

INTRODUCTION TO TEXT MINING

Yogesh Kulkarni

January 24, 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.

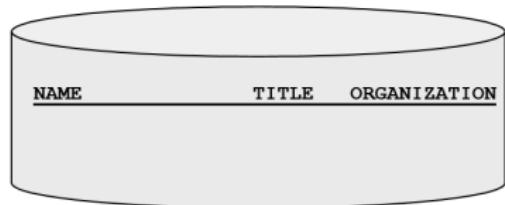
October 14, 2002, 4:00 a.m. PT

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

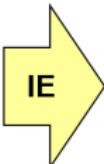
October 14, 2002, 4:00 a.m. PT

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...



NAME	TITLE	ORGANIZATION
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	founder	Free Soft...

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

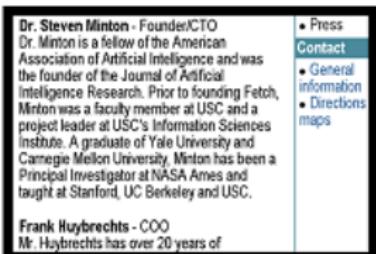
Resumes



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.

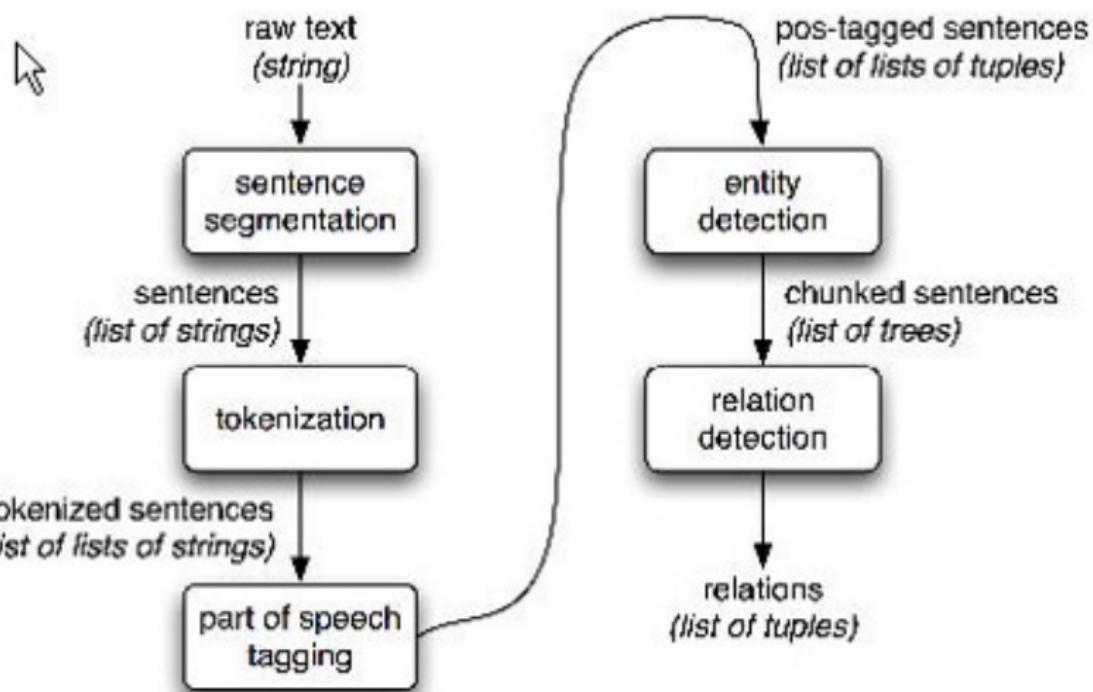
Pawel Opalinski, Software Engineer at WhizBang Labs.

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

Elon Musk PERSON apparently wasn't aware that his company SpaceX had a Facebook ORG page. The SpaceX and Tesla PRODUCT CEO has responded to a comment on Twitter ORG calling for him to take down the SpaceX, Tesla and Elon Musk ORG official pages in support of the #deletefacebook movement by first ORDINAL acknowledging he didn't know one existed, and then following up with promises that he would indeed take them down.

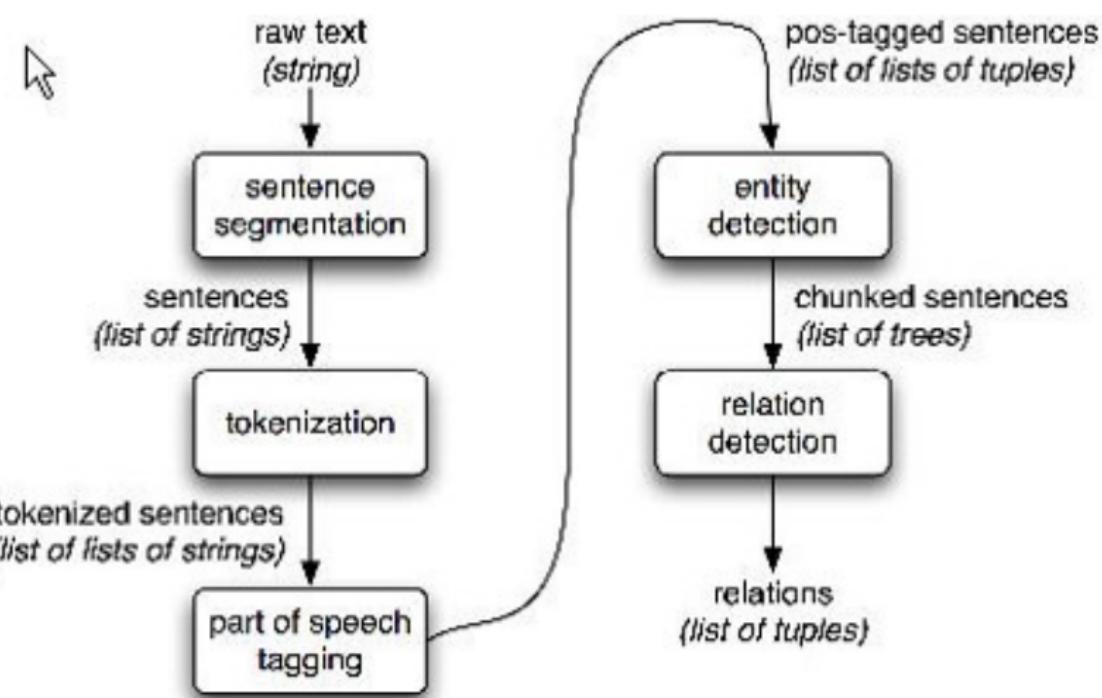
He's done just that, as the SpaceX NNP Facebook page is now gone, after having been live earlier today DATE (as you can see from the screenshot included taken at around 12:10 PM ET) TIME .

Example of Information Extraction

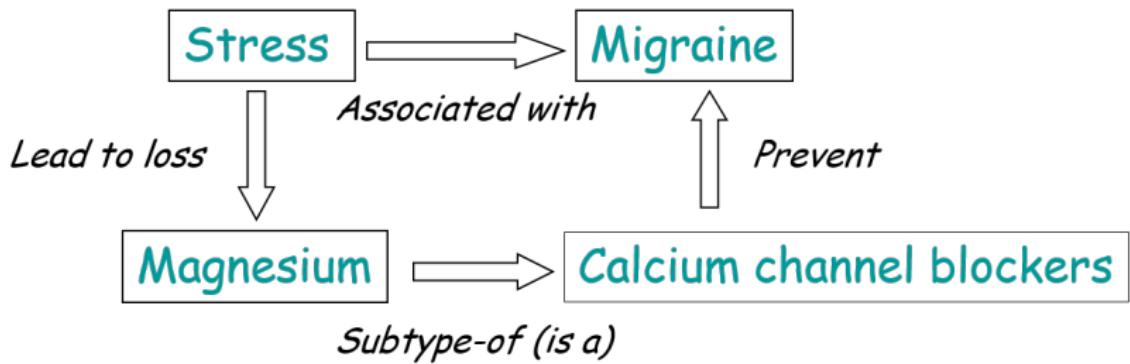
Text:

- ▶ 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

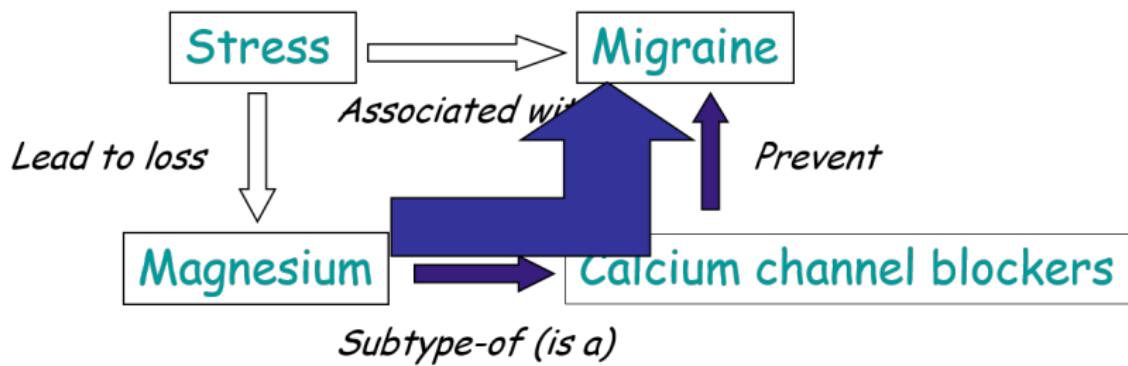
Extract semantic entities from text



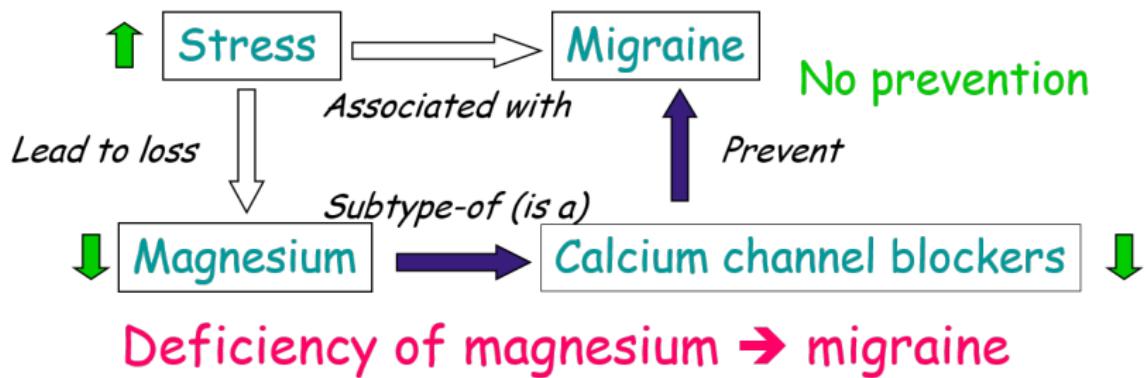
Classify relations between entities



Do reasoning: find new correlations



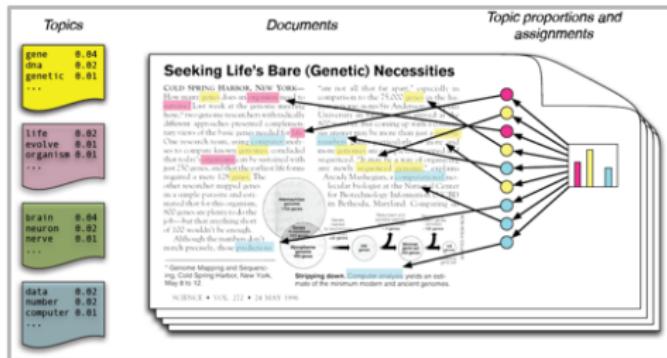
Do reasoning: infer causality



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.

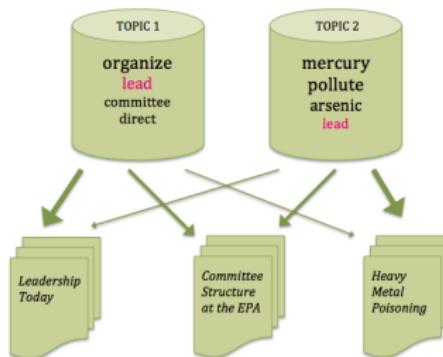
How does it work?

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?

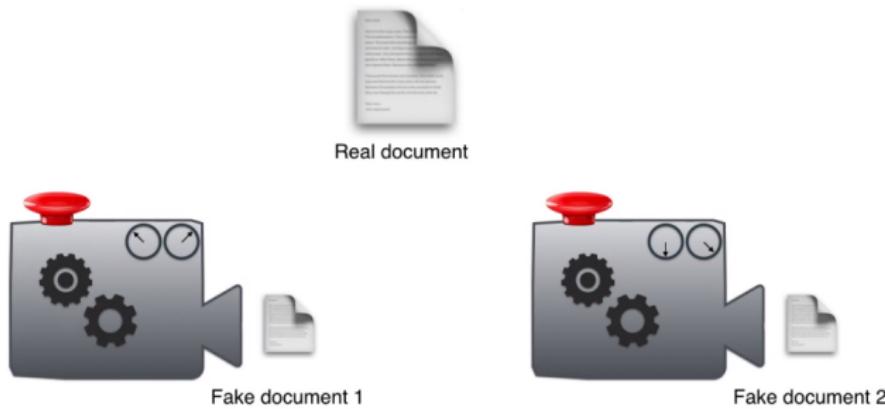
- ▶ Can't directly observe topics; what we have are documents
- ▶ Topic modeling is a way of extrapolating backward to infer the discourses ("topics") that could have generated them.
- ▶ How are we going to decide whether this occurrence of W belongs to topic Z?
- ▶ But one way to guess is to consider 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:

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

- ▶ 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).

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

Topics Words

Topics Words

Dirichlet Distributions Multinomial Distributions

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

Example generation

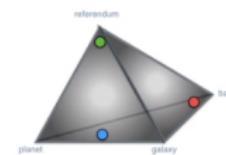
Example document is generated.

$$\prod_{j=1}^M P(\theta_j; \alpha)$$



science	politics	sports
0.7	0.1	0.2

$$\prod_{i=1}^K P(\varphi_i; \beta)$$

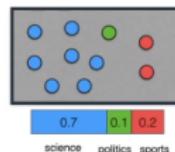


Galaxy	Planet	Ball	Referendum
0.4	0.4	0.1	0.1

Galaxy	Planet	Ball	Referendum
0.1	0.1	0.1	0.7

Galaxy	Planet	Ball	Referendum
0.3	0.1	0.5	0.1

$$\prod_{t=1}^N P(Z_{j,t} | \theta_j)$$



$$P(W_{j,t} | \varphi_{Z_{j,t}})$$

galaxy	galaxy	planet
galaxy	planet	ball
galaxy	planet	planet
referendum		referendum

planet	ball	referendum
referendum	referendum	referendum
galaxy	referendum	referendum

galaxy	ball	ball
ball	ball	galaxy
planet	ball	referendum



Topics

science

science

sports

science

science

politics

sports

sports

science

planet

galaxy

ball

planet

galaxy

referendum

galaxy

ball

referendum

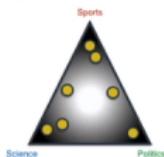


(Ref: Natural Language Processing - Luis Serrano)

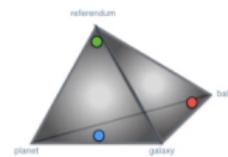
Example generation

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

$$\prod_{j=1}^M P(\theta_j; \alpha)$$

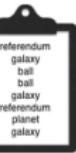


$$\prod_{i=1}^K P(\varphi_i; \beta)$$



$$\prod_{t=1}^N P(Z_{j,t} | \theta_j) \quad P(W_{j,t} | \varphi_{Z_{j,t}})$$

$P(\text{same articles}) = \text{low}$



The settings which give original article 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?

```
1 >>> lsi_model.show_topics()  
2 '-0.203*"smith" + 0.166*"jan" + 0.132*"soccer" + 0.132*"software" +  
3   0.119*"fort" + -0.119*"nov" + 0.116*"miss" + -0.114*"opera" +  
4   -0.112*"oct" + -0.105*"water",  
5  
6 '0.179*"squadron" + 0.158*"smith" + -0.140*"creek" + 0.135*"chess" +  
7   -0.130*"air" + 0.128*"en" + -0.122*"nov" + -0.120*"fr" + 0.119*"jan" +  
8   -0.115*"wales",  
9  
10 '0.373*"jan" + -0.236*"chess" + -0.234*"nov" + -0.208*"oct" + 0.151*"dec" +  
11   -0.106*"pennsylvania" + 0.096*"view" + -0.092*"fort" + -0.091*"feb" +  
12   -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
2 word1	1	0	36
Word2	6	3	2
4 :			
6	Topic1	Topic2	Topic3
doc1	25	2	11
8 doc2	3	7	34
9 :			

- ▶ 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 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 TopicP 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

Topic 1

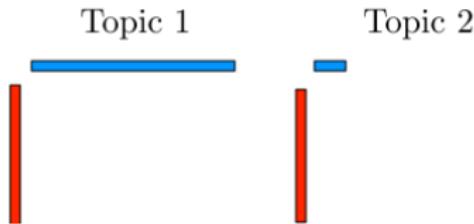
Topic 2



How prevalent are topics in the document?

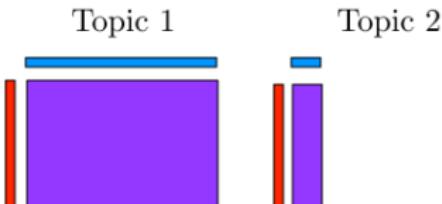
- Now within the single document, ie Doc Y, how are topics distributed?
- 2 TopicFs, 2 TopicPs. 50-50%
- So, their length bars are equal (but vertical!!)

	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 areas



- ▶ Topic1 has more area, so “fish” going to TopicF 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.

Topic Modeling using Gensim

Recap LDA via Matrix

- ▶ In vector space, any corpus (collection of documents) can be represented as a document-term matrix.
- ▶ The following matrix shows a corpus of N documents D₁, D₂, D₃ ... D_n and vocabulary size of M words W₁, W₂ ... W_n.
- ▶ The value of i,j cell gives the frequency count of word W_j in Document D_i.

	W1	W2	W3	Wn
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
Dn	1	1	3	0

Recap LDA via Matrix

- LDA converts this Document-Term Matrix into two lower dimensional matrices - M1 and M2.
- M1 is a document-topics matrix and M2 is a topic - terms matrix with dimensions (N, K) and (K, M) respectively,
- Where N is the number of documents, K is the number of topics and M is the vocabulary size.

	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
Dn	1	0	1	0

	W1	W2	W3	Wm
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

Parameters of LDA

- ▶ Alpha represents document-topic density. More alpha more topics per document and vice versa.
- ▶ Beta represents topic-word density. More beta more words per topic and vice versa.
- ▶ Number of Topics - Number of topics to be extracted from the corpus. Optimal number of topics by using Kullback Leibler Divergence Score.
- ▶ Number of Topic Terms - Number of terms composed in a single topic. A low number is recommended.

What is Gensim?

- ▶ Gensim is a free Python framework for doing topic modelling.
- ▶ Installation: pip install gensim

```
import gensim  
2 >>> help(gensim)
```

- ▶ We can transform corpora from one vector space to another using models.
- ▶ Transformations can bring out hidden structure in the corpus, such as revealing relationships between words and documents.
- ▶ They can also represent the corpus in a more compact way, preserving much information while consuming fewer resources.
- ▶ A gensim 'transformation' is any object which accepts a sparse document via dictionary notation and returns another sparse document."

Preparing Corpus

- ▶ Start by turning a large set of documents into numeric vectors.
- ▶ A document can be as short as one word, or as long as many pages of text, or anywhere in between.
- ▶ A sentence or a tweet can be a document.
- ▶ In gensim, a corpus is an iterable that returns its documents as sparse vectors (one-hot like).
- ▶ How you generate those vectors from the documents is up to you?
- ▶ If your features are the presence or absence of words, your corpus is in what's called "bucket of words" (BOW) format.
- ▶ Gensim provides a convenience class called `TextCorpus` for creating a such corpus from a text file.

Preparing Corpus

```
texts = [" ".join(file.readlines()[1:]) for file in files]
2 tokenised = [[word.lower() for word in WordPunctTokenizer().tokenize(text) if
    word.lower() not in stopwords]]
dictionary = corpora.Dictionary(tokenised)
4 corpus = [dictionary.doc2bow(text) for text in tokenised]
corpora.MmCorpus.serialize('./corpus.mm', corpus)
6
for word in corpus[0]:
    print dictionary.id2token[word[0]], word[1]
8
```

Vectorization

- ▶ For BOW, we also need a dictionary
- ▶ Dictionary maps feature ids back to features (words).
- ▶ So the vectors indicate the presence of words in particular documents, and the resulting matrix containing these vectors will represent all the words appearing in all the documents.

```
2 >>> corpus = TextCorpus(file_like_object)
>>> print corpus.dictionary
Dictionary(8472 unique tokens)
```

Vectorization

- ▶ One useful transformation that we can generate from our BOW corpus is term frequency/inverse document frequency (TFIDF).
- ▶ Instead of a count of word appearances in a document, we get a score for each word that also takes into account the global frequency of that word.
- ▶ So a word's TFIDF value in a given document increases proportionally to the number of times a word appears in that particular document, but is offset by the frequency of the word in the entire corpus, which helps to control for the fact that some words are generally more common than others.

Vectorization

```
1 >>> tfidf_trans = models.TfidfModel(wiki_corpus, id2word=dictionary)
>>> tfidf_trans[documents]
3 [[(40, 0.23), (6, 0.12), (78, 0.65)], [(39, ...)]
```

- ▶ The `wiki_corpus` TFIDF space as `(word_id, word_weight)` tuples where weight is a positive, normalized float.
- ▶ These documents can be anything, new and unseen, as long as they have been tokenized and put into the BOW representation using the same tokenizer and dictionary $word \rightarrow id$ mappings that were used for the `wiki corpus`.

Gensim Models

Let's now run the LDA algorithm, which actually takes only one line.

```
1 lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,  
2     num_topics=5)  
3 lda.show_topics(topics=-1)
```

Example of LDA: Preparing Documents

Here are the sample documents combining together to form a corpus.

```
1 doc1 = "Sugar is bad to consume. My sister likes to have sugar, but not my  
2     father."  
3 doc2 = "My father spends a lot of time driving my sister around to dance  
4     practice."  
5 doc3 = "Doctors suggest that driving may cause increased stress and blood  
6     pressure."  
7 doc4 = "Sometimes I feel pressure to perform well at school, but my father  
8     never seems to drive my sister to do better."  
9 doc5 = "Health experts say that Sugar is not good for your lifestyle."  
10  
11 # compile documents  
12 doc_complete = [doc1, doc2, doc3, doc4, doc5]
```

Example of LDA: Cleaning and Preprocessing

We will remove the punctuations, stopwords and normalize the corpus.

```
from nltk.corpus import stopwords
2 from nltk.stem.wordnet import WordNetLemmatizer
import string
4 stop = set(stopwords.words('english'))
exclude = set(string.punctuation)
6 lemma = WordNetLemmatizer()
def clean(doc):
8     stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
10    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
        return normalized
12
doc_clean = [clean(doc).split() for doc in doc_complete]
```

Example of LDA: Preparing Document-Term Matrix

To run any mathematical model on text corpus, it is a good practice to convert it into a matrix representation. LDA model looks for repeating term patterns in the entire DT matrix.

```
1 # Importing Gensim
2 import gensim
3 from gensim import corpora
4
5 # Creating the term dictionary of our courpus, where every unique term is
6     assigned an index. dictionary = corpora.Dictionary(doc_clean)
7
8 # Converting list of documents (corpus) into Document Term Matrix using
9     dictionary prepared above.
10 doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
```

Example of LDA: Running LDA Model

Next step is to create an object for LDA model and train it on Document-Term matrix.

```
# Creating the object for LDA model using gensim library
2 Lda = gensim.models.ldamodel.LdaModel

4 # Running and Trainign LDA model on the document term matrix.
ldamodel = Lda(doc_term_matrix, num_topics=3, id2word = dictionary, passes=50)
```

Example of LDA: Results

Next step is to create an object for LDA model and train it on Document-Term matrix.

```
1 print(ldamodel.print_topics(num_topics=3, num_words=3))
[ '0.168*health + 0.083*sugar + 0.072*bad,
3  '0.061*consume + 0.050*drive + 0.050*sister,
 '0.049*pressur + 0.049*father + 0.049*sister]
```

Tips to improve results of topic modeling

- ▶ The results of topic models are completely dependent on the features (terms) present in the corpus.
- ▶ The corpus is represented as document term matrix, which in general is very sparse in nature.
- ▶ Reducing the dimensionality of the matrix can improve the results of topic modelling.
- ▶ Frequency Filter: get rid of all those weak features ie low frequency words.
- ▶ Part of Speech Tag Filter: Only NN* can be kept.
- ▶ Add to stopwords (domain specific)
- ▶ Batch Wise LDA: a corpus can be divided into batches of fixed sizes. Running LDA multiple times on these batches will provide different results, however, the best topic terms will be the intersection of all batches.

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.

When Sebastian Thrun PERSON started at Google ORG in 2007 DATE, few people outside of the company took him seriously. "I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I wasn't worth talking to," said Thrun PERSON, now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode ORG earlier this week DATE.

A little less than a decade later DATE, dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

(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
Organization

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 doesn'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



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
- ▶ 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

Barack Obama is 44th United States President.

Barack Obama	PER
United States	LOC

CoNLL

- **BIO – Begin In Out.**
 - 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**

Barack	B-PER
Obama	I-PER
is	O
44 th	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

Q
A
Q
A
A
A
Q
A

Text
segmen-
tation

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

Algorithms

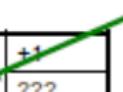
Maximum Entropy Markov Model (MEMM)

- ▶ (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

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

Decision Point



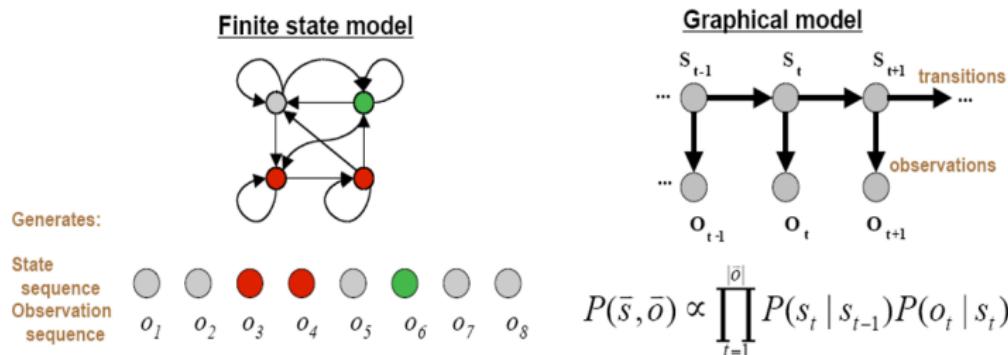
Features

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

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

Hidden Markov Models (HMMs)

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



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

Start state probabilities: $P(s_t)$

Transition probabilities: $P(s_t | s_{t-1})$

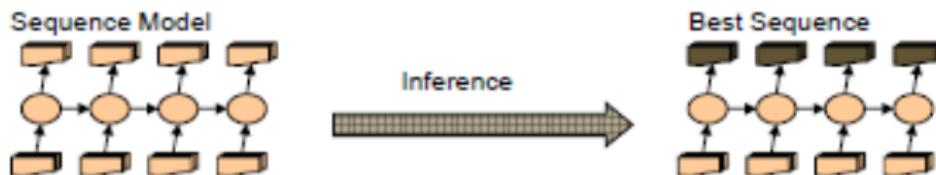
Observation (emission) probabilities: $P(o_t | s_t)$ Usually a multinomial over atomic, fixed alphabet

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.

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

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 funny 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)

Named Entity Recognition (NER) with spaCy

(Ref: <https://spacy.io/usage/spacy-101>)

spaCy

- ▶ SpaCy is an open-source python library for NLP written in Python and Cython.
- ▶ Offers pre-trained models for multi-language NER, as well as allowing developers to train and deploy custom NER models on domain specific corpuses.
- ▶ SpaCy models are designed to be production-ready.
- ▶ Uses 1D residual convolutional neural networks (CNN) and incremental parsing with Bloom embeddings for NER

When Sebastian Thrun PERSON started at Google ORG in 2007 DATE, few people outside of the company took him seriously. "I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I wasn't worth talking to," said Thrun PERSON, now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode ORG earlier this week DATE.

A little less than a decade later DATE, dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

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

(Ref: Google Cloud AI Hub Named Entity Recognition using spaCy and Tensorflow)

spaCy Default NER

Identifies:

- ▶ PERSON: People, including fictional
- ▶ ORG: Companies, agencies, institutions, etc
- ▶ GPE: Countries, cities, states
- ▶ PRODUCT: Objects, vehicles, foods, etc. (Not services.)
- ▶ DATE: Absolute or relative dates or periods
- ▶ TIME: Times smaller than a day
- ▶ PERCENT: Percentage, including "
- ▶ MONEY: Monetary values, including unit
- ▶ QUANTITY: Measurements, as of weight or distance
- ▶ ...

(Ref: Google Cloud AI Hub Named Entity Recognition using spaCy and Tensorflow)

spaCy Default NER

```
1 import spacy
2 nlp = spacy.load('en_core_web_sm')
3
4 doc = nlp("Indians spent over $71 billion on clothes in 2018")
5
6 for ent in doc.ents:
7     print(ent.text, ent.label_)
8
9 Indians NORP
10 over $71 billion MONEY
11 2018 DATE
12
13 spacy.explain("NORP")
14 Output: 'Nationalities religious or political groups'
```

(Ref: spaCy Tutorial to Learn and Master Natural Language Processing (NLP) - Prateek Joshi - Analytics Vidhya)

Named Entity Recognition NER

```
text = 'Apple is looking for buying a U.K. startup. Government has given
       permission.'
2
doc = nlp(text)
4 print(doc)

6 >> Apple is looking for buying a U.K. startup for $1 billion

8 for token in doc:
    print(token.text, token.label_)

10 Apple ORG
12 U.K. GPE
$1 billion MONEY

14 doc = nlp('Apple is looking for buying a UK startup for $1 billion in 2020')
16 displacy.render(doc, style = 'ent')
```

Apple **ORG** is looking for buying a **UK GPE** startup for **\$1 billion MONEY** in **2020 DATE**

Exercise

- ▶ Process the text and create a doc object.
- ▶ Iterate over the doc.ents and print the entity text and label_ attribute.

```
import spacy
2
nlp = spacy.load("en_core_web_sm")
4
text = "It's official: Apple is the first U.S. public company to reach a $1
      trillion market value"
6
# Process the text
doc = ----
8
# Iterate over the predicted entities
10 for ent in ----.----:
    # Print the entity text and its label
    print(ent.----, ----.----)
12
14
```

Custom NER

Training your own NER

- ▶ Either rule-based or machine learning (ML) based.
- ▶ Requires a large amount of labeled training data
- ▶ ML approaches from scratch: Hidden Markov Models, Maximum Entropy, and Conditional Random Fields, as well as deep learning approaches with Recurrent Neural Networks, such as Seq2Seq
- ▶ Need sentence inputs and annotated sentence outputs.
- ▶ May also involve additional feature engineering
- ▶ Some libraries like spaCy and Stanford allow training of custom NER.

NER Datasets

- ▶ Domain-specific (i.e. Twitter, biomedical, advertising, news).
- ▶ i2b2 - Medication, treatments, diseases, risk factors, and medications
- ▶ CoNLL 2003 - English and german news articles annotated with location, organization, person, and miscellaneous

NER Evaluation metrics

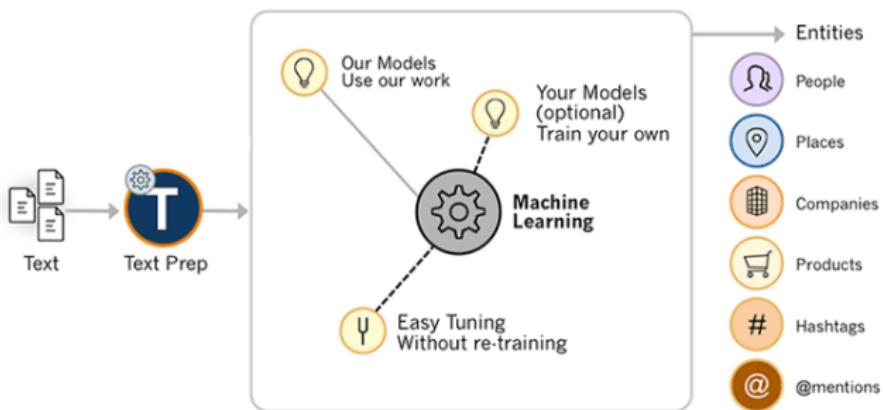
- ▶ NER is most commonly evaluated with precision, recall, and F1-score.

Training your own system

- ▶ The feature extraction works almost identical as the one implemented in the Training a Part-Of-Speech Tagger, except need to add many features.
- ▶ Since the previous IOB tag is a very good indicator of what the current IOB tag is going to be, we have included the previous IOB tag as a feature.
- ▶ spaCy Provides Gold Parse method
- ▶ CRF++ can be used to generate custom NER tags.

spaCy Custom NER

Custom NER Process



(Ref: Training Custom NER - Nishanth N)

Training Data Format

```
1 TRAIN_DATA = [
2     ('Who is Nishanth?', {
3         'entities': [(7, 15, 'PERSON')]
4     }),
5     ('Who is Kamal Khumar?', {
6         'entities': [(7, 19, 'PERSON')]
7     }),
8     ('I like London and Berlin.', {
9         'entities': [(7, 13, 'LOC'), (18, 24, 'LOC')]
10    })
11]
```

(Ref: Training Custom NER - Nishanth N)

NER Pipeline

```
2 nlp = spacy.load(model)
ner = nlp.create_pipe('ner')
4 nlp.add_pipe(ner, last=True)

6 for _, annotations in TRAIN_DATA:
    for ent in annotations.get('entities'):
        8     ner.add_label(ent[2])

10 # Disable pipeline components you dont need to change
pipe_exceptions = ["ner", "trf_wordpiecer", "trf_tok2vec"]
12 unaffected_pipes = [pipe for pipe in nlp.pipe_names if pipe not in
    pipe_exceptions]
```

(Ref: Training Custom NER - Nishanth N)

Training

```
1 # Disable pipeline components you dont need to change
2 other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
3 with nlp.disable_pipes(*other_pipes): # only train NER
4     optimizer = nlp.begin_training()
5     for itn in range(n_iter):
6         random.shuffle(TRAIN_DATA)
7         losses = {}
8         for text, annotations in tqdm(TRAIN_DATA):
9             nlp.update(
10                 [text],
11                 [annotations],
12                 drop=0.5,
13                 sgd=optimizer,
14                 losses=losses)
15             print(losses)
```

(Ref: Training Custom NER - Nishanth N)

Testing

```
1 doc = nlp("I was driving a Alto")
print("Entities", [(ent.text, ent.label_) for ent in doc.ents])
```

(Ref: Training Custom NER - Nishanth N)

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 Extraction, 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