Text

Arthur Charpentier

UQAM

Actuarial Summer School 2019

Natural Language Processing

Imagine you get a sentence in a language you don't understand

フォークで食べてください

What can you do ? Where are the words ?

フォークで食べてください

Language is compositional : natural idea = find regular patterns

$$\underbrace{\text{フォーク}}_{\text{fork}}\text{で}\underbrace{\text{食べ(る)}}_{\text{eat}}\underbrace{\text{てください}}_{\text{please}}$$

$$\underbrace{\text{お箸}}_{\text{chopsticks}}\ \text{で}\underbrace{\text{食べ(る)}}_{\text{eat}}\underbrace{\text{てください}}_{\text{please}}$$

Natural Language Processing

I don't eat fish $\underbrace{\text{魚}}_{\text{fish}}$ は $\underbrace{\text{食べ}}_{\text{eat}}\underbrace{\text{ない}}_{\text{not}}$

I don't eat meat $\underbrace{\text{肉}}_{\text{meat}}$ は $\underbrace{\text{食べ}}_{\text{eat}}\underbrace{\text{ない}}_{\text{not}}$

I don't eat $\underbrace{\text{ブロッコリー}}_{?}$ は $\underbrace{\text{食べ}}_{\text{eat}}\underbrace{\text{ない}}_{\text{not}}$ (broccoli)

I don't eat $\underbrace{\text{早く}}_{?}\underbrace{\text{食べ}}_{\text{eat}}\underbrace{\text{ない}}_{\text{not}}$ (fast)

I don't eat $\underbrace{\text{電車の中で}}_{?}\underbrace{\text{食べ}}_{\text{eat}}\underbrace{\text{ない}}_{\text{not}}$ (in the train)

$\underbrace{\text{電}}_{\text{electricity}}\ \underbrace{\text{車}}_{\text{vehicle}}\ \text{の}\ \underbrace{\text{中で}}_{\text{inside}}$

Natural Language Processing

Same in Chinese

$$\underbrace{\text{小心地滑}}_{?} = \underbrace{\text{小心}}_{\text{be careful}}\quad\underbrace{\text{地滑}}_{\text{ground slip}} = \underbrace{\text{小}}_{\text{small}}\ \underbrace{\text{心}}_{\text{heart}}\ \underbrace{\text{地}}_{\text{ground}}\ \underbrace{\text{滑}}_{\text{slip}}$$

at least in English, words are easily identified

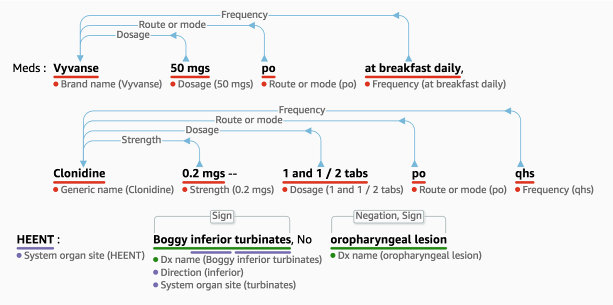
Natural Language Processing

Think the same way when reading a sentence in English,

Bill Gates founded Microsoft
Bill Gates, now retired, founded the famous company Microsoft
Microsoft, founded by Bill Gates, is a big company
Microsoft was founded by Bill Gates
Microsoft was founded not by Larry Page but by Bill Gates

Natural Language Processing

Kaufman *et al.* (2016, Natural Language Processing-Enabled and Conventional Data Capture Methods for Input to Electronic Health Records), Chen *et al.* (2018, A Natural Language Processing System That Links Medical Terms in Electronic Health Record Notes to Lay Definitions) or Simon (2018, Natural Language Processing for Healthcare Customers)



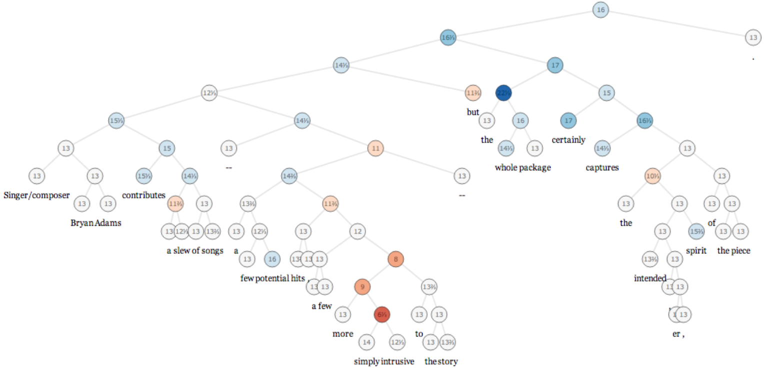
Sentiment Analysis / Classification

Natural language processing technique that identify and quantify affective states and subjective information

Let $\boldsymbol{x}$ denote a sentence (or a general corpus)
and we want $y = m(\boldsymbol{x}) \in \{\text{positive}, \text{negative}, \text{neutral}\}$.
See Pang & Lee (2008, Opinion Mining and Sentiment Analysis)

I liked the movie (+)
I didn't like the movie (-)
I thought I would like that movie
I knew I would like that movie
I hope I will like that movie
I didn't know that movie would be so great

Natural Language Processing



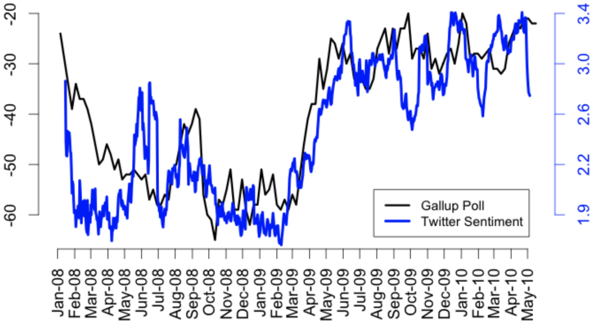
via https://kaggle.com

Natural Language Processing

Need lists of positive / negative words

Stone *et al.* (1966, The General Inquirer: A Computer Approach to Content Analysis), Positiv (1915 words) and Negativ (2291 words) or Wilson *et al.* (2005, Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis), 2718 positive, 4912 negative Baccianella *et al.* (2010, An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining), degrees of positivity, negativity, and neutrality/objectiveness

Natural Language Processing



from O'Connor *et al.* (2010, From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series)

Natural Language Processing

Adjective Polarity, in Hatzivassiloglou & McKeown (1997, Predicting the Semantic Orientation of Adjectives) or Moghaddam & Popowich (2010, Opinion Polarity Identification through Adjectives)
Score Based approach in Whitelaw *et al.* (2005, Using Appraisal Groups for Sentiment Analysis)
If a phrase has better association with the word “Excellent” than with “Poor” it is positively oriented and conversely.

Opinion retrieval started with the work of Hurst and Nigam (2004, Retrieving topical sentiments from online document collections)

Part-of-Speech

PoS (Part-of-Speech)

A part of speech is a category of words (or lexical items) which have similar grammatical properties.
Commonly listed English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection.

PoS (Part-of-Speech) tagging

(also called grammatical tagging or word-category disambiguation)
It is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech based on both its definition and its context.

Part-of-Speech

Nouns : Proper : Microsoft, Switzerland [NNP Proper Noun]
Nouns : Common : snow, cat, cats [NN Noun]
Verbs : Main : see, registered [VB Verbs]
Verbs : Modals : can, had [VB Verbs]
Adjectives : old, older, oldest [JJ Adjective R comparative S superlative]
Adverbs : slowly [RB Adverb]
Numbers : one, 1, million, 123 [CD, Cardinal number]
Prepositions : to, with, under, over, near, by [IN Preposition]
Particles : off, up [RP, Particle]
Interjections : eh, waoo [UH, Interjection]
Determiners : the, some [DT, Determiner]
Conjunctions : and, or [CC Coordinating conjunction]
Pronuns : he, its, I [PRP Personal Pronoun]
Symbols : \$, m^2, $^\circ C$ [SYM Symbol]

Part-of-Speech

one-of-a-kind : JJ Adjective
English cuisine : JJ Adjective
an English sentence : NNP Proper Noun
The back door : JJ Adjective
On my back : NN Noun
Win the voters back : RB Adverb
Promised to back the bill : VB Verb

Greene & Rubin (1971, Automatic Grammatical Tagging of English) - taggit model - Automatic grammatical tagging of English with a deterministic rule, 77% accuracy
Charniak (1993, Statistical Language Learning) : Brown corpus (most frequent tag, and unknown as nouns) 90% accuracy

Part-of-Speech

Mrs	Shaefer	never	got	around	to	joining
NNP	NNP	RB	VBD	RP	TO	VBG

All	we	gotta	do	is	go	around	the	corner
NNP	PRP	VBN	VB	VB	VB	IN	DT	NN

Chateau	Petrus	costs	around	$	250
NNP	NNP	VBZ	RB	SYM	CD

I know that he is honest
that: IN

Yes, that play was nice
that: DT

You can't go that far
that: RB

40% of word tokens are anbiguous...

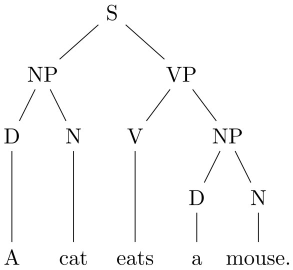
prefixes : unconfortable : JJ

suffixes : confortably : RB

shape : $\underset{\text{xxx}}{35} - \underset{\text{xxx}}{\text{year}}$: JJ

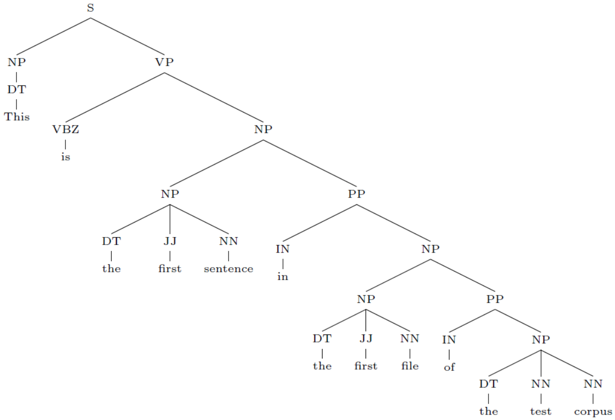
PoS tagging state of the art https://en.wikipedia.org

A classical tool is the constituency parse tree



Part-of-Speech

Can be done in R, see Martin Schweinberger's paRsing function, from openNLP package.



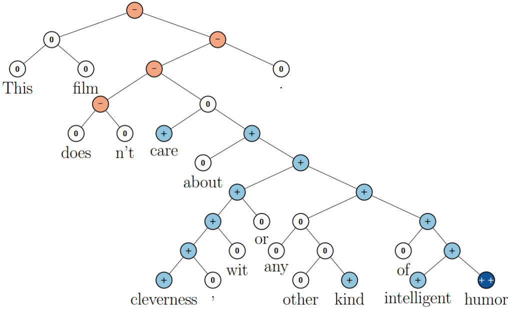
see also https://stackoverflow.com's page

Natural Language Processing

One can use probabilistic context-free grammars,
Nivre (2018, parsing) gave the following probabilities (in English)

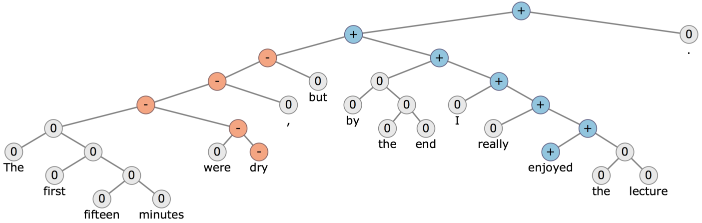
S $\rightarrow$ NP VP	1
NP $\rightarrow$ Pronoun	1/3
NP $\rightarrow$ Proper-Noun	1/3
NP $\rightarrow$ Det Nominal	1/3
Nominal $\rightarrow$ Nominal PP	1/3
Nominal $\rightarrow$ Noun	2/3
VP $\rightarrow$ Verb NP	8/9
VP $\rightarrow$ Verb NP PP	1/9
PP $\rightarrow$ Preposition NP	1

PoS and Sentiment Analysis



via https://codeburst.io/

PoS and Sentiment Analysis



Is it always that simple...?

via Julia Hockenmaier's slides Introduction to NLP,

eat sushi with tuna or eat sushi with chopsticks

$$\underset{V}{\text{eat}}\ \overbrace{\underset{NP}{\text{sushi}}\ \underbrace{\underset{P}{\text{with}}\ \underset{NP}{\text{tuna}}}_{PP}}^{NP} \quad \text{or} \quad \underbrace{\underset{V}{\text{eat}}\ \underset{NP}{\text{sushi}}}_{VP}\ \underbrace{\underset{P}{\text{with}}\ \underset{NP}{\text{chopsticks}}}_{PP}$$

Is it always that simple...?

via Mark Liberman's blog https://languagelog.ldc.upenn.edu,

San Jose cops kill man with knife

Close

Text Paper Translate Listen

San Jose cops kill man with knife

Ex-college football player, 23, shot 9 times allegedly charged police at fiancee's home

By Hamed Aleaziz and Vivian Ho

A man fatally shot by San Jose police officers while allegedly charging at them with a knife was a 23-year-old former football player at De Anza College in Cupertino who was distraught and depressed, his family said Thursday.

Police officials said two officers opened fire Wednesday afternoon on Phillip Watkins outside his fiancee's home because they feared for their lives. The officers had been drawn to the home, officials said, by a 911 call reporting an armed home invasion that, it turned out, had been made by Watkins himself.

But the mother of Watkins' fiancee, who also lives in the home on the 1300 block of Sherman Street, said she witnessed the shooting and described it as excessive. Faye Buchanan said the confrontation happened shortly after she called a suicide intervention hotline in hopes of getting Watkins medical help.

Watkins' 911 call came in at 5:01 p.m., said Sgt. Heather Randol, a San Jose police spokeswoman. "The caller stated there was a male breaking into his home armed with a knife," Randol said. "The caller also stated he was locked in an upstairs bedroom with his children and requested help from police."

She said Watkins was on the sidewalk in front of the home when two officers got there. He was holding a knife with a 4-inch blade and ran toward the officers in a threatening manner, Randol said.

"Both officers ordered the suspect to stop and drop the knife," Randol said. "The suspect continued to charge the officers with the knife in his hand. Both officers, fearing for their safety and defense of their life, fired at the suspect."

On the police radio, one officer said, "We have a male with a knife. He's walking toward us."

"Shots fired! Shots fired!" an officer said moments later.

A short time later, an officer reported, "Male is down. Knife's still in hand."

Buchanan said she had been prompted to call the

Shoot continues on D8

Back Continue

San Jose cops kill man with knife

NP1 V NP2 PP

San Jose cops kill man with knife

NP1 V N PP

$\underbrace{\text{N} \quad \text{PP}}_{\text{NP2}}$

Is it always that simple...?

via https://languagelog.ldc.upenn.edu, in the New York Daily News 8/22/2016:

police kill unarmed deaf man using sign language

which became

police kill unarmed deaf man who was using sign language

or in the Financial Times 1/29/2015:

EU reforms to break up big banks at risk

(are reforms at risk, or are reforms on track to break up banks that are at risk?)

or in the New York Times 13/03/2013:

Researchers Find 25 Countries Using Surveillance Software

Is it always that simple...?

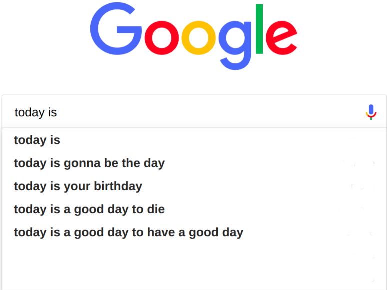
via https://printwand.com/

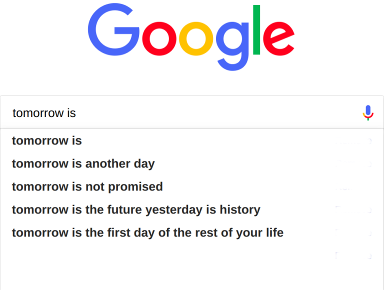
Ensure means guarantee, insure means to protect

Probabilistic Language Models

Idea : $\mathbb{P}[\text{today is Wednesday}] > \mathbb{P}[\text{today Wednesday is}]$

$\mathbb{P}[\text{today is Wednesday}] > \mathbb{P}[\text{today is Wendy}]$





E.g. try to predict the missing word

I grew up in France, I speak fluent __________

Probabilistic Language Models

Bayes Formula

$$\mathbb{P}[A_1, A_2] = \mathbb{P}[A_1] \cdot \mathbb{P}[A_2|A_1]$$

Chain Rule

$$\mathbb{P}[A_1, A_2, A_3] = \mathbb{P}[A_1] \cdot \mathbb{P}[A_2|A_1] \cdot \mathbb{P}[A_3|A_1, A_2]$$

and more generally

Chain Rule

$$\mathbb{P}[A_1, A_2, \cdots, A_n] = \mathbb{P}[A_1] \cdot \mathbb{P}[A_2|A_1] \cdots \mathbb{P}[A_n|A_1, \cdots, A_{n-1}]$$

$$\mathbb{P}[A_1, A_2, \cdots, A_n] = \prod_{i=1}^{n} \mathbb{P}[A_i|A_1, A_2, \cdots, A_{i-1}]$$

Probabilistic Language Models

Unigram model

$$\mathbb{P}[A_1, A_2, \cdots, A_n] \sim \prod_{i=1}^{n} \mathbb{P}[A_i]$$

Markov assumption: bigram model

$$\mathbb{P}[A_1, A_2, \cdots, A_n] \sim \prod_{i=1}^{n} \mathbb{P}[A_i | A_{i-1}]$$

Markov assumption: k-gram model

$$\mathbb{P}[A_1, A_2, \cdots, A_n] \sim \prod_{i=1}^{n} \mathbb{P}[A_i | A_{i-k}, \cdots, A_{i-1}]$$

since language has long-distance dependencies

Use a corpus of sentences, and just count...

$$\mathbb{P}(A_i | A_{i-1}) = \frac{\mathbb{P}(A_{i-1}, A_i)}{\mathbb{P}(A_{i-1})} = \frac{\text{count}(A_{i-1}, A_i)}{\text{count}(A_{i-1})}$$

e.g. Shakespeare 300,000 bigrams

Probabilistic Language Models

$\mathbb{P}(\text{the wine is so good})$? Using bigrams...

the wine is so good
the wine is so good
the wine is so good
the wine is so good
the wine is so good

$$= \mathbb{P}(\text{the}) \cdot \mathbb{P}(\text{wine}|\text{the}) \cdot \mathbb{P}(\text{is}|\text{wine}) \cdot \mathbb{P}(\text{so}|\text{is}) \cdot \mathbb{P}(\text{good}|\text{so})$$

Natural Language Processing

$\boldsymbol{x}$ is a document, and y is a class (for a fixed set of classes $\mathcal{C} = \{c_1, \cdots, c_j\}$).
We have a training dataset $\{(y_i, \boldsymbol{x}_i)\}$
Documents are given by bag-of-words, i.e. the set of words and the occurrences of each term.

Mary is richer than John and John is richer than Mary

are equivalent... The classifier

$$m(\text{Mary is richer than John})$$

can be written

$$m\left(\begin{array}{c|c|c|c|c} \text{Mary} & \text{is} & \text{richer} & \text{than} & \text{John} \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

for a multinomial model, but a binomial model can also be considered

$$m\left(\begin{array}{c|c|c|c|c|c|c|c} \text{Mary} & \text{John} & \text{Greg} & \text{Daisy} & \text{is} & \text{eat} & \text{like} & \text{speak} \\ \hline 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right)$$

A Naive Bayes classifier is such that

$$c^\star \in \underset{c\in\mathcal{C}}{\text{argmax}}\{\mathbb{P}[c|\boldsymbol{x}]\}$$

(maximum *a posteriori*, or most likely class) i.e.

$$c^\star \in \underset{c\in\mathcal{C}}{\text{argmax}}\{\mathbb{P}[\boldsymbol{x}|c]\cdot\mathbb{P}[c]\}$$

(strong) assumption: conditional independence

$$\mathbb{P}[\boldsymbol{x}|c] = \prod_{j=1}^{d}\mathbb{P}[x_j|c]$$

Bag-of-word assumption: positional independence
The maximum likelihood estimator is based on

$$\widehat{\mathbb{P}}[c] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{c_i=c} = \frac{n_c}{n}$$

$$\widehat{\mathbb{P}}[x_j|c] = \frac{1}{n_c}\sum_{i:c_i=c}\mathbf{1}_{x_j\in\boldsymbol{x}_i}$$

Related to Latent Dirichlet Allocation, see Blei, Ng, & Jordan (2003, Latent Dirichlet Allocation) and Landauer & Dumais (1997, A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge)

Word Embeddings

Naive approach : words as atomic symbols

$$\text{insurance} = (0, 0, \cdots, 0, 1, 0, \cdots) \in \{0, 1\}^n$$

With n potentially (very) large. Very very sparse vector

$$\langle \text{insurance}, \text{actuary} \rangle = \langle \text{insurance}, \text{sunrise} \rangle = 0$$

We want to take into account word similarities...
Need to relate them to simple concepts, with appropriate weights (and then use a proper matrix)

	royalty	masculinity	feminity	eatability	fast	$\cdots$
$\boldsymbol{u}_{\text{king}} =$	0.87	0.93	0.05	0.00	0.00	$\cdots$
$\boldsymbol{u}_{\text{queen}} =$	0.83	0.03	0.92	0.00	0.00	$\cdots$

Word Embeddings

Then use a (classical) cosine similarity index,

$$\cos[\boldsymbol{u},\boldsymbol{v}] = \frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\cdot\|\boldsymbol{v}\|}$$

See Levy & Goldberg (2014, Linguistic Regularities in Sparse and Explicit Word Representations) for a discussion on similarity metrics

See also GloVe, Pennington *et al.* (2014, GloVe: Global Vectors for Word Representation) and Word2vect, Mikolov *et al.* (2013, Distributed Representations of Words and Phrases and their Compositionality)

Word Embeddings

Encode single-relational data in a matrix, i.e.
- Synonyms (e.g., from a thesaurus)
- Co-occurrence (e.g., from a general corpus)

Word2Vec: words that are semantically similar often occur near each other in text, based on the idea "*a word is characterized by the company it keeps*", Firth (1957, A synopsis of linguistic theory 1930-1955) i.e. a word's meaning is given by the words that frequently appear close-by

Word Embeddings

When a word w appears in a text, its context is the set of words that appear *nearby* (within a fixed-size window)

Remember what the	King	of Jordan said here at the
make three of a kind, since no	king	can come on the board
today, if you were	king	and assuming we can't
he had dinner with the	King	of France, he dropped
which meant that the	king	also had to have due
city where conformity is	king	and colour is being drained

Apply SVD to the matrix to find latent components

$$\underset{d\times n}{\boldsymbol{M}} = \underset{d\times k}{\boldsymbol{U}}\,\underset{k\times k}{\boldsymbol{\Delta}}\,\underset{k\times n}{\boldsymbol{V}^\top}$$

where $\boldsymbol{\Delta}$ is a diagonal matrix, with $r = \text{rank}(\boldsymbol{M})$ non-zero terms. If $\boldsymbol{\Delta}_k$ has $k < r$ non-null values, $\widetilde{\boldsymbol{M}}_k = \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{V}^\top$ is the best k-rank approximation,

$$\widetilde{\boldsymbol{M}}_k = \text{argmin}\{\|\boldsymbol{M} - \boldsymbol{M}_k\|_{\text{Frobenius}}\},\ \text{s.t. rank}(\boldsymbol{M}_k) = k$$

where $\|\boldsymbol{M}\|_{\text{Frobenius}} = \sqrt{\sum_{i,j} m_{i,j}^2}$

Word similarity $=$ cosine of two column vectors in $\boldsymbol{\Sigma}\boldsymbol{V}^\top$

Natural Language Processing

Word2Vec (or skip-gram) objective : maximize the log-likelihood of some context word

$$\cdots \underbrace{\text{dinner}}_{\mathbb{P}[w_{t-3}|w_t]} \underbrace{\text{with}}_{\mathbb{P}[w_{t-2}|w_t]} \underbrace{\text{the}}_{\mathbb{P}[w_{t-1}|w_t]} \underset{w_t}{\text{king}} \underbrace{\text{of}}_{\mathbb{P}[w_{t+1}|w_t]} \underbrace{\text{France}}_{\mathbb{P}[w_{t+2}|w_t]} \cdots$$

Given m (say 5-10), maximize

$$\frac{1}{T}\sum_{t=1}^{T}\log\mathcal{L}(w_t) \text{ where } \log\mathcal{L}(w_t) = \sum_{j=-m\ (j\neq 0)}^{m} \log p(w_{t+j}|w_t)$$

To model $\log p(w_{t+j}|w_t)$, use

$$p(w_{t+j}|w_t) = \frac{\exp[\langle \boldsymbol{u}_{w_{t+j}}, \boldsymbol{v}_{w_t}\rangle]}{\sum \exp[\langle \boldsymbol{u}_{w}, \boldsymbol{v}_{w_t}\rangle]}$$

where word w is associated to 2 vectors, $\boldsymbol{u}_w$ (center word) and $\boldsymbol{v}_w$ (context / outside word)

$$\cdots \underbrace{\text{dinner}}_{\mathbb{P}[w_{t-3}|\boldsymbol{v}_{w_t}]} \underbrace{\text{with}}_{\mathbb{P}[\boldsymbol{u}_{w_{t-2}}|\boldsymbol{v}_{w_t}]} \underbrace{\text{the}}_{\mathbb{P}[\boldsymbol{u}_{w_{t-1}}|\boldsymbol{v}_{w_t}]} \underset{w_t}{\text{king}} \underbrace{\text{of}}_{\mathbb{P}[\boldsymbol{u}_{w_{t+1}}|\boldsymbol{v}_{w_t}]} \underbrace{\text{France}}_{\mathbb{P}[\boldsymbol{u}_{w_{t+2}}|\boldsymbol{v}_{w_t}]} \cdots$$

Use gradient descent to maximize this log-likelihood, based on

$$\frac{\partial \log p(w_{t+j}|w_t)}{\partial \boldsymbol{v}_{w_t}} = \boldsymbol{u}_{w_t} - \mathbb{E}_{W\ p(W|w_t)}[\boldsymbol{u}_W]$$

given vocabulary set W But ∇J is big ($2\times$ number of words) and very very sparse

Word Embeddings

With a large vocabulary set, stochastic gradient descent is still not enough : approximate it again, remove sample that do not appear in the context Idea of negative sampling : write

$$\log p(w_o|w_c) = \log[\text{sig}(\langle \boldsymbol{u}_{w_o}, \boldsymbol{v}_{w_c}\rangle)] + \sum_{i=1}^{k} \log[\text{sig}(-\langle \boldsymbol{u}_{w_i}, \boldsymbol{v}_{w_c}\rangle)]$$

With $k << $ number of words.

Heuristically, we want to maximize the probability that real outside word appears, and minimize the probability that random word appear around the center word w_c

Why not use co-occurence counts ?

Corpus : I like actuarial science I like deep learning I enjoy jogging

Word Embeddings

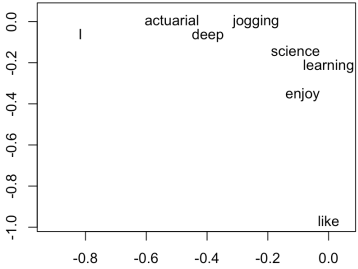
	I	like	enjoy	actuarial	science	deep	learning	jogging
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	0	1
actuarial	0	1	0	0	1	0	0	0
science	0	0	0	1	0	0	0	0
deep	0	1	0	0	0	0	1	0
learning	0	0	0	0	0	1	0	0
jogging	0	0	1	0	0	0	0	0

```r
W <- c("I like actuarial science","I like deep
   learning","I enjoy jogging")
L <- base::strsplit(W," ")
W <- unique(unlist(L))
X <- matrix(0,length(W),length(W))
```

Word Embeddings

Use Singular Value Decomposition of co-occurence matrix, $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{V}^\top$

```
for(i in 1:length(L)){
for(j in 2:(length(L[[i]]))){
    ix <- which(W==L[[i]][j-1])
    jx <- which(W==L[[i]][j])
    X[ix,jx] <- X[ix,jx]+1
    X[jx,ix] <- X[jx,ix]+1 }}
colnames(X) <- rownames(X) <- W
SVD <- svd(X)
plot(SVD$u[,1],SVD$u[,2])
```



with the first two columns of $\boldsymbol{U}$ (two largest singular values)

Word Embeddings

Levy & Goldberg (2014, Neural Word Embedding as Implicit Matrix Factorization) proved that skip-gram model factorize (pointwise mutual information) matrice

$$P_{w,c} = \log \frac{\mathbb{P}[w|c]}{\mathbb{P}[w]} \log \frac{\#(w,c)}{\#(w)\#(c)}$$

That measures the association between a word w and a context c
Using this similarity, rainy and sunny seem to be close Need to take into account polarity, see Yih, Zweig & Platt (2012, Polarity Inducing Latent Semantic Analysis)
Encode two opposite relations (synonyms & antonyms) in a matrix using polarity (see text2vec R package, and the related vignette)

Word Embeddings

As shown in Turian *et al.* (2010, Word representations) Word2Vec is based on a 2-layer neural net

the input is a text, the output is a set of vectors

With enough training data, it can guess meanings based on past appearances

That guess is based on associations between words, using a cosine similarity measure. Neighbors of Sweden are

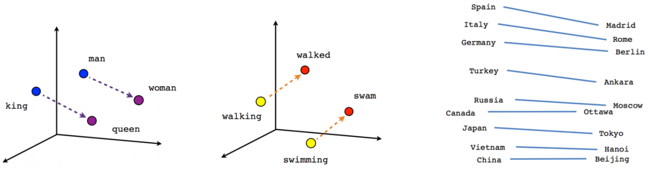
Norway	Denmark	Finland	Switzerland	Belgium	Netherlands	Ic
0.7601	0.7155	0.6200	0.5881	0.5858	0.5746	0

Word2Vec maps words into a numeric (continuous) space...

Similar things or concepts are somehow close in that space

Word Embeddings

A comprehensive geometry of words



Male-Female Verb tense Country-Capital

from Mikholov *et al.* (2013, Linguistic Regularities in Continuous Space Word Representations)
Consider the two dimension PCA projections of countries and capitals: observe that vectors *capital-country* are almost equals, so

$$\text{Beijing} - \text{China} \approx \text{Paris} - \text{France}$$

Word Embeddings

Beijing = China + Paris - France

i.e. Beijing is to China as Paris is to France

In Word2Vec forget +, - and = signs, and use : for *is to* and :: for *as*, e.g

Beijing:China::Paris:France

Some strange relationships have been observed

China:Taiwan::Russia:[Ukraine,Moscow,Moldavia,Armenia]

house:roof::castle:[dome,bell_tower,spire,crenellations,turrets]

knee:leg::elbow:[forearm,arm,ulna_bone]

see http://skymind.ai for more details

Regular Expressions

Regular Expressions

A regular expression is a sequence of characters that define a search pattern.

Introduced by Kleene (1951, Representation of Events in Nerve Nets and Finite Automata)

Regular Expressions

Pattern: [A-Z]

<h2>Patterns[<a href="/w/index.php?title=Regular_expression&action=edit&se
title="Edit section: Patterns">edit]</h2>
<p>The phrase <i>regular expressions</i>, and consequently, <i>regexes</i>, is often used to mean the specific, standard textual syntax (distinct from the mathematical notation described below) for representing patterns for matching text. Each character in a regular expression (that is, each character in the string describing its pattern) is either a metacharacter, having a special meaning, or a regular character that has a literal meaning. For example, in the regex <code>a.</code>, <i>a</i> is a literal character which matches just 'a', while '.' is a meta character that

Pattern: [A-Z]

<h2>Patterns[<a href="/w/index.php?title=Regular_expression&action=edit&se
title="Edit section: Patterns">edit]</h2>
<p>The phrase <i>regular expressions</i>, and consequently, <i>regexes</i>, is often used to mean the specific, standard textual syntax (distinct from the mathematical notation described below) for representing patterns for matching text. Each character in a regular expression (that is, each character in the string describing its pattern) is either a metacharacter, having a special meaning, or a regular character that has a literal meaning. For example, in the regex <code>a.</code>, <i>a</i> is a literal character which matches just 'a', while '.' is a meta character that

Regular Expressions

Pattern: ([A -Z])\w+

\w : matches any word character

<h2>Patterns[<a href="/w/index.php?title=Regular_expression&action=edit&se
title="Edit section: Patterns">edit]</h2>
<p>The phrase <i>regular expressions</i>, and consequently, <i>regexes</i>, is often used to mean the specific, standard textual syntax (distinct from the mathematical notation described below) for representing patterns for matching text. Each character in a regular expression (that is, each character in the string describing its pattern) is either a metacharacter, having a special meaning, or a regular character that has a literal meaning. For example, in the regex <code>a.</code>, <i>a</i> is a literal character which matches just 'a', while '.' is a meta-character that

Regular Expressions

Pattern: [h]\w+

<h2>Patterns[<a href="/w/index.php?title=Regular_expression&action=edit&se
title="Edit section: Patterns">edit]</h2>
<p>The phrase <i>regular expressions</i>, and consequently, <i>regexes</i>, is often used to mean the specific, standard textual syntax (distinct from the mathematical notation described below) for representing patterns for matching text. Each character in a regular expression (that is, each character in the string describing its pattern) is either a metacharacter, having a special meaning, or a regular character that has a literal meaning. For example, in the regex <code>a.</code>, <i>a</i> is a literal character which matches just 'a', while '.' is a meta character that matches every character except a newline

Regular Expressions

Pattern: [a-z]{10,}

<h2>Patterns[<a href="/w/index.php?title=Regular_expression&action=edit&se
title="Edit section: Patterns">edit]</h2>
<p>The phrase <i>regular expressions</i>, and consequently, <i>regexes</i>, is often used to mean the specific, standard textual syntax (distinct from the mathematical notation described below) for representing patterns for matching text. Each character in a regular expression (that is, each character in the string describing its pattern) is either a metacharacter, having a special meaning, or a regular character that has a literal meaning. For example, in the regex <code>a.</code>, <i>a</i> is a literal character which matches just 'a', while '.' is a meta character that matches every character except a newline

Regular Expressions

see http://saa-iss.ch/
http://saa-iss.ch/outline/
contact charpentier.arthur@uqam.ca
or on twitter @freakonometrics

Pattern: @\w+
see http://saa-iss.ch/
http://saa-iss.ch/outline/
contact charpentier.arthur@uqam.ca
or on twitter @freakonometrics

Pattern: \w+@
see http://saa-iss.ch/
http://saa-iss.ch/outline/
contact charpentier.arthur@uqam.ca
or on twitter @freakonometrics

Regular Expressions

Pattern: ^([a-z0-9\\.-]+)@([\\da-z\\.-]+)\\.([a-z\\.]{2,6})$
(see https://regular-expressions.info)
see http://saa-iss.ch/
http://saa-iss.ch/outline/
contact charpentier.arthur@uqam.ca
or on twitter @freakonometrics

Pattern: http://([\\
da-z\\.-]+)\\.([a-z\\.]{2,6})/[a-zA-Z0-9]{1,}
see http://saa-iss.ch/
http://saa-iss.ch/outline/
contact charpentier.arthur@uqam.ca
or on twitter @freakonometrics

Regular Expressions

```
library(stringr)
tweet <- "Emerging #climate change, environment and sustainability concerns for #actuaries Apr 9 #Toronto. Register TODAY http://bit.ly/CIAClimateForum"
hash <- "#[a-zA-Z]{1,}"
str_extract(tweet,hash)
 [1] "#climate"
```

```
str_extract_all(tweet,hash)
 [[1]]
 [1] "#climate"    "#actuaries" "#Toronto"
str_locate_all(tweet,hash)
 [[1]]
      start end
 [1,]    10  17
 [2,]    71  80
 [3,]    88  95
```

Regular Expressions

```
urls <- "http://([\\da-z\\.-]+)\\.([a-z\\.]{2,6})/[a-zA-Z0-9]{1,}"
str_extract_all(tweet,urls)
 [[1]]
 [1] "http://bit.ly/CIAClimateForum"
```

```
email <- "^([a-z0-9_\\.-]+)@([\\da-z\\.-]+)\\.([a-z\\.]{2,6})$"
grep(pattern = email, x = "charpentier.arthur@uqam.ca")
 [1] 1
grepl(pattern = email, x = "charpentier.arthur@uqam.ca")
 [1] TRUE
str_detect(pattern = email, string=c("charpentier.arthur@uqam.ca","@freakonometrics"))
 [1]  TRUE FALSE
```

Regular Expressions

```
ex_sentence <- "This is 1 simple sentence, just to play with, then we'll play with 4, and that will be more difficult"
grep("difficult", ex_sentence)
  [1] 1
word(ex_sentence,4)
  [1] "simple"
word(ex_sentence,1:20)
  [1] "This"       "is"         "1"          "simple"     "sentence," "just"
  [7] "to"         "play"       "with,"      "then"       "we'll"     "play"
 [13] "with"       "4,"         "and"        "that"       "will"      "be"
 [19] "more"       "difficult"
grep(pattern="w",ex_words,value=TRUE)
  [1] "with," "we'll" "with"  "will"
```

```
grep(pattern="[ai]",ex_words,value=TRUE)
  [1] "This"       "is"         "simple"     "play"
    "with,"     "play"
  [7] "with"       "and"        "that"       "will"
    "difficult"
grep(pattern="[[:punct:]]",ex_words,value=TRUE)
  [1] "sentence," "with,"      "we'll"      "4,"
```

```
fix_contractions <- function(doc) {
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'s", "", doc)
  return(doc)
}
```

Regular Expressions

```
library(tm)
ex_corpus <- Corpus(VectorSource(ex_sentence))
ex_corpus <- tm_map(ex_corpus, fix_contractions)
ex_corpus <- tm_map(ex_corpus, tolower)
ex_corpus <- tm_map(ex_corpus,gsub, pattern= "[[:punct
    :]]", replacement = " ")
```

See Ewen Gallic application with tweets...