

Observation sequence




$$P(\bar{s},\bar{o}) \propto \prod_{t=1}^{|\bar{o}|} P(s_t \mid s_{t-1}) P(o_t \mid s_t)$$

**Parameters:** for all states $S=\{s_1, s_2, \ldots\}$

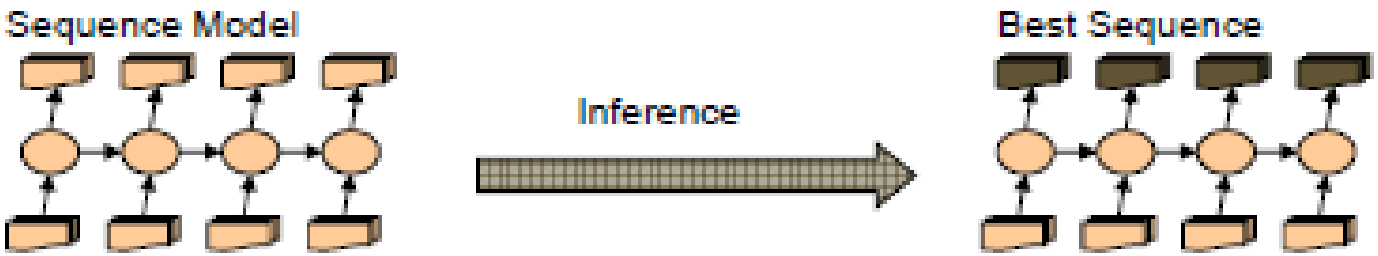
- Start state probabilities: $P(s_t)$
- Transition probabilities: $P(s_t|s_{t-1})$
- Observation (emission) probabilities: $P(o_t|s_t)$ Usually a multinomial over atomic, fixed alphabet

**Training:**

- Maximize probability of training observations (w/ prior)

# Viterbi Inference

- Dynamic programming or memoization.
- Requires small window of state influence (e.g., past two states are relevant).
- Advantage:: Exact: the global best sequence is returned.
- Disadvantage: Harder to implement long-distance state-state interactions (but beam inference tends not to allow long-distance resurrection of sequences anyway).



# Conditional Random Fields (CRFs)

- CRFs are used for predicting the sequences that use the contextual information to add information which will be used by the model to make a correct prediction.
- The output sequence is modeled as the normalized product of the feature function.
- Actually its similar to basic Logistic Regression, done for each token, with a custom feature lists, having info of the context.

$$p(y|X,\lambda) = \frac{1}{Z(X)} exp \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(X, i, y_{i-1}, y_i)$$

# Conclusions

## Hand-crafted vs. Automated

Hand-made systems:

- Can achieve higher performance than ML systems
- Non-local phenomena best handled by regular expressions
- Several person-months for rule-writing, requires experienced linguists
- Rules depend on specific properties of language, domain & text format
- Manual adaption necessary when domain changes
- Re-write rules for other languages

# Hand-crafted vs. Automated

Automated approaches:

- Train on human-annotated texts
- No expensive computational linguists needed
- 1,00,000 words can be tagged in 1-3 days
- Ideally, no manual work required for domain changes
- Easier to port to other languages
- Features are locally limited

# Evaluation for NER

- Recall and precision are straightforward for tasks like IR and text categorization, where there is only one grain size (documents)
- The measure behaves a bit funnily for IE/NER when there are boundary errors (which are common):
- First Bank of Chicago announced earnings
- This counts as both a FP and a FN
- Selecting nothing would have been better
- Some other metrics (e.g., MUC scorer) give partial credit (according to complex rules)

# NER with NLTK

# Steps for NER

- Take a string as input
- Tokenize it into sentences
- Tokenize the sentences into words
- Add part-of-speech tags to the words using `nltk.pos_tag()`
- Run this through the NLTK-provided NER classifier using `nltk.ne_chunk()`
- Parse these intermediate results and return any extracted entities

# NLTK NER Chunker

- `ne_chunk` needs part-of-speech annotations to add NE labels to the sentence. The output of the `ne_chunk` is a `nltk.Tree` object.

```
from nltk import word_tokenize, pos_tag, ne_chunk
sentence = "Mark and John are working at Google."
print ne_chunk(pos_tag(word_tokenize(sentence)))
"""
(S
  (PERSON Mark/NNP)
  and/CC
  (PERSON John/NNP)
  are/VBP
  working/VBG
  at/IN
  (ORGANIZATION Google/NNP)
  ./.)
"""
```

# Steps for NER

- NLTK provides a classifier that has already been trained to recognize named entities, accessed with the function `nltk.ne_chunk()`.
- If we set the parameter `binary=True`, then named entities are just tagged as `NE`; otherwise, the classifier adds category labels such as `PERSON, ORGANIZATION, and GPE.`

```
 >>> print(nltk.ne_chunk(sent))
(S
  The/DT
  (GPE U.S./NNP)
  is/VBZ
  one/CD
  ...
  according/VBG
  to/TO
  (PERSON Brooke/NNP T./NNP Mossman/NNP)
  ...)

```

# IOB tagging

The IOB Tagging system contains tags of the form:

- `B-{CHUNK_TYPE}` - for the word in the Beginning chunk
- `I-{CHUNK_TYPE}` - for words Inside the chunk
- `O` - Outside any chunk

# IOB tagging

```
from nltk.chunk import conlltags2tree, tree2conlltags
sentence = "Mark and John are working at Google."
ne_tree = ne_chunk(pos_tag(word_tokenize(sentence)))
iob_tagged = tree2conlltags(ne_tree)
print iob_tagged
"""[('Mark', 'NNP', u'B-PERSON'), ('and', 'CC', u'O'), ('John', 'NNP',
     u'B-PERSON'), ('are', 'VBP', u'O'), ('working', 'VBG', u'O'), ('at',
     'IN', u'O'), ('Google', 'NNP', u'B-ORGANIZATION'), ('.', '.', u'O')]
"""
ne_tree = conlltags2tree(iob_tagged)
print ne_tree
""" (S
  (PERSON Mark/NNP)
  and/CC
  (PERSON John/NNP)
  are/VBP
  working/VBG
  at/IN
  (ORGANIZATION Google/NNP)
  ./.)
"""
```

# NLTK.Stanford NER

```
wget "http://nlp.stanford.edu/software/stanford-ner-2014-06-16.zip"
unzip stanford-ner-2014-06-16.zip
mv stanford-ner-2014-06-16 stanford-ner
sudo mv stanford-ner /usr/share/

from nltk import word_tokenize
from nltk.tag.stanford import NERTagger

classifier =
     '/usr/share/stanford-ner/classifiers/english.all.3class.distsim.crf.ser.gz'
jar = '/usr/share/stanford-ner/stanford-ner.jar'
st = NERTagger(classifier,jar)
sentence = word_tokenize("Rami Eid is studying at Stony Brook University in
     NY")
print st.tag(sentence)
```

# References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:


- Nltk - Steven Bird, Ewan Klein, Edward Loper
- Machine Learning for Natural Language Processing - Traian Rebedea, Stefan Ruseti - LeMAS 2016 - Summer School
- Natural Language Processing with Python - District Data Labs
- Natural Language Processing+ Python - Ann C. Tan-Pohlmann
- Applied Natural Language Processing - Barbara Rosario
- Named Entity Recognition - Stephan Lesch
- Galvanize NLP
- Text Mining - Behrang QasemiZadeh
- Natural Language Processing - Information Extractoin, Christopher Manning
- Topic Modeling - (William Bert, Megan R Brett, Ted Underwood, Algobeans, Shivam Bansal)
- Word2Vec - (Girish K.,Sivan Biham & Adam Yaari, Adrian Colyer)

Thanks ... yogeshkulkarni@yahoo.com