31.7.-10.8.

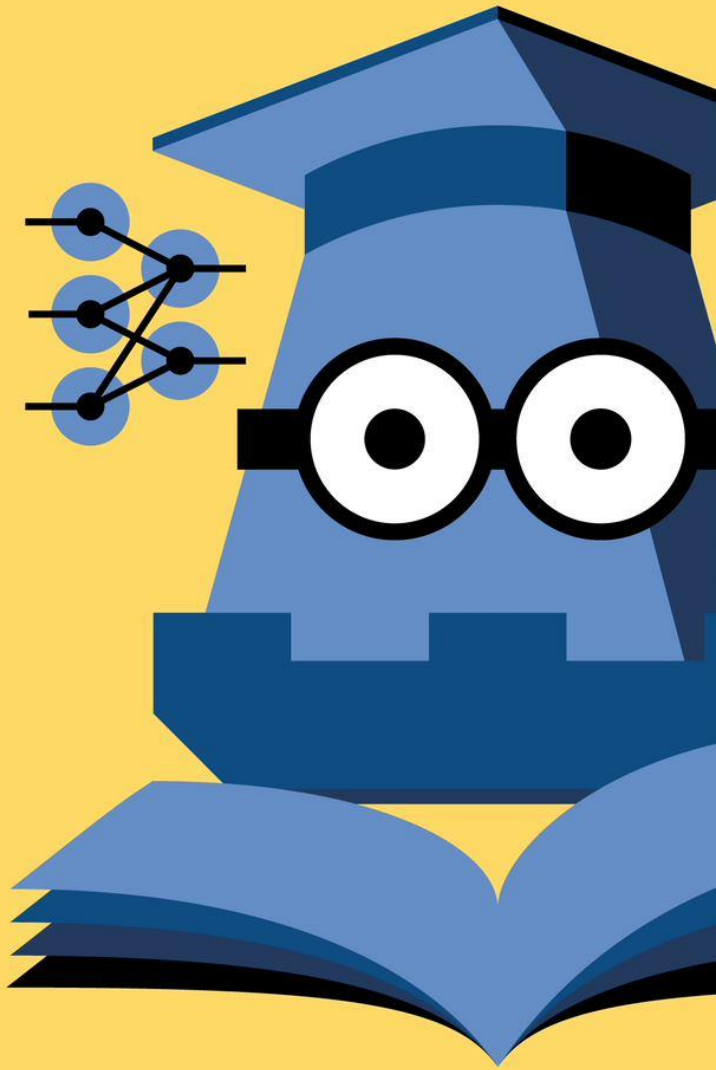
PETNICA SUMMER INSTITUTE

MACHINE LEARNING 6

PETNICA SUMMER INSTITUTE
MACHINE LEARNING 6

Natural Language Processing

- an overview -



Miloš Jovanović

milos.jovanovic@fon.bg.ac.rs

ФОН

УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

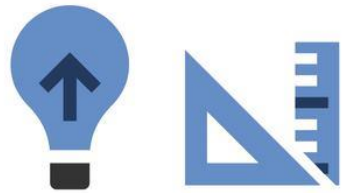
Textual data

Examples:

- Rental description
- Job descirption
- Medical notes
- Address?
  - 911 Thorne Avenue - Cuyahoga Falls, OH 44221
  - 101 Foulois Ave; Barksdale Afb, Louisiana(LA), 71110

Textual data vs Natural language

How much information in textual data?



Overview of NLP tasks

- Text as input - **Use language** (read)

Sentiment analysis

Best roast chicken in San Francisco!

The waiter ignored us for 20 minutes.

Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

Classification (e.g. Spam)

Let's go to Agra! ✓

Millions of DOLLARS… ✗

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30

Party
May 27
add

- Text as output - **Produce language** (write)

Machine translation (MT)

第13届上海国际电影节开幕…

The 13th Shanghai International Film Festival…

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Dialog

Where is Citizen Kane playing in SF?

Castro Theatre at 7:30. Do you want a ticket?

Summarization

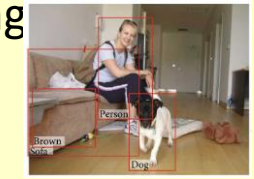
The Dow Jones is up

The S&P500 jumped

Housing prices rose

Economy is good

Captioning

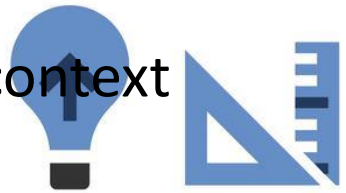


The person is against the brown sofa, and the dog is near her.

*Slide from: Danijela Stepanovic, NLP, PSIML 2018

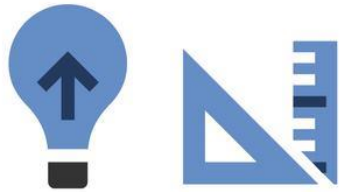
Challenges of NLP

1. Language noise
   - errors
   - multiple ways to say the same thing
   - ...and multiple words
   - syntatic and semantic
   - should express in concepts, not words (people think in concepts, but express in words)
   - Language ambiguity
2. Structure/Order of words
   - appearance of words is not independent (different languages!)
   - single sentence (order very important);
   - long text (order important, and memory);
   - broken sentence - informal (order not as important, common sense context important)



Basic text processing: RegEx

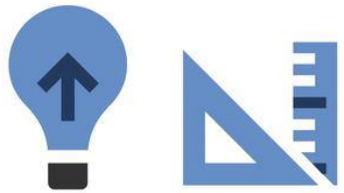
- Regular expressions – a **formal** language for specifying text strings
  - [^A-Za-z]* [Mm]achine [Ll]earning [\d]*
- Text normalization - preprocessing
  1. Capturing generalizations
     - number [\d]+ => @NUMBER
     - email (^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$) => @EMAIL
  2. Defining separators for Tokenizing words in text
     - space, comma, semicolon, ...
  3. Segmenting sentences in text
     - full stop </p> </br>
  4. Normalizing word formats
     - reduce all letters to lower case
     - removing prefixes and modifiers



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Text classification

- Examples:
  - Assigning subject categories or topics
  - Spam detection
  - Authorship identification
  - Age/gender identification
  - Language Identification
  - **Sentiment analysis**
  - …
- Text classification definition:
  - *Input*:
    - document *d*
    - fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$
  - *Output*: predicted class $c \in C$

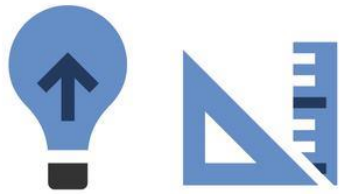


*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Text classification

- Approaches:
  - Hand-coded rules:
    - Rules based on combinations of words or other features
      - spam: black-list-words, phrases or addresses (“dollars” AND “have been selected”)
    - But building and maintaining these rules is expensive

  - Supervised Machine Learning:
    - *Input:*
      - A training set of *m* hand-labeled documents $(d_1,c_1),....,(d_m,c_m)$
    - *Output:*
      - a learned classifier $f: d \rightarrow c$
      - *“rules” on the combination of words or other features*



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Text classification

- Naïve Bayes classifier:
  - For a document *d* and a class *c*

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$
$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$
$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

f (I **love** this movie! It's **sweet**, but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun**… It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.)=c

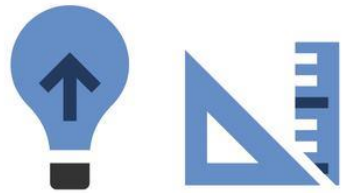
  - **Bag of Words assumption**: Assume position doesn't matter

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots x_n \mid c)P(c)$$

Document d represented as **set** of words x1..xn

  - **Conditional Independence ("naïve")**: Assume the word probabilities $P(x_i|c_j)$ are independent given the class c.

$$= \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{x \in X} P(x \mid c)$$



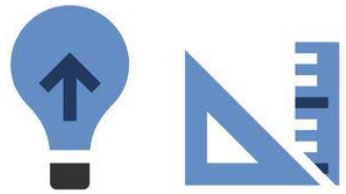
*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Bag-of-words (Vector space) models

- Representation of documents in vector space (One-hot encoding)
- Apply any ML algorithm for tabular data

	MARY	IS	HUNGRY	HAPPY	FOR	APPLES	NOT	JOHN	HE	
"Mary is hungry for apples." →	1	1	1	0	1	1	0	0	0	→ [1, 1, 1, 0, 1, 1, 0, 0, 0]
"John is happy he is not hungry for apples." →	0	2	1	1	1	1	1	1	1	→ [0, 2, 1, 1, 1, 1, 1, 1, 1]

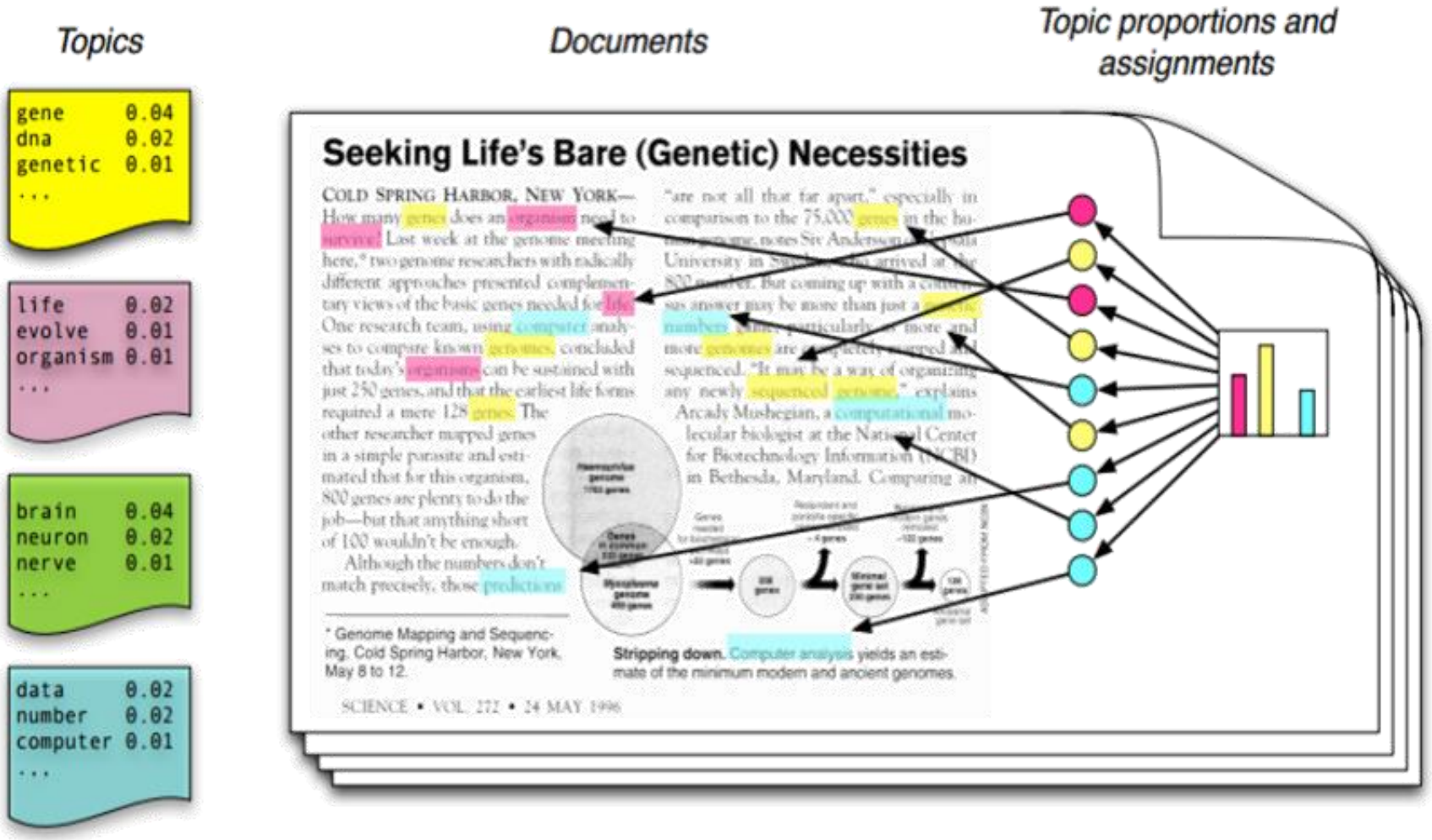
- No order/structure preserved
- Extend to N-grams (phrases e.g. "New York")
- Values could be:
  - binary (0,1): if a word appeared in a document
  - count - term frequency (TF): how many times a word appears
  - inverse document frequency (IDF): how specific a word is
  - TF-IDF



https://www.topbots.com/solve-ai-nlp-problems-guide/

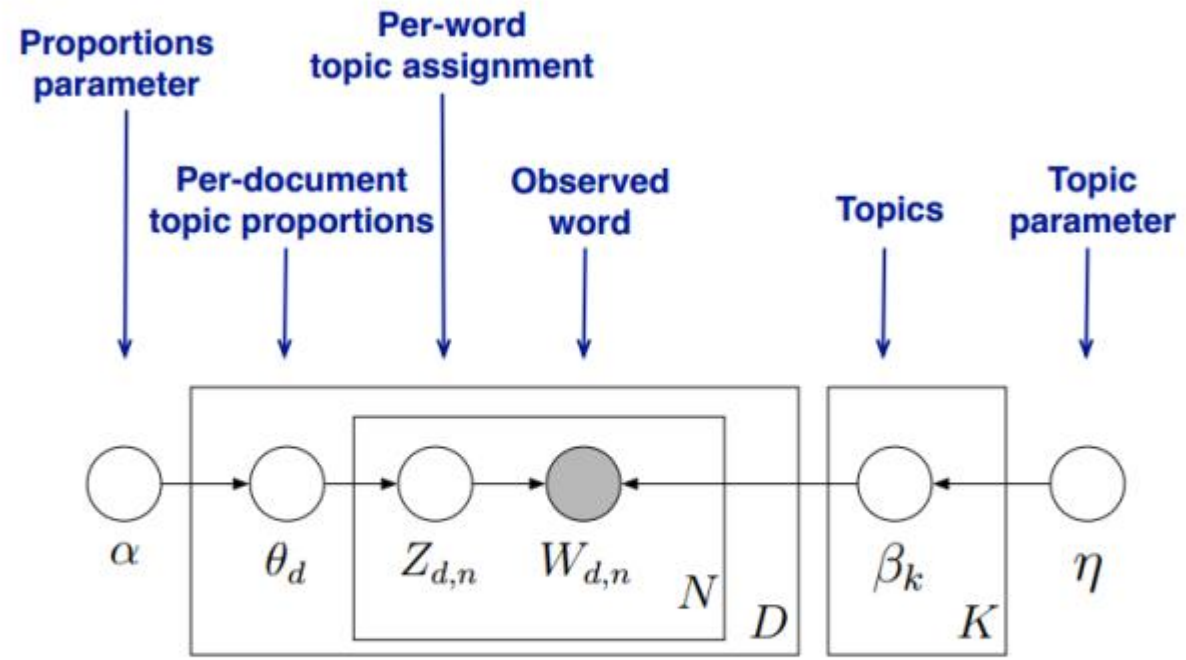
Topic modeling

- Hidden topics (classes), expressed in different language use
- Unsupervised, dimensionality reduction (clustering)



Topic modeling – Latent Dirichlet Allocation (LDA)

- Word generation process – assumption
- Bayesian framework, where (Dirichlet) priors enforce sparse representations



$$p(\beta, \theta, \mathbf{z}, \mathbf{w}) = \left(\prod_{i=1}^{K} p(\beta_i | \eta) \right) \left(\prod_{d=1}^{D} p(\theta_d | \alpha) \prod_{n=1}^{N} p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right)$$

Topic modeling – Latent Dirichlet Allocation (LDA)

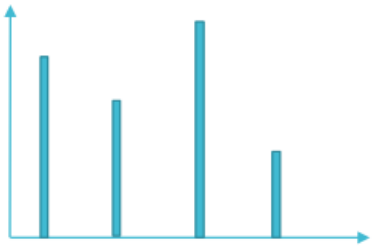
- Word generation process – assumption
- Bayesian framework, where (Dirichlet) priors enforce sparse representations

$$p(\beta,\theta,z,w)=\left(\prod_{i=1}^{K}p(\beta_i|\eta)\right)\left(\prod_{d=1}^{D}p(\theta_d|\alpha)\prod_{n=1}^{N}p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K},z_{d,n})\right)$$

$$(\beta_d|\eta)\sim Dir(\beta) \qquad (\theta_d|\alpha)\sim Dir(\alpha) \qquad Z_{d,n}\sim Multi(\theta_d) \qquad W_{d,n}\sim Multi(\beta_{z_{d,n}})$$

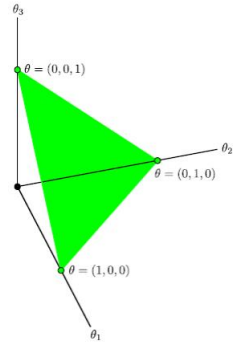
$$p(z_{d,n}|\theta_d)=\theta_{d,z_{d,n}} \qquad p(w_{d,n}|z_{d,n},\beta_{1:K})=\beta_{z_{d,n},w_{d,n}}$$

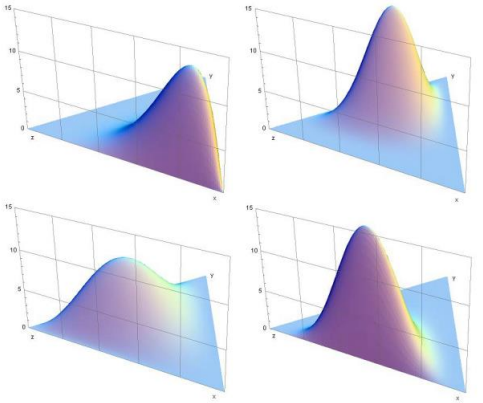
θ_d:



β:

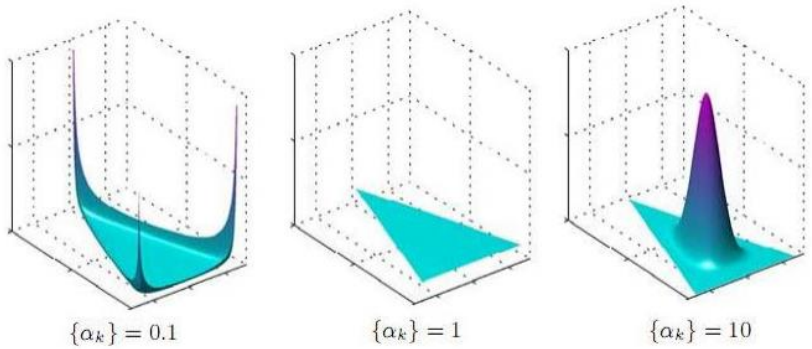
Topics	Word probabilities for each topic		



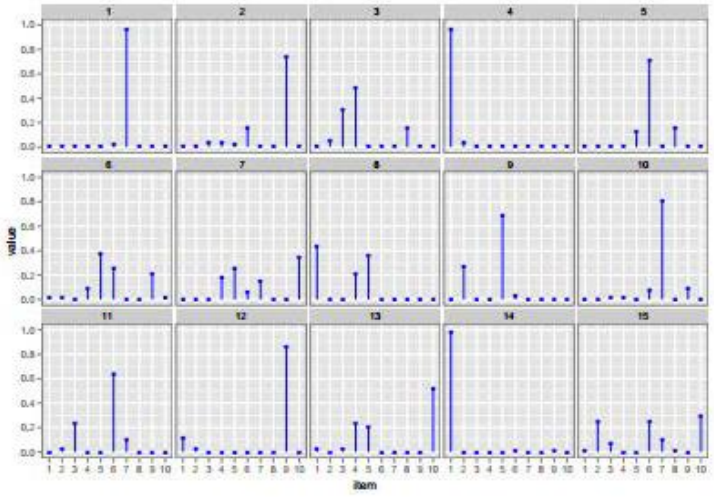


Topic modeling – Latent Dirichlet Allocation (LDA)

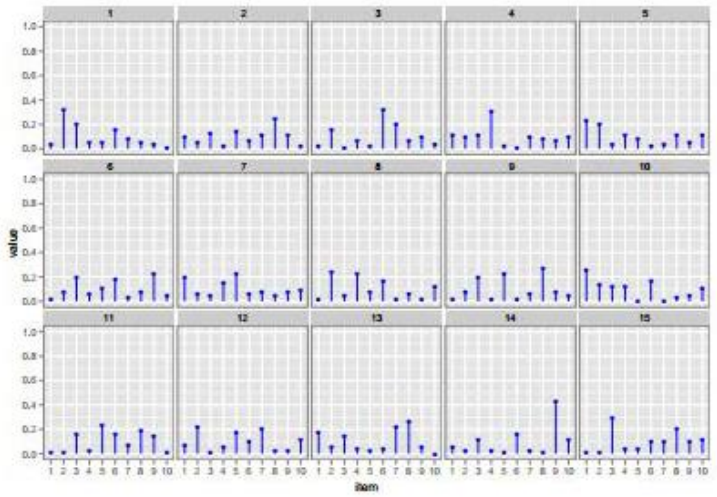
- Word generation process – assumption
- Bayesian framework, where (Dirichlet) priors enforce sparse representations



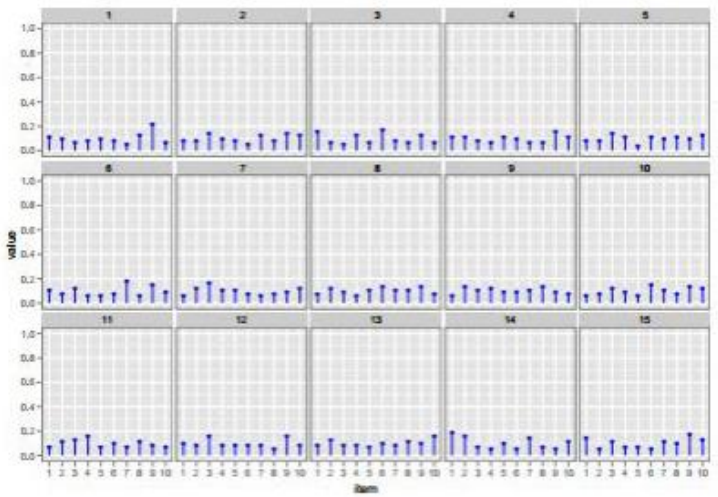
$\alpha_i = 0.1$



$\alpha_i = 1$

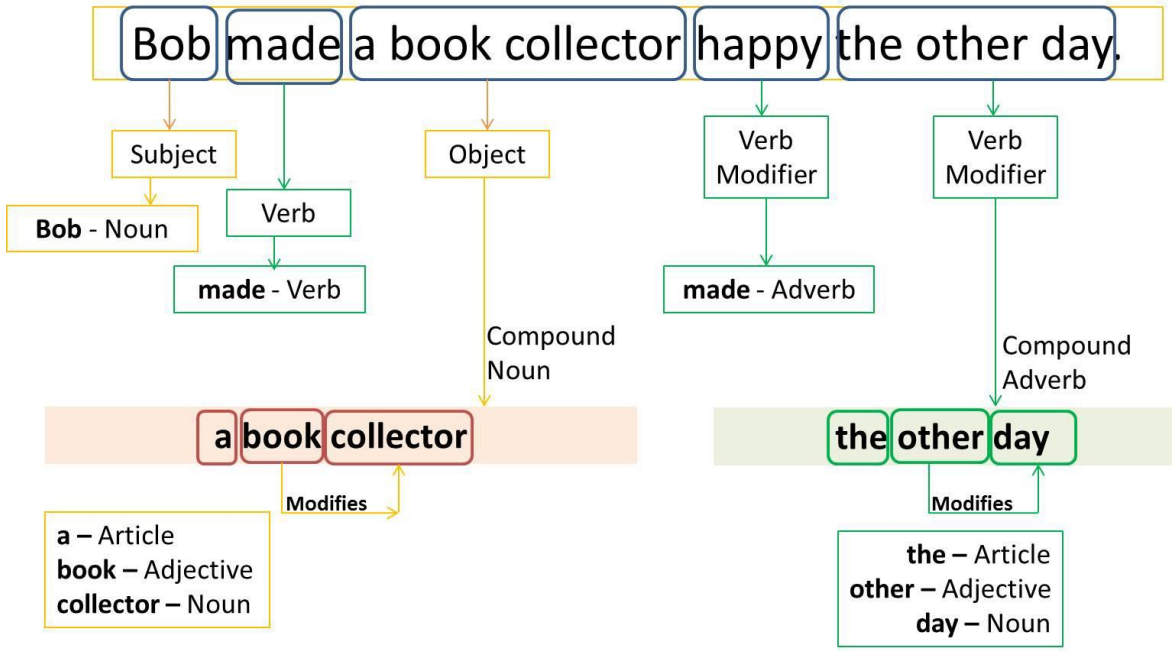


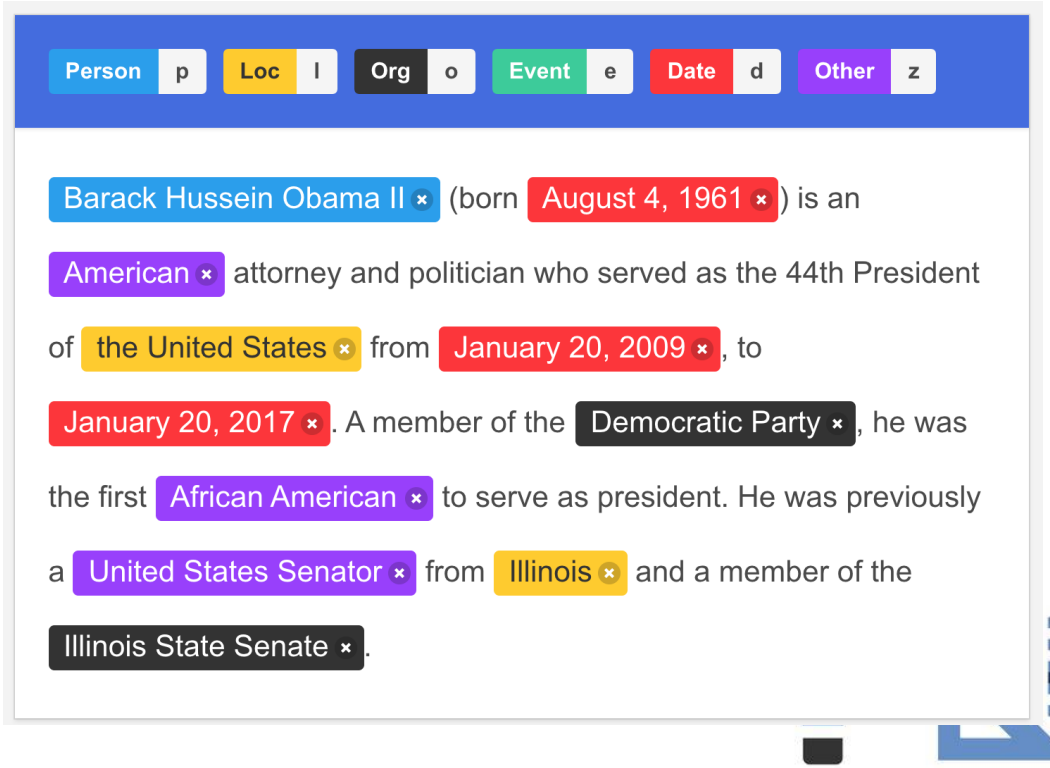
$\alpha_i = 10$



Word tagging

- Classification of words
- Structure/order of words very important
  - requires learning **structured models**: Hidden markov models, Conditional random fields, Recurent neural networks
- Different kinds of labels: Part-of-speech, Named Entity Recognition



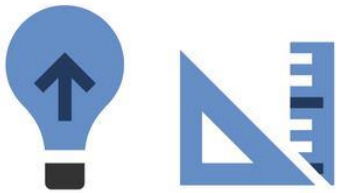


Word tagging

- Classification of words
- Structure/order of words very important
  - requires learning structured models: Hidden markov models, Conditional random fields, Recurent neural networks
- Different kinds of labels: Special domain labels

tag	$P(c\|d)$	Concept	$P(w\|c)$
a	0.1702	PHYSICS	electrons (0.2767) electron (0.1367) radiation (0.0899) protons (0.0723) ions (0.0532) radioactive (0.0476) proton (0.0282)
b	0.1325	CHEMICAL ELEMENTS	oxygen (0.3023) hydrogen (0.1871) carbon (0.0710) nitrogen (0.0670) sodium (0.0562) sulfur (0.0414) chlorine (0.0398)
c	0.0959	ATOMS, MOLECULES, AND SUB-ATOMIC PARTICLES	atoms (0.3009) molecules (0.2965) atom (0.2291) molecule (0.1085) ions (0.0262) isotopes (0.0135) ion (0.0105) isotope (0.0069)
d	0.0924	ELECTRICITY AND ELECTRONICS	electricity (0.2464) electric (0.2291) electrical (0.1082) current (0.0882) flow (0.0448) magnetism (0.0329)
o	0.5091	OTHER	

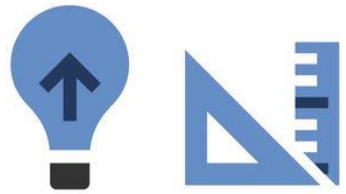
The hydrogen[b] ions[a] immediately° attach° themselves to water° molecules[c] to form° combinations° called° hydronium ions[a]. The chlorine[b] ions[a] also associate° with water° molecules[c] and become hydrated. Ordinarily°, the positive° hydronium ions[a] and the negative° chlorine[b] ions[a] wander° about freely° in the solution° in all directions°. However, when the electrolytic cell° is connected° to a battery°, the anode[c] becomes positively° charged[a] and the cathode[d] becomes negatively° charged[a]. The positively° charged[a] hydronium ions[a] are then attracted° toward the cathode[d] and the negatively° charged[a] chlorine[b] ions[a] are attracted° toward the anode[d]. The flow[d] of current[d] inside° the cell° therefore consists of positive° hydronium ions[a] flowing[d] in one direction° and negative° chlorine[b] ions[a] flowing[d] in the opposite° direction°. When the hydronium ions[a] reach° the cathode[d], which has an excess° of electrons[a], each takes° one electron[a] from it and thus neutralizes° the positively° charged[a] hydrogen[b] ion[a] attached° to it. The hydrogen[b] ions[a] thus become hydrogen[b] atoms[c] and are released° into the solution°. Here they pair° up to form° hydrogen[b] molecules[c] which gradually° come out of the solution° as bubbles° of hydrogen[b] gas°. When the chlorine[b] ions[a] reach° the anode[d], which has a shortage° of electrons[a], they give° up their extra° electrons[a] and become neutral[a] chlorine[b] atoms[c]. These pair° up to form° chlorine[b] molecules[c] which gradually° come out of the solution° as bubbles° of chlorine[b] gas°. The behavior° of hydrochloric acid° solution° is typical° of all electrolytes°. In general°, when acids°, bases°, and salts° are dissolved° in water°, many of their molecules[c] break° up into positively° and negatively° charged[a] ions[a] which are free° to move° in the solution°.



Named Entity Recognition - Data

Foreign	ORG
Ministry	ORG
spokesman	O
Shen	PER
Guofang	PER
told	O
Reuters	ORG
:	:

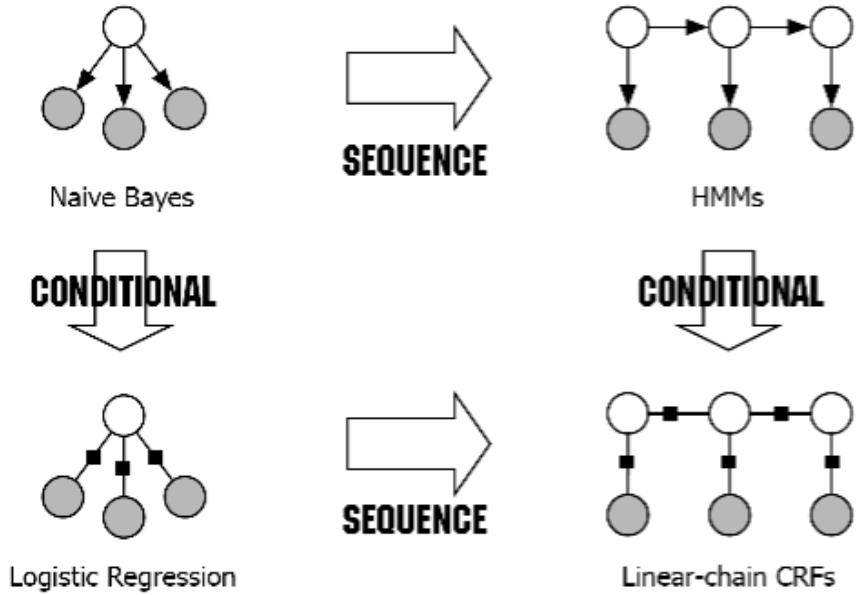
Document	IO encoding	IOB encoding
Fred	PER	B-PER
showed	O	O
Sue	PER	B-PER
Van	PER	B-PER
Gogh	PER	I-PER
‘s	O	O
new	O	O
painting	O	O



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Structured learning using CRFs

- Conditional Random Fields – generalization of Logistic regression
- Allows dependence between output labels!



$$P(\overline{y}|\overline{x};w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \overline{x}, i))}$$

Length of sequence x

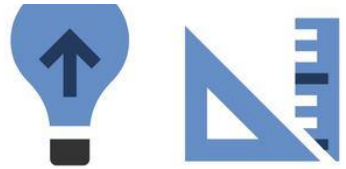
Sum over all feature function

Weight for given feature function

Feature Functions

Sum over all possible label sequence

Feature function can access all of observation

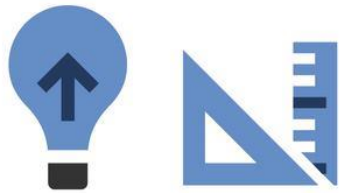


Discriminative model features

- **Features f** are elementary pieces of evidence that link aspects of what we observe *d* with a category *c* that we want to predict
- A feature is a function with a bounded real value: $f: C \times D \rightarrow \mathbb{R}$
- Example: $f(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$

LOCATION	PERSON	LOCATION
in Petnica	*saw Stefan*	*at home*

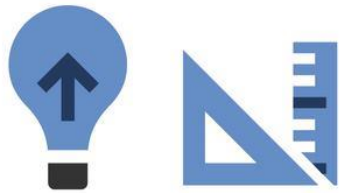
- Models will assign to each feature a *weight*:
  - A positive weight votes that this configuration is likely correct
  - A negative weight votes that this configuration is likely incorrect



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

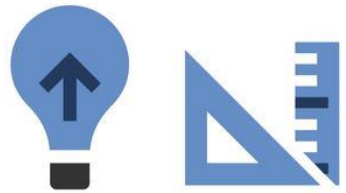
Discriminative model features

- Features for sequence labeling:
  - Words
    - Current word (essentially like a learned dictionary)
    - Previous/next word (context) – N-grams
    - Word substrings
    - Word shapes (*Club-99* represented as *Xxxx-dd*)
      - Map words to simplified representation that encodes attributes such as length, capitalization, numerals, Greek letters, internal punctuation, etc.
  - Other kinds of inferred linguistic classification
    - Part-of-speech tags
  - Label context
    - Previous (and perhaps next) label



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Language modeling



Probabilistic language modeling

- Goal - compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5|w_1,w_2,w_3,w_4)$$

- A model that computes either of these: $P(W)$ or $P(w_n|w_1,w_2...w_{n-1})$ is called a language model
- Reminder: The Chain rule

$$P(x_1,x_2,x_3,...,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)...P(x_n|x_1,...,x_{n-1})$$

*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Probabilistic language modeling

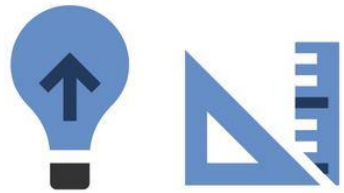
- How can we estimate joint probability of words in sentence?

$$P(\text{"we all love machine learning"}) = P(we) \cdot P(all|we) \cdot P(love|we, all) \cdot P(machine|we, all, love) \cdot P(learning|we, all, love, machine)$$

- From the corpus, using counts (C):

$$P(machine|we, all, love) = \frac{C(we, all, love, machine)}{C(we, all, love)}$$

- Far too many possible sentences for us to estimate
- We'll never see enough data for estimating these



*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Probabilistic language modeling

Markov Assumption

- Simplifying assumption:

$$P(\text{learning}|\text{machine,love,all,we}) \approx P(learning|machine)$$

- Or maybe 2 last words:

$$P(\text{learning}|\text{machine,love,all,we}) \approx P(learning|machine, love)$$

- General **N-grams**:

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Probabilistic language modeling

Smoothing: Add-one(Laplace) smoothing

- Redistribute probability to unseen words

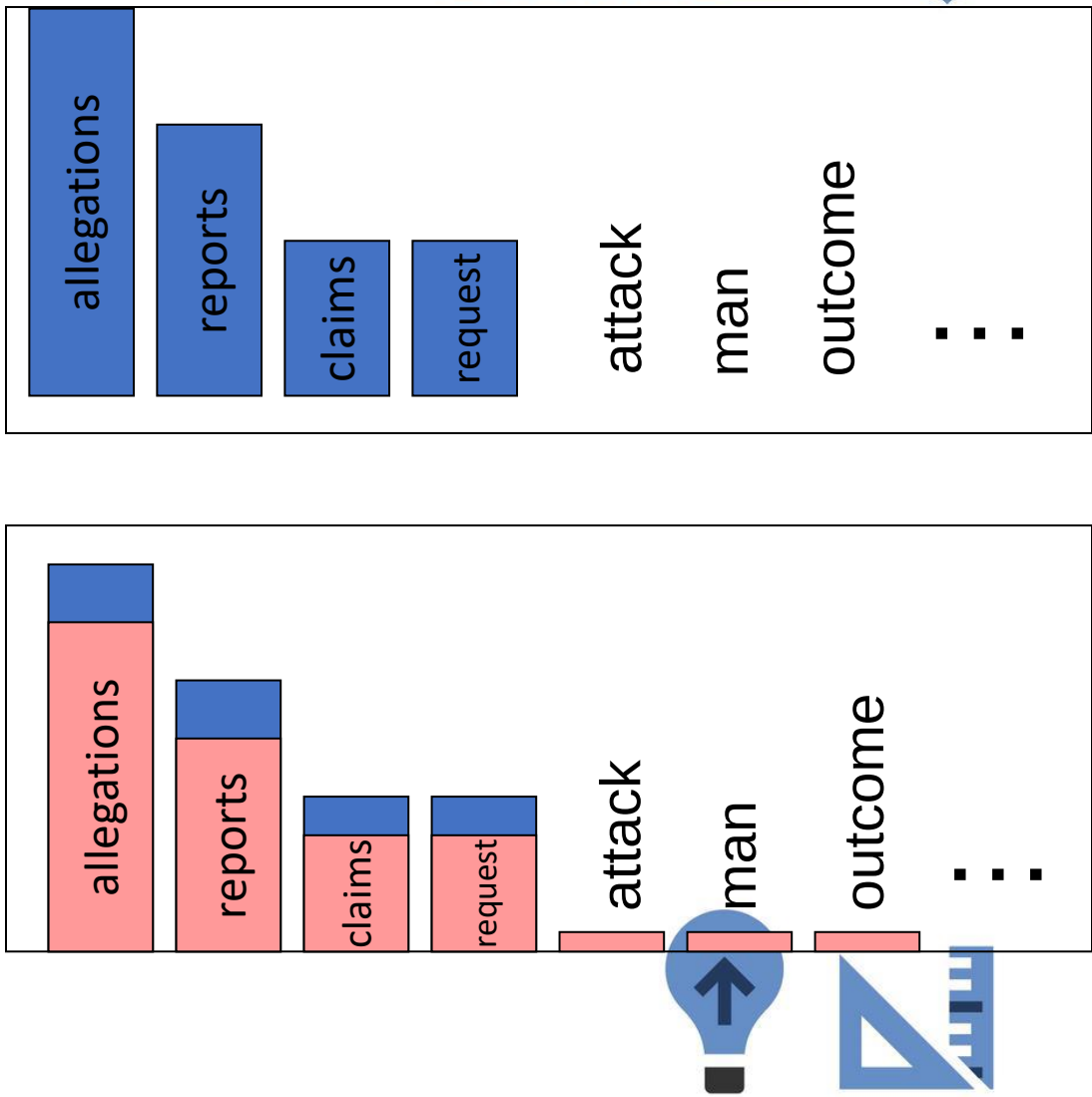
- MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-one estimate

$$P_{Add\text{-}1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

- There are better smoothing alghorithms: backoff, interpolation, Kneser-Ney



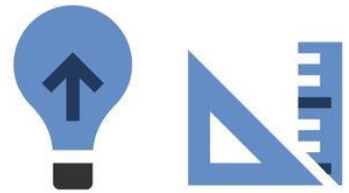
*Slide from: Danijela Stepanovic, NLP, PSIML 2018

Probabilistic language modeling

- Google web 1T 5-gram (2006, "All Our N-gram are Belong to You")
- What is the purpose of LM?
  - Text generation? (Generative model) https://www.youtube.com/watch?v=no_elVGGgW8
  - Text classification, but with structured models (not-naive Bayes):

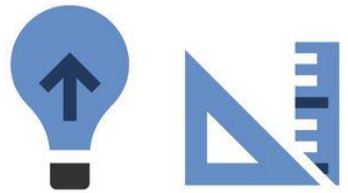
$$\underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}, \qquad P(d|c) \text{ - laguage model!}$$

- Recognition of: Language, Authorship, Age, ...
  - classes that use the language differently, both words and structure
- Prior for more advanced models: Optical Character Recognition, Speech recognition, Captioning, ...



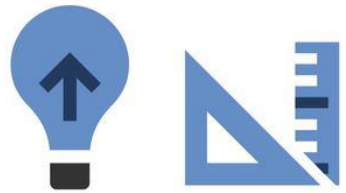
Word embeddings

Capturing semantic relationships from structure



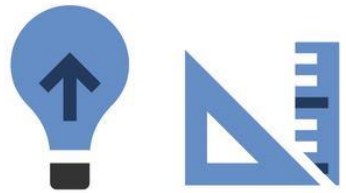
N-gram LM - Problems

- Out of vocabulary words
- Size of the models
- Requires large corpora
- Not accounting for language noise: same meaning can be expressed using different words
  - Sentence 1: Obama speaks to the media in Illinois
  - Sentence 2: The president greets the press in Chicago



Word classes

- Very successful in the practice of NLP
- Group similar words, to increase generalization
  - $Class1$ = ($yellow$, $green$, $blue$, red)
  - $Class2$ = ($Italy$, $Germany$, $France$, $Spain$)
- Mapping words to a single class
- Instead of counting words in documents, count classes
- “Class-based n-gram models of natural language” (Brown, 1992)



One-hot representation

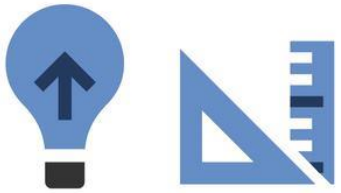
- Represent each word as a code:

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

- Bag-of-words: sum of all word codes from a document
- Could be extended to bag-of-N-grams

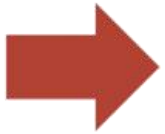


https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Word classes

- Manually:

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

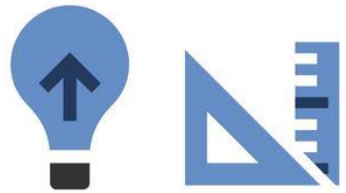


	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

Each word gets a 1x3 vector

Similar words… similar vectors

- Dimensionality reduction (nonlinear)



https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Learning Distributed semantics

- A bottle of tesgüino is on the table.
- Everybody likes tesgüino.
- Tesgüino makes you drunk.
- We make tesgüino out of corn.

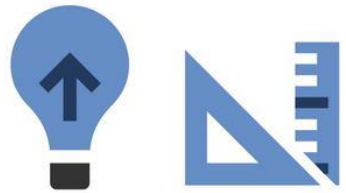
The meaning of a word is related to the distribution of the words around it.

from Speech and Language Processing by Dan Jurafsky and James H. Martin.

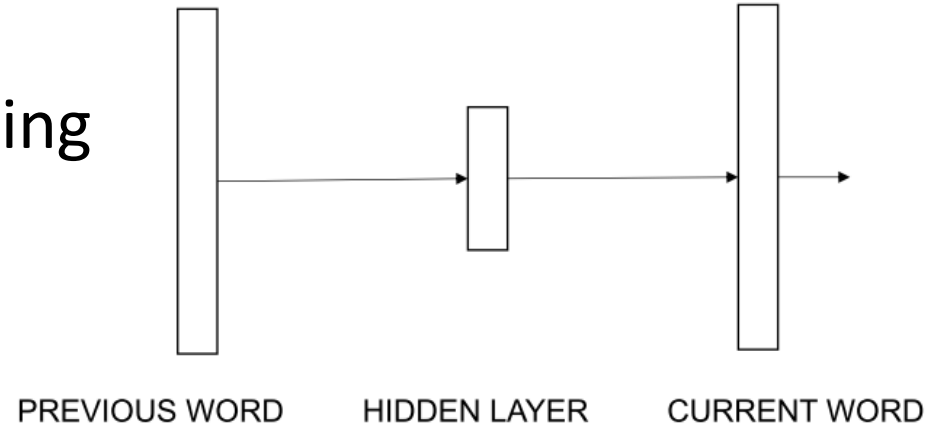
You shall know a word by the company it keeps

(Firth, J. R. 1957)

*Tesgüino is a corn beer made by the Tarahumara Indians of Sierra Madre in Mexico.



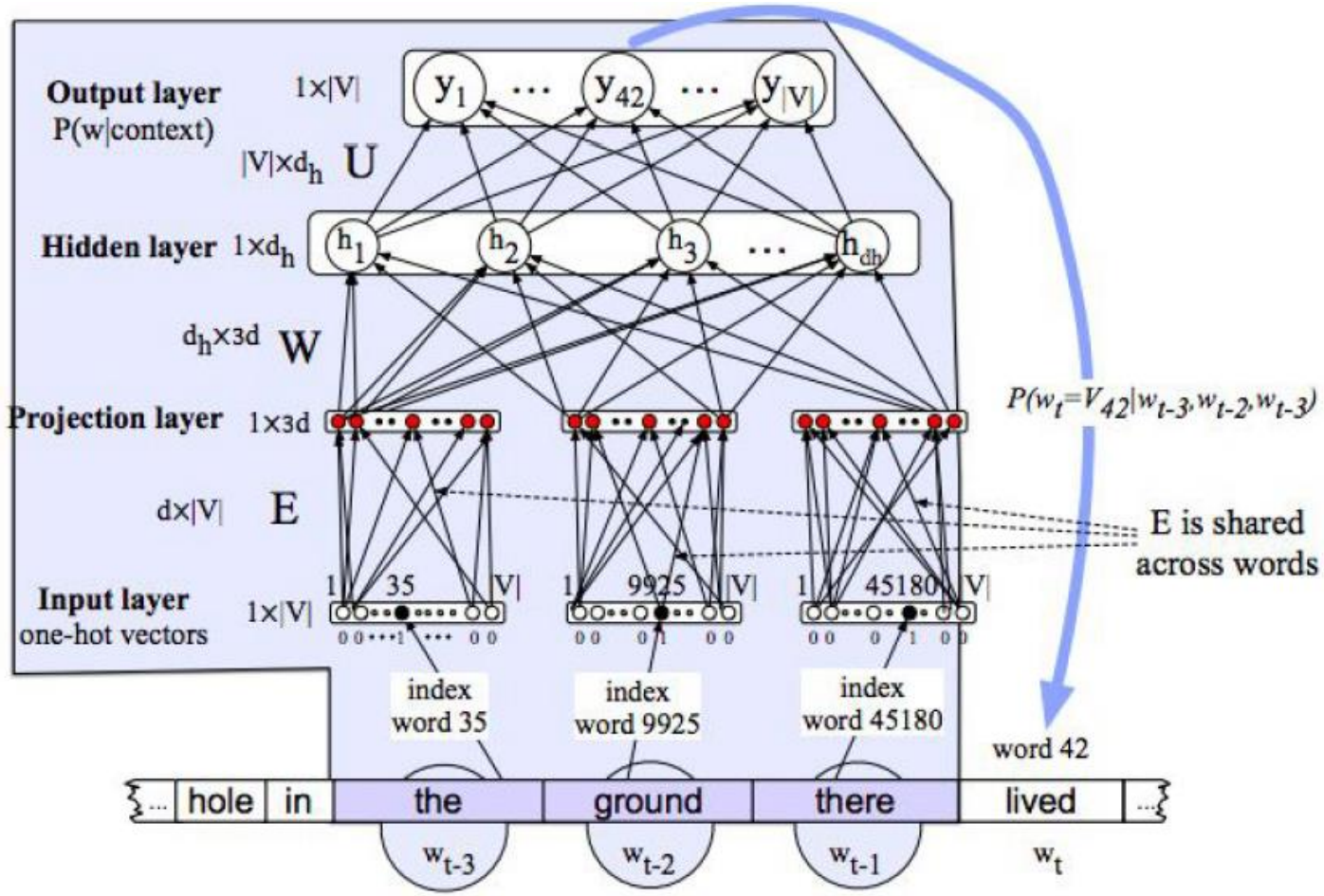
Distributed representations



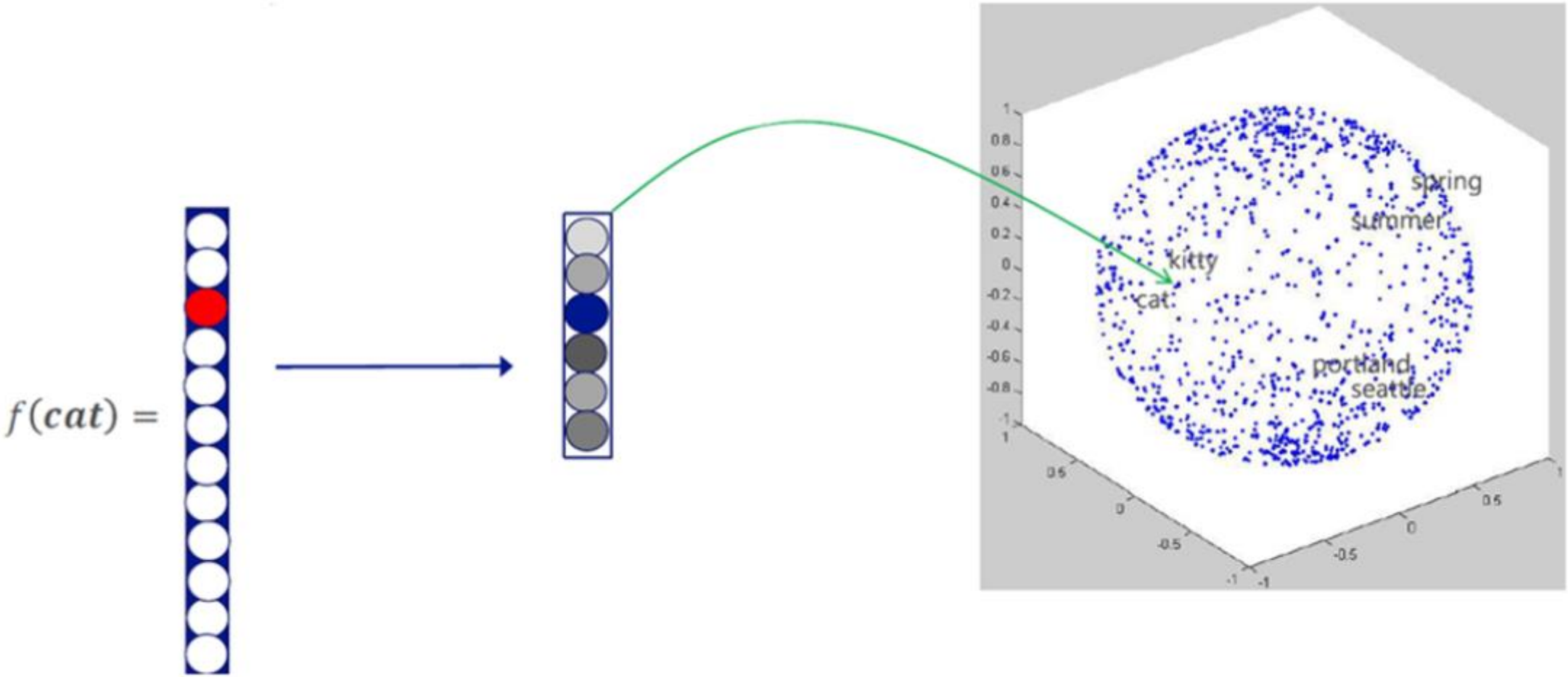
- Neural network applied to Language modeling
  - input is one-hot codes
  - hidden-layer: compressed representation
    - word embedding
    - dense (N: 50-1000)
    - similar properties to word classes, but multiple similarities (Belgrade is similar to Warsaw, but also to Serbia)
- Embeddings could be used for many NLP tasks
  - Pretrained (transfer learning)
  - “Natural Language Processing (Almost) from Scratch”, Collobert et al 2011
- Many architectures, some deep

NN Language models

- “A neural Probabilistic Language Model”, Bengio et al. 2003.



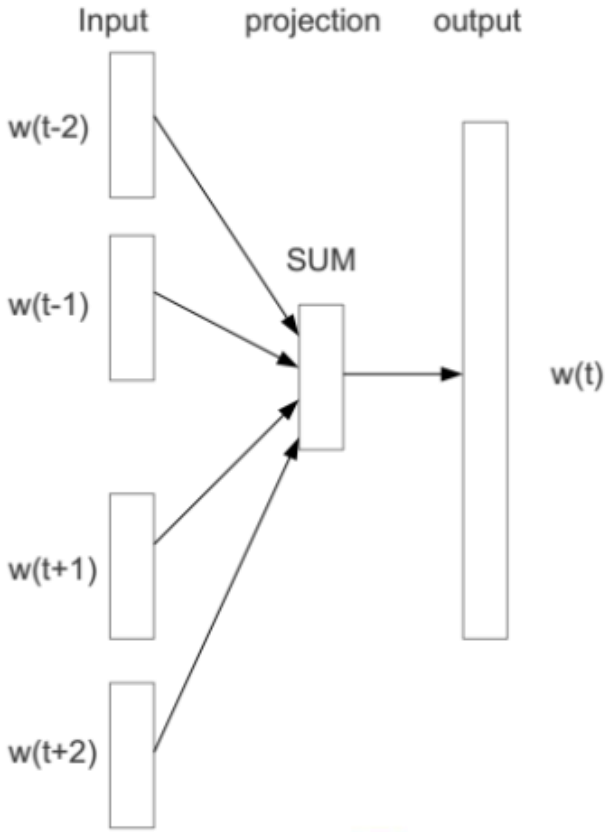
Word embeddings



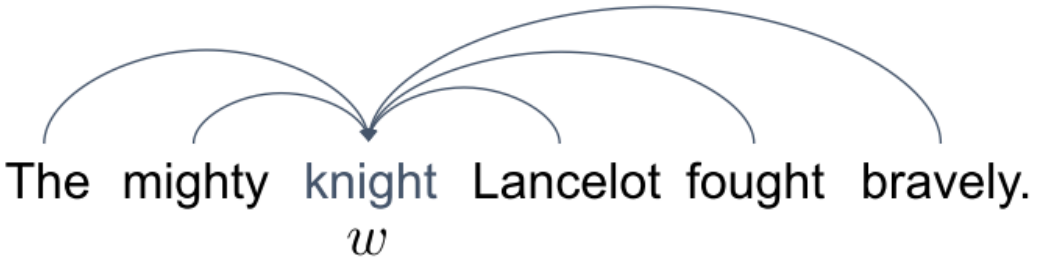
word2vec (Mikolov, 2013)

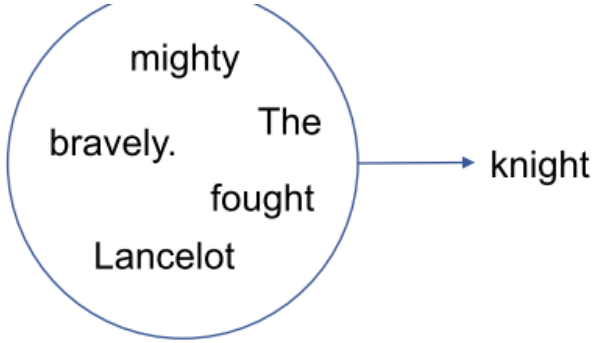
- Shallow network
- The hidden layer is linear (no activation)
- The weights for different positions are shared
- Much more efficient than normal NNLM
  - but cannot model n-grams

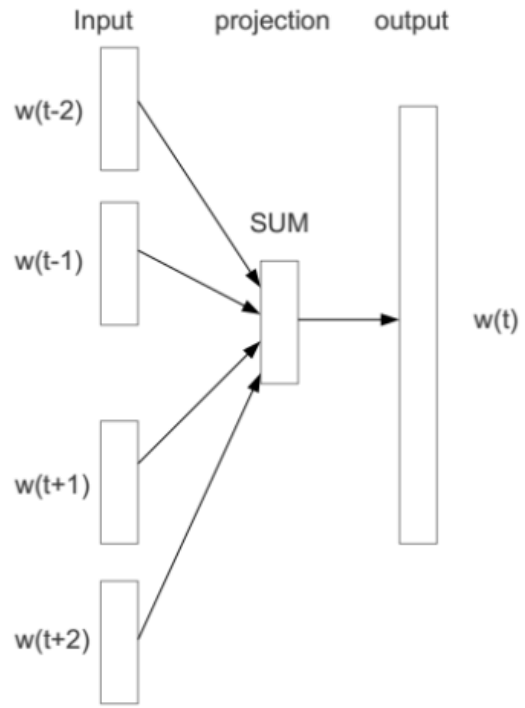
- https://github.com/tmikolov/word2vec



CBOW model







- Model probability of a word given a context

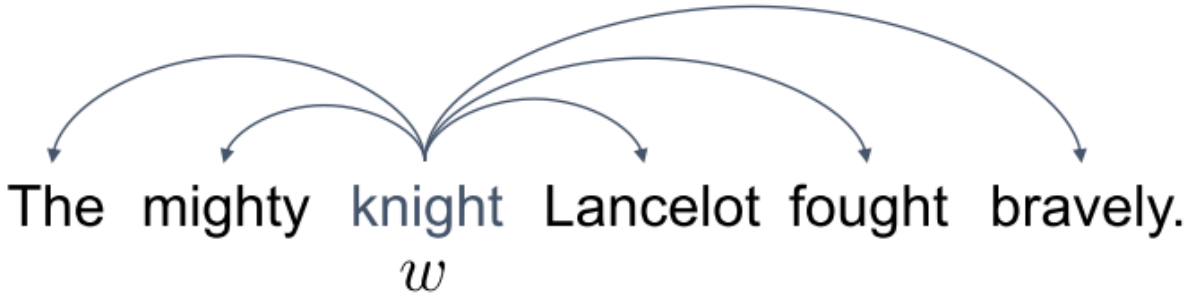
$$\text{feature for context } \mathcal{C}\text{: } h_{\mathcal{C}}$$

$$\text{classifier for word } w\text{: } v_w$$

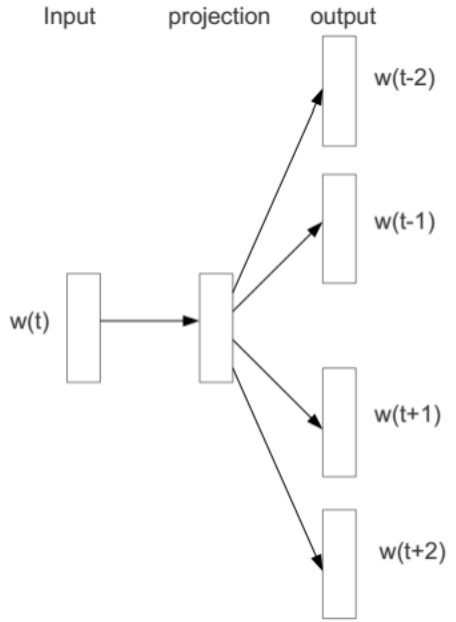
$$p(w|\mathcal{C}) = \frac{e^{h_{\mathcal{C}}^\top v_w}}{\sum_{k=1}^{K} e^{h_{\mathcal{C}}^\top v_k}}$$

- Continuous Bag Of Words $h_{\mathcal{C}} = \sum_{c \in \mathcal{C}} x_c$

Skip-gram model



knight → The
knight → mighty
knight → Lancelot
knight → fought
knight → bravely.



- Model probability of a context given a word

feature for word w: x_w

classifier for word c: v_c

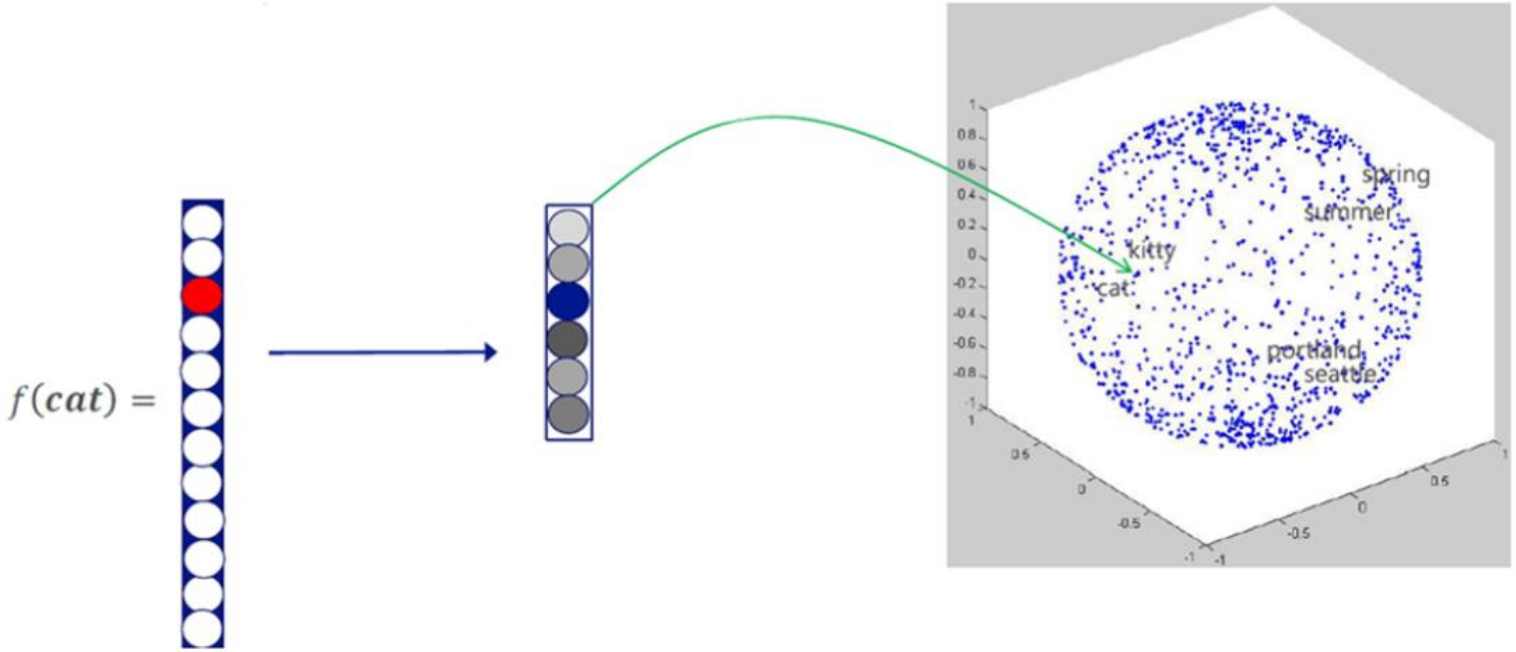
$$p(c|w) = \frac{e^{x_w^\top v_c}}{\sum_{k=1}^{K} e^{x_w^\top v_k}}$$

- Word vectors $x_w \in \mathbb{R}^d$

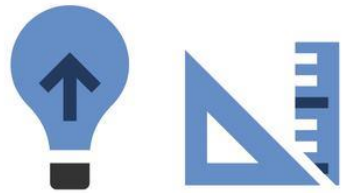
Words in embedding space

- We can do nearest neighbor search around result of vector operation
- Similarity using cosine :
  - empirically best
- Learned word classes!

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

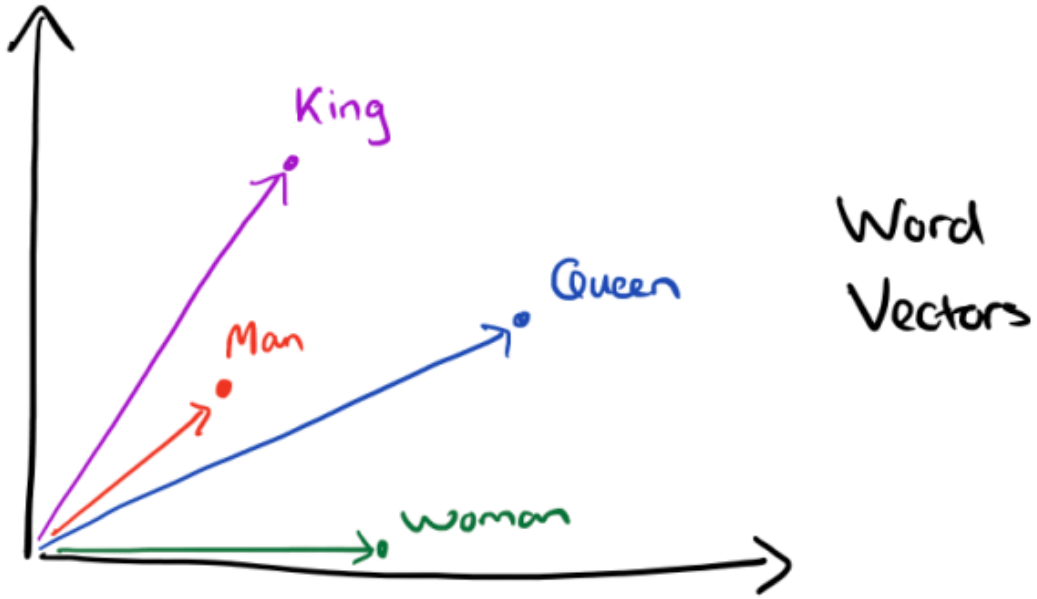


Pretrained w2v demo



Analogies by vector arithmetic

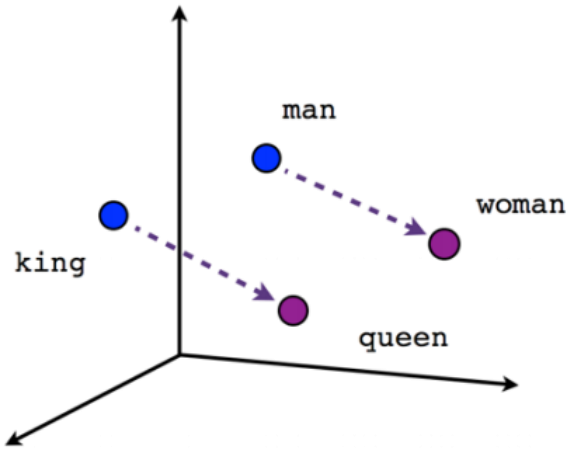
- It was shown that word vectors capture many linguistic properties (gender, tense, plurality, even semantic concepts like “capital city of”)
- “King” – “Man” + “Woman” = ?



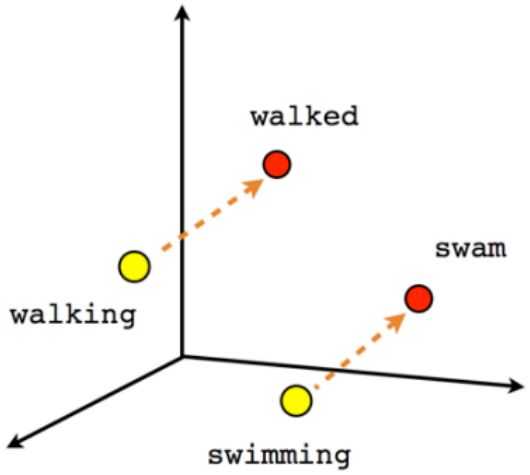
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

Analogies by vector arithmetic

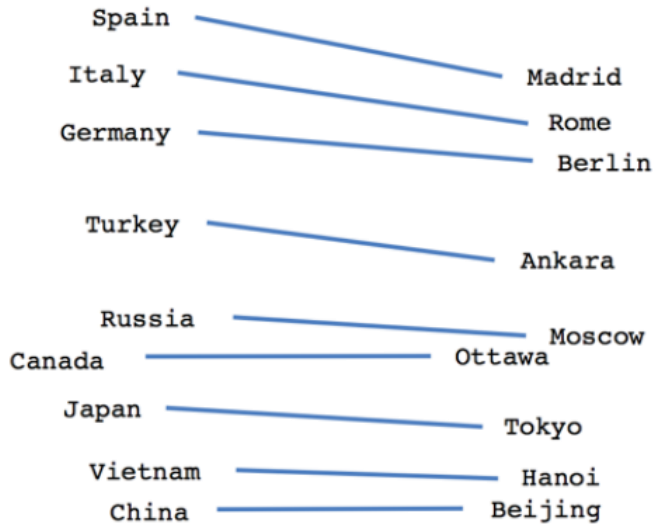
- It was shown that word vectors capture many linguistic properties (gender, tense, plurality, even semantic concepts like “capital city of”)



Male-Female



Verb tense

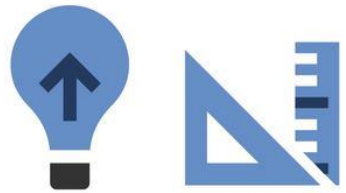


Country-Capital

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

Relations between words

- synonyms: big and large
- concept categories: dog, cat → animals
- associations: bee & honey
- analogies: big and bigger as small and smaller



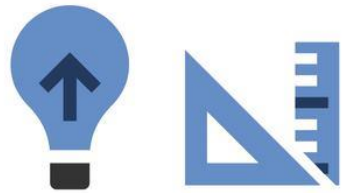
Analogies by vector arithmetic

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Training

$$p(c|w) = \frac{e^{x_w^\top v_c}}{\sum_{k=1}^{K} e^{x_w^\top v_k}}$$

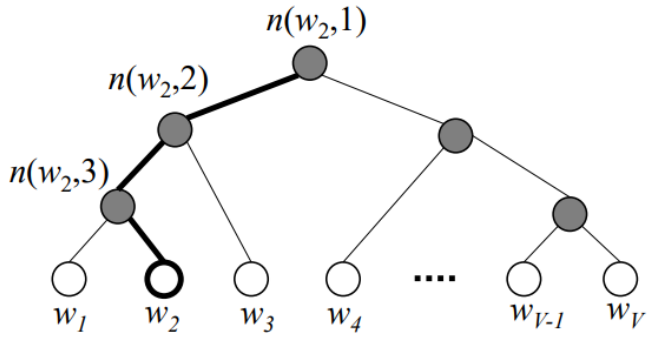
- Typical for NN: SGD + backpropagation
- Very large softmax in the output layer (up to millions of outputs) (computationally expensive):
  - Negative sampling (independent logistic regressions, training does not depend on vocabulary size)
  - Hierarchical softmax (vocabulary with Huffman coding)



Negative sampling

- Independent logistic regressions output, instead of multinomial
- Each training sample modify only a small percentage of weights
- In output layer we are updating weights only for positive words and few negative samples
  - “we love **machine** learning” => (context, target) pairs
    - [we, love, learning] , machine => 1
    - [we, love, learning] , dream => 0
    - [we, love, learning] , theater => 0
    - [we, love, learning] , the => 0
- Selecting negative samples
  - Unigram distribution where more frequent words are more likely to be selected
  - Subsampling frequent words – for each word there is a chance that we will delete it – Probability that we cut word is related to word frequency

Hierarchical Softmax



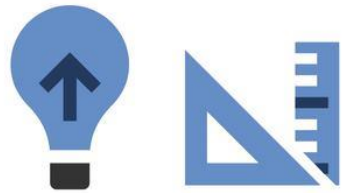
- Model uses a binary tree to represent all words in the vocabulary
- No output vector representation for words
- Each of the V-1 inner units has an output vector $v'_{n(w,j)}$
- Probability of a word being the output word

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w, j))]\!] \cdot \mathbf{v'}_{n(w,j)}{}^{T}\mathbf{h}\right)$$

- ch(n) – left child of unit n
- Logistic function $\sigma(u) = \frac{1}{1+e^{-u}}$
- $[\![x]\!] = \begin{cases} 1 \; if \; x \; is \; true \\ -1 \; otherwise \end{cases}$

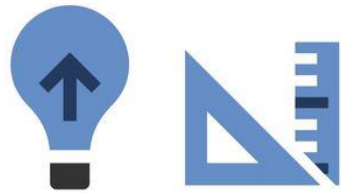
$$p(n, \text{left}) = \sigma\left(\mathbf{v'_n}^{T} \cdot \mathbf{h}\right) \qquad p(n, \text{right}) = 1 - \sigma\left(\mathbf{v'_n}^{T} \cdot \mathbf{h}\right) = \sigma\left(-\mathbf{v'_n}^{T} \cdot \mathbf{h}\right)$$

Learning embeddings demo



Evaluation

- Predicting Analogies known in advance:
  - Microsoft Research dataset with 8K “analogies” - examples:
    - good:better => rough: ___
    - good:best => rough: ___
    - better:best => rougher: ___
    - year:years => law: ___
    - see:saw => return: ___
- Evaluating a Downstream task that uses embeddings as input
  - Text classification
  - Named entity recognition
  - …



Comparison of performance

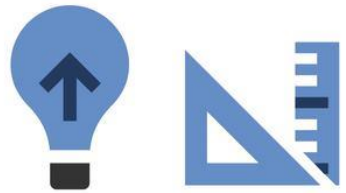
- Google 20K questions dataset (word based, both syntax and semantics)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Comparison of performance

- Going from weeks of training to minutes while improving accuracy!

Model	*Vector Dimensionality*	*Training Words*	*Training Time*	*Accuracy [%]*
Collobert NNLM	50	660M	2 months	11
Turian NNLM	200	37M	few weeks	2
Mnih NNLM	100	37M	7 days	9
Mikolov RNNLM	640	320M	weeks	25
Huang NNLM	50	990M	weeks	13
Skip-gram (hier.s.)	1000	6B	hours	66
CBOW (negative)	300	1.5B	**minutes**	**72**

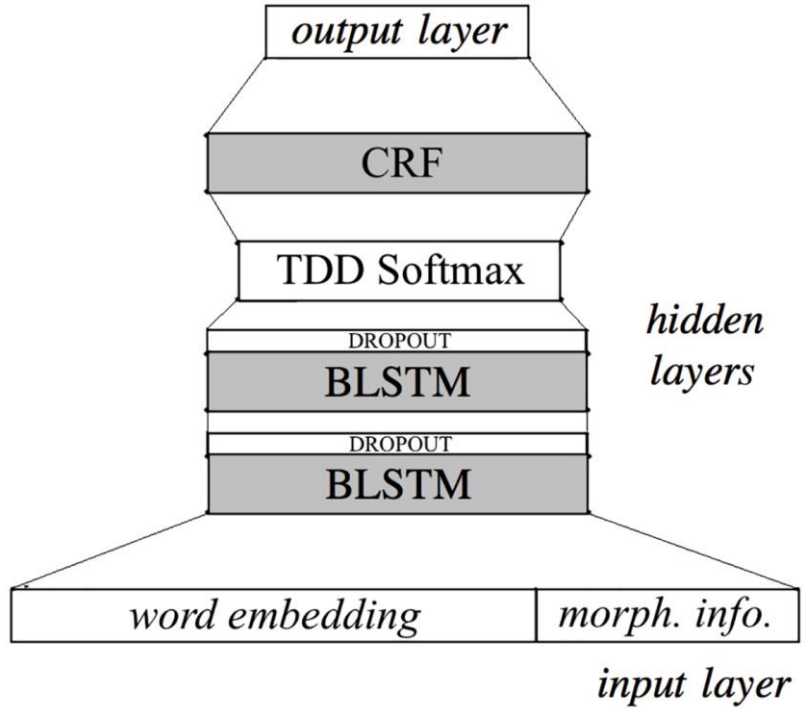


Comparison of performance

	Redmond	Havel	graffiti	capitulate
Collobert NNLM	conyers lubbock keene	plauen dzerzhinsky osterreich	cheesecake gossip dioramas	abdicate accede rearm
Turian NNLM	McCarthy Alston Cousins	Jewell Arzu Ovitz	gunfire emotion impunity	- - -
Mnih NNLM	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-gram (phrases)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	spray paint grafitti taggers	capitulation capitulated capitulating

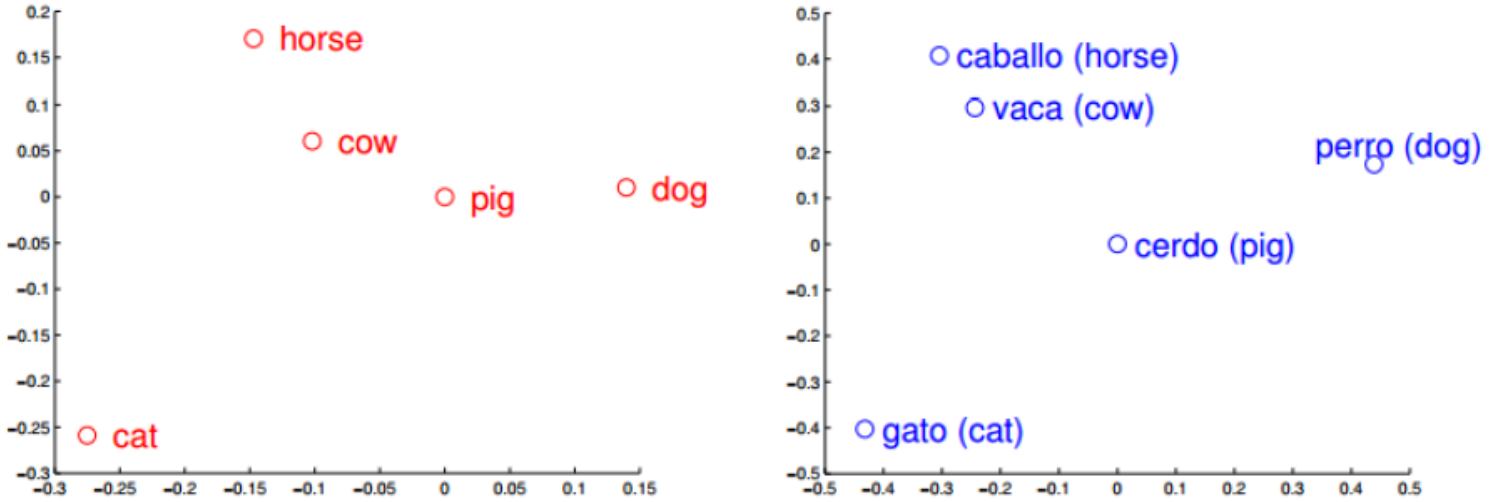
Downstream tasks

- Sentiment analysis
- Named entity recognition
- Translation
- In many other NLP tasks as preprocessing task
- Music/Image/Video recommendation systems



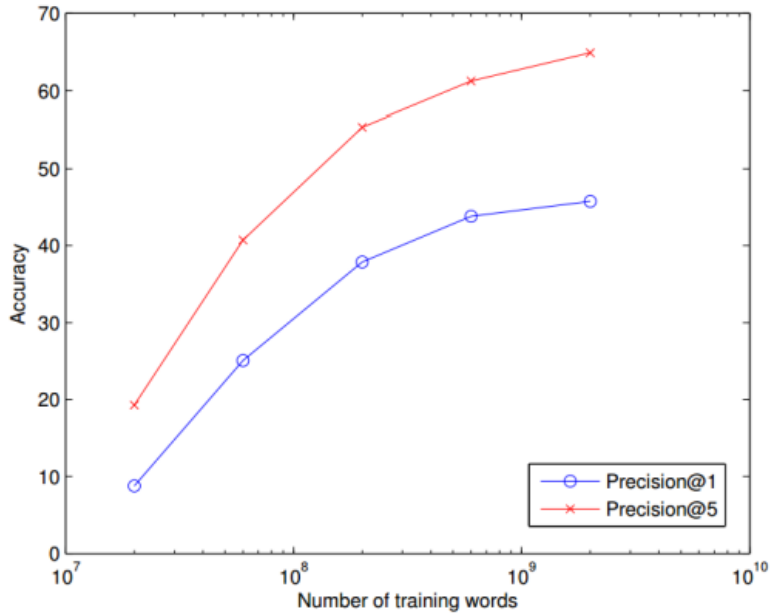
Downstream tasks

- Translation of words using vector spaces (same concept appear in same contexts, even in different languages)
  - need small dictionary to aligh two language embeddings



Downstream tasks

- Translation of words using vector spaces (same concept appear in same contexts, even in different languages)
  - need small dictionary to aligh two language embeddings



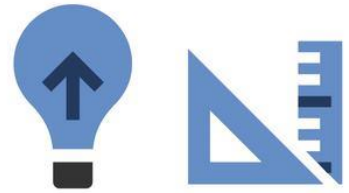
Spanish word	Computed English Translations	Dictionary Entry
emociones	emotions emotion feelings	emotions
protegida	wetland undevelopable protected	protected
imperio	dictatorship imperialism tyranny	empire
destacaron	highlighted emphasized emphasised	highlighted

Using word embeddings – transfer learning

1. Learn word embeddings from large text corpus (~100B words)
   (Or download pre-trained embedding online)
2. Transfer embedding to new task with smaller training set
3. Optional: Continue to finetune the word embeddings with new data
   (Only in case if your data set is big enough)

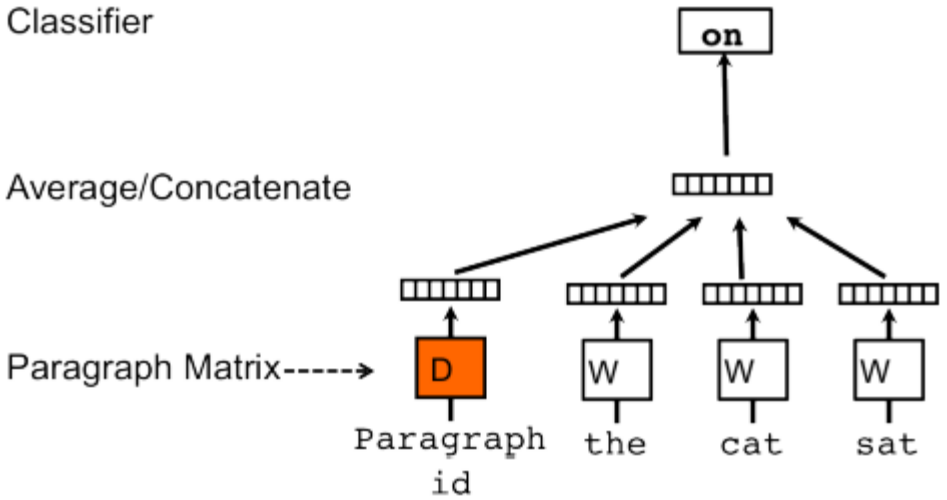
Solved both of the NLP challenges:

- noise in the exact words used (words => concepts)
- uses the structure of the sentence to capture semantics



Improvements

- Text classification – beyond averaging word vectors
- Sentence-level or document-level representations
  - Need to be trained for many epochs
  - **Doc2Vec**, Distributed Representations of Sentences and Documents (Le et al, 2014)



Improvements

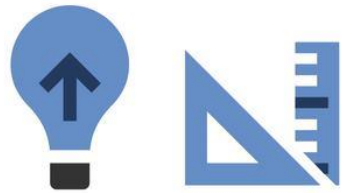
- Extract phrases (Novi Sad, Air Serbia) as new tokens (Air_Serbia)
- Position-dependent weighting of context
- Remove duplicate sentences

Model	Sem	Syn	Tot
cbow + uniq	79	73	76
cbow + uniq + phrases	82	78	80
cbow + uniq + phrases + weighting	87	82	85

Advances in Pre-Training Distributed Word Representations

Similar algorithms

- Glove (similar to word2vec, but based on matrix factorization)
  - GloVe: Global Vectors for Word Representation, Pennington et al 2014
- Par2Vec, Doc2Vec
- Node2Vec – embedding graphs instead of streams (context?)
- …

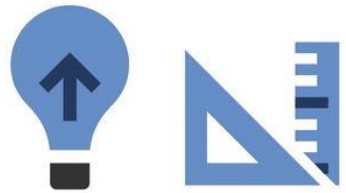


Examples of applications

- Job skills embedding from job postings
- Disease embeddings from pacient medical records
- User action embeddings from user click streams

FastText (Mikolov et al. 2016-2018)

using subword information



Subword information

- Problems with morphology
  - Words with same radicals don’t share parameters
  - disastrous / disaster
  - manage / managing
  - Especially for Morphologically rich languages
- Compound nouns and agglutinative languages
  - Tisch, Tennis, Tischtennis
- “Enriching Word Vectors with Subword Information”, Bojanowski, Grave, Joulin, Mikolov. TACL 2017
- FastText
- Represent words using its character n-grams

Polish declension

	Singular	Plural
Nominative	uniwersytet	uniwersytety
Genetive	uniwersytetu	uniwersytetów
Dative	uniwersytetowi	uniwersytetom
Accusative	uniwersytet	uniwersytety
Instrumental	uniwersytetem	uniwersytetami
Locative	uniwersytecie	uniwersytetach
Vocative	uniwersytecie	uniwersytety

FastText

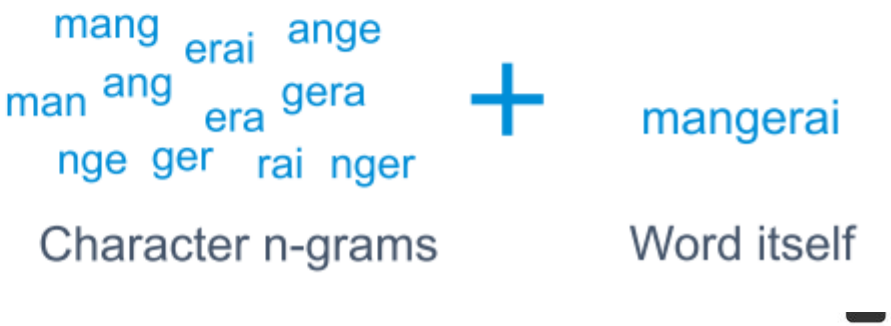
- As in skip-gram: model probability of a context word given a word

$$\text{feature for word } w\text{: } h_w$$
$$\text{classifier for word } c\text{: } v_c$$

$$p(c|w) = \frac{e^{h_w^\top v_c}}{\sum_{k=1}^{K} e^{h_w^\top v_k}}$$

- Feature of a word computed using n-grams:

$$h_w = \sum_{g \in w} x_g$$



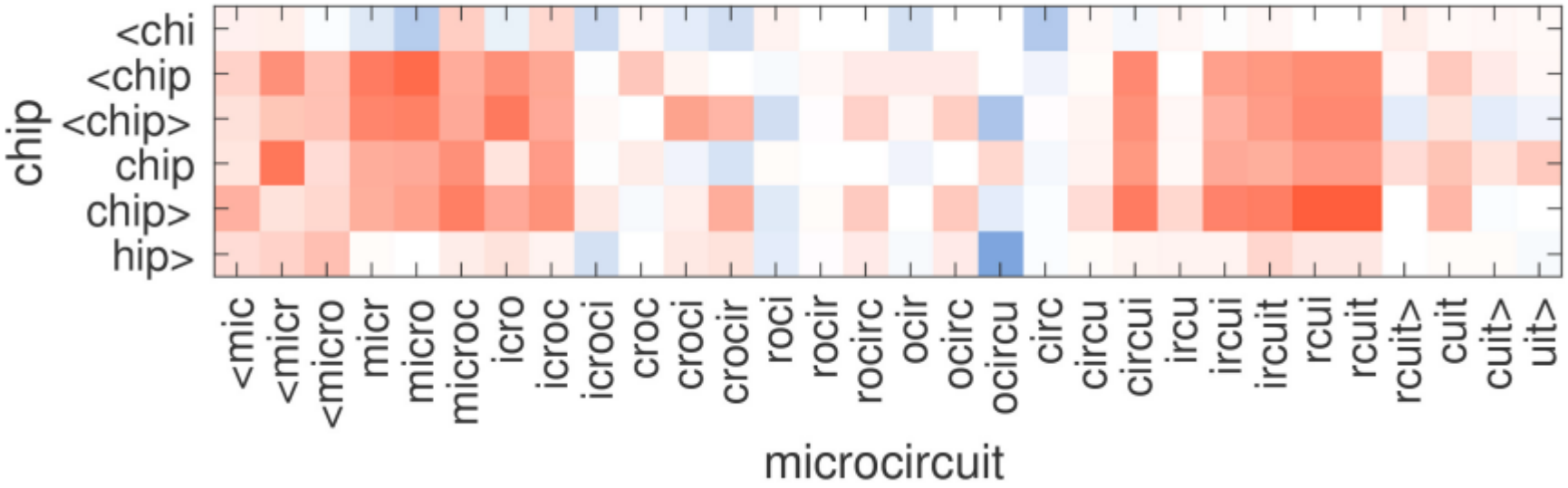
FastText

- Out-of-vocabulary words
  - possible!

$$h_w = \sum_{g \in w} x_g$$

FastText

- Matching n-grams



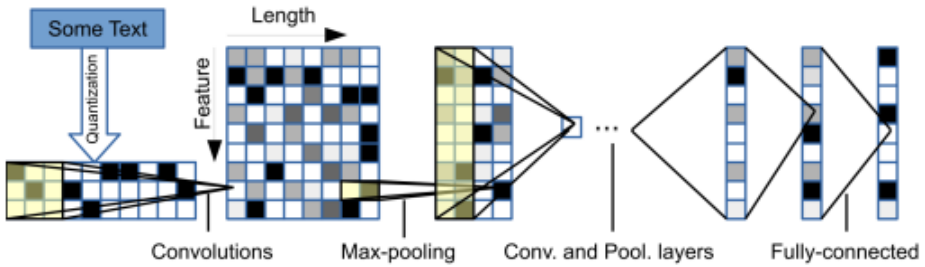
FastText downstream tasks

- Language Identification
  - Able to detect 176 languages
  - https://fasttext.cc/docs/en/language-identification.html

	TCL		Wikipedia		EuroGov	
Model	Acc.	Time	Acc.	Time	Acc.	Time
`langid.py`	93.1	8.8	91.3	9.4	98.7	13.1
`fastText`	94.7	1.3	93.0	1.3	98.7	2.9

FastText downstream tasks

- Text classification
  - Efficient Text Classification, Joulin, Grave, Bojanowski, Mikolov. EACL 2017
  - https://fasttext.cc/docs/en/supervised-tutorial.html
- Compared against:
  - Classical TF-IDF bag of n-grams + SVM
  - Deep models



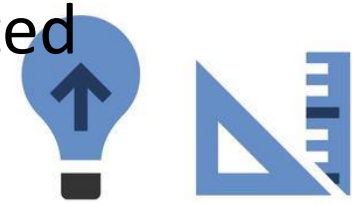
[Zhang et al., 2015]

FastText downstream tasks

- Text classification

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
`fastText`, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
`fastText`, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

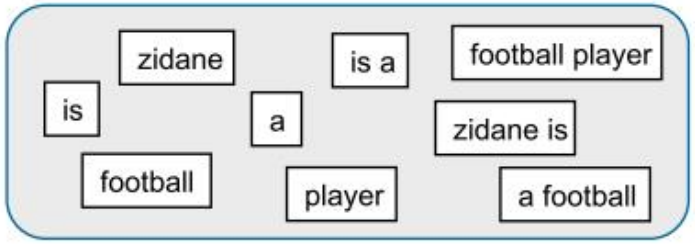
- Fast and simple, but powerful, baseline for any new complicated (deep) solution!!



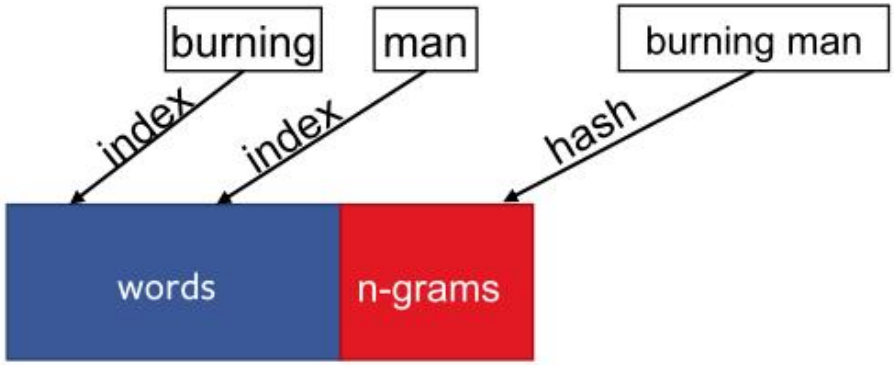
Improvements

- Word n-grams

Zidane is a football player.

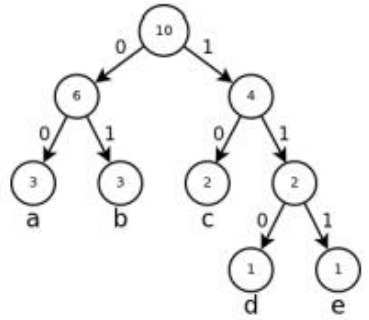


- Hashing of n-grams



- Hierarchical softmax

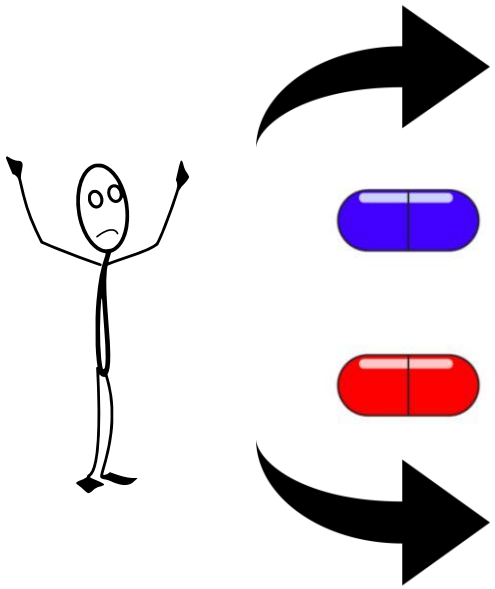
$$\frac{e^{w_k^\top x}}{\sum_{k'=1}^{K} e^{w_{k'}^\top x}}$$



- SGD with Hogwild

$$w_{(t+1)} \leftarrow w_{(t)} - \gamma \nabla_w f(w_{(t)}, x_n)$$

  - Lock-free, asynchronous updates



Query by Example

using **semantical similarity** to query text

goto: slide

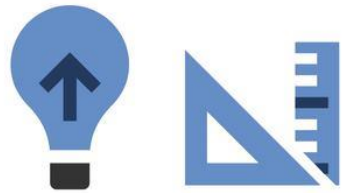
Deep Learning in NLP

using neural networks to understand context in text

goto: slide

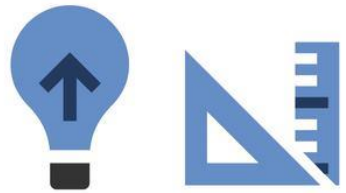
Word Movers Distance

calculating semantical similarity between sentences



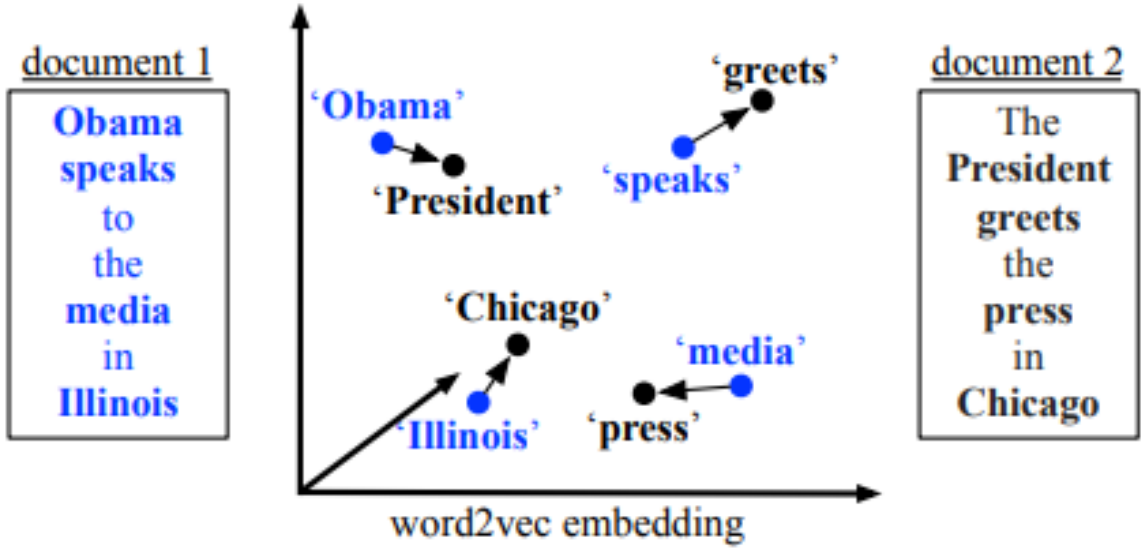
Similarity of texts

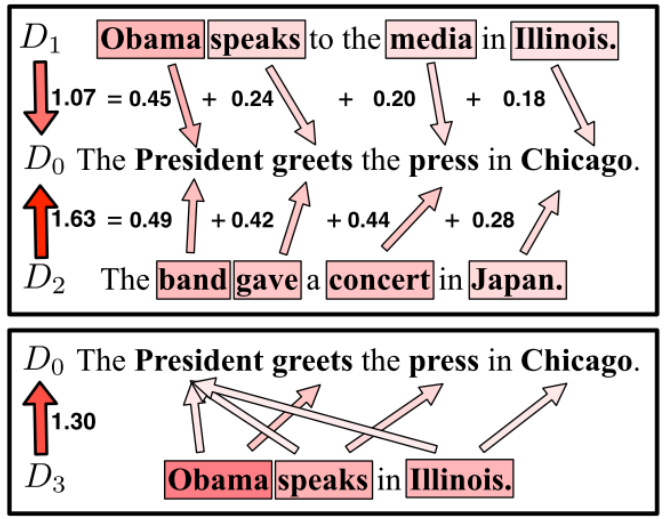
- Jaccard distance over one-hot codes
- but:
  - Sentence 1: Obama speaks to the media in Illinois
  - Sentence 2: The president greets the press in Chicago
- Cosine similarity between average embeddings of words from each document
  - loses too much information



Word Movers Distance

- Minimal matching problem
- Within embedding space





- "From Word Embeddings to Document Distances“, MJ Kusner, 2015

Word Movers Distance

- Optimization task – Transportation problem
- Solve for T (a matrix of connections between two documents)

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j)$$

$$\text{subject to:} \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1, \ldots, n\}$$

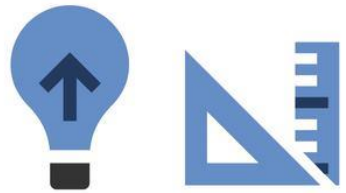
Word Movers Distance

- Problem?
- Complexity: $O(p^3 \log p)$
- Relaxations – lower-bounds:
  - Word Centroid Distance (WCD)
  - Relaxed Word Movers Distance (RWMD)
  - $O(p^2)$

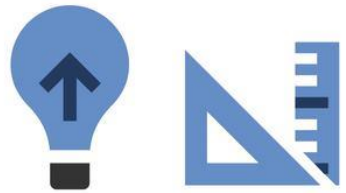
$$\mathbf{T}^*_{ij} = \begin{cases} d_i \text{ if } j = \operatorname{argmin}_j c(i,j) \\ 0 \text{ otherwise.} \end{cases}$$

Query by example

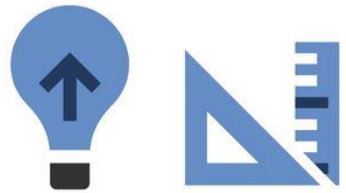
- Application of WMD
- Given a relevant item, find the closest:
  - Article from the document archive
  - Resume from candidate profiles
  - Patient from medical records
  - Movie from movie database
  - Review from review database
  - …



Query by Example demo

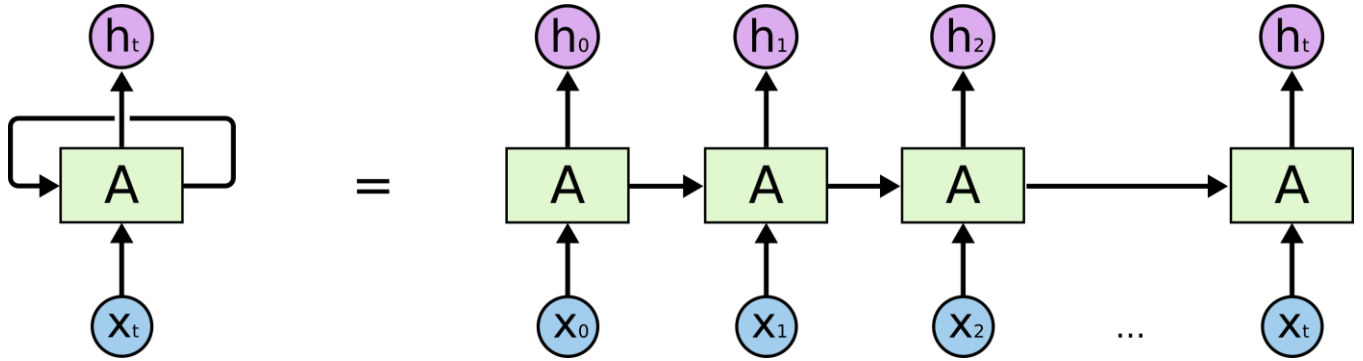


Deep Learning approaches to NLP



Recurrent networks approach (RNN)

- Encodes words in hidden layers (embedding)
- Captures structure
- Can be used for:
  - word tagging
  - text classification (single output)

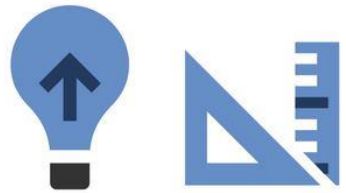


- Hard to model longer dependancies (vanishing gradients)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

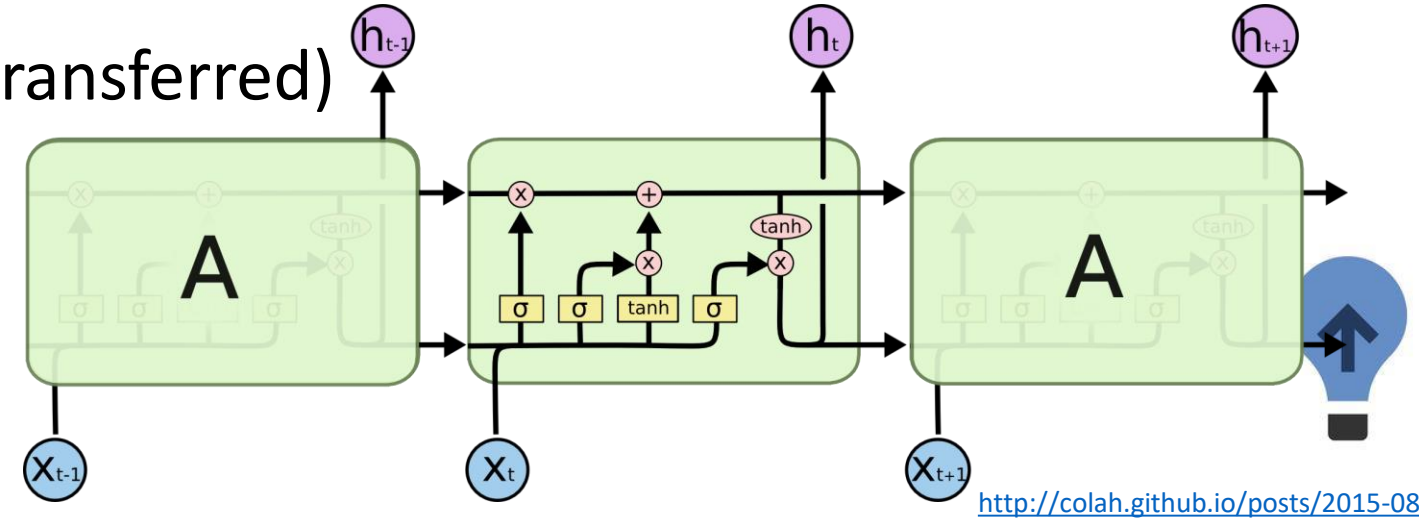
Longer memory/context LMs

- Context in usually only several neighboring words
- Problem:
  - „I met Miloš yesterday, and after talking for a while, I agreed to meet with _____ again“ ?
- Especially important for Question answering and conversational systems



LSTM (Long Short Term Memory)

- A type of RNN
- 1997. original paper, 2015. in Google Voice, 2016. in Google Translate
- Uses **gates** to selectively chose what part of hidden state gets **modified**, what gets **carried over**, and what gets **rewritten**
- Addition instead of propagation (for vanishing gradients)
- Hard to train
- Cannot be reused (transferred)

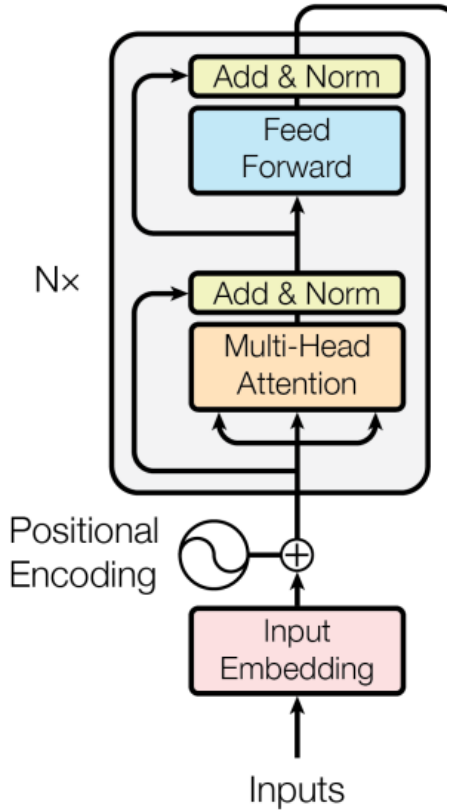


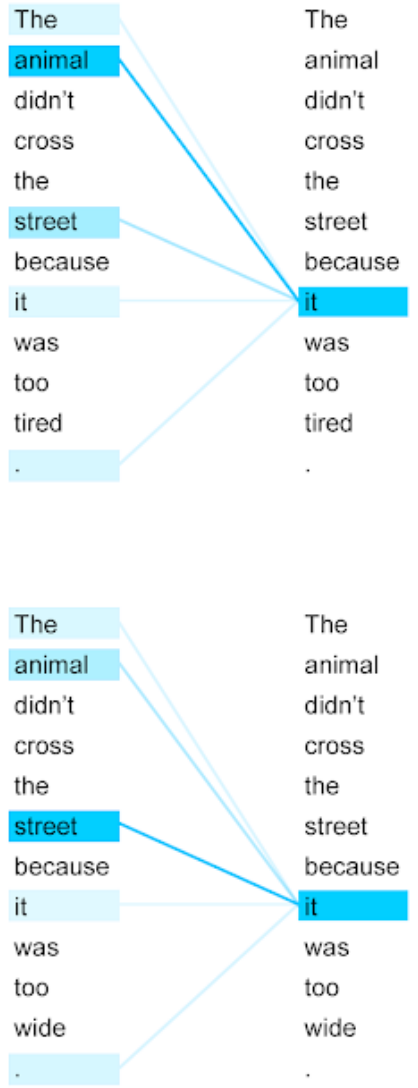
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Transformer Networks

- Not recurrent, takes whole text as input
- Encodes position using sin/cosine waves with different frequencies and relative positions
- **Attention mechanism** (multi-head)
  - N^2 additional parameters
- Interpretable
- Parallelizable
  - even N^2 easy on GPU
- Transfer!

- Vaswani A, Shazeer N, Parmar N, Uszkoreit j, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017). **Attention Is All You Need**, NIPS 2017.

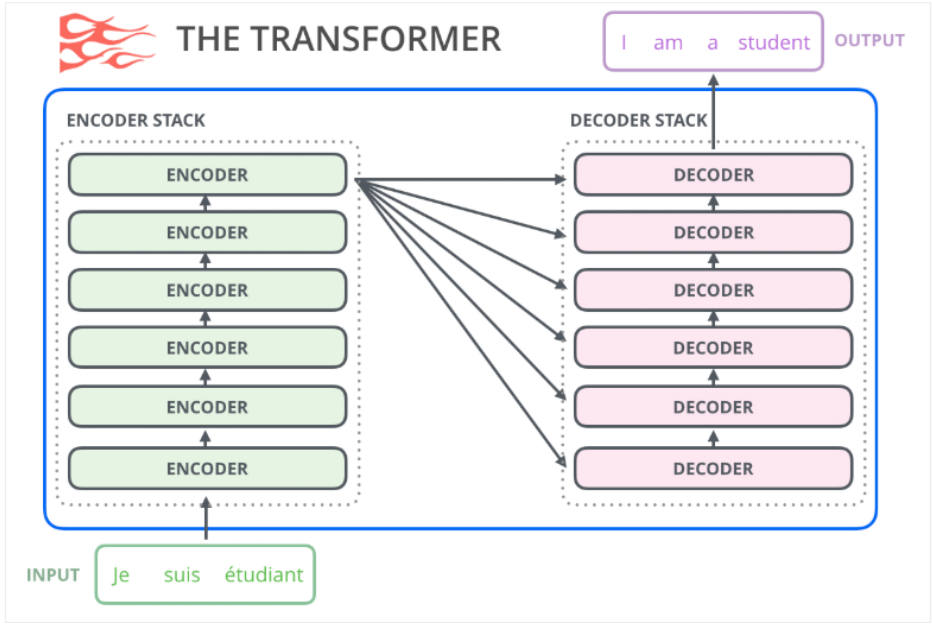


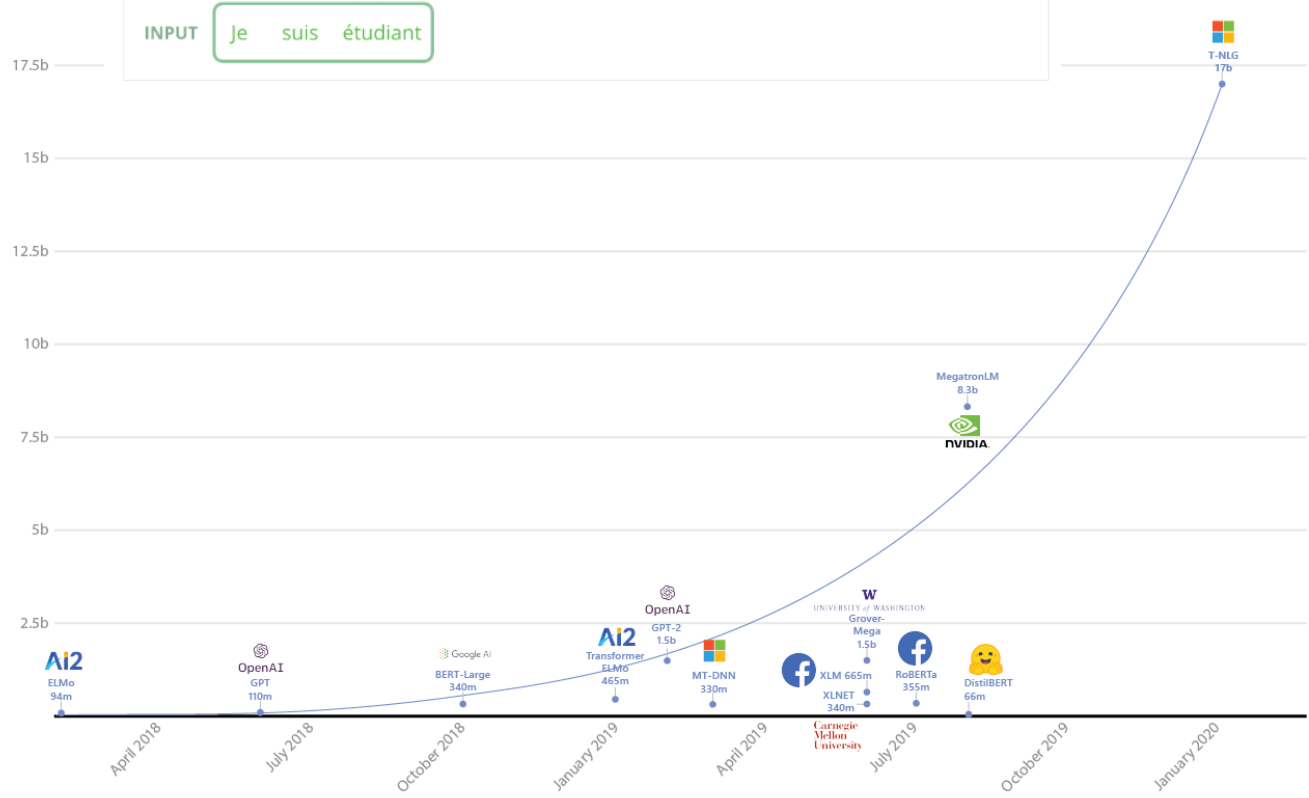


Transformer LM Race

- Tranformers for NLP original paper (2017)
- BERT (Google 2018; $7K)
- RoBERTa (Facebook 2019)
- XLNet (Google 2019; $250K)
- Megatron-LM (Nvidia 2019; $450K)
- Turing-NLG (Microsoft 2020)
- GPT3 (OpenAI 2020, $12M)

Most are freely available for transfer learning.





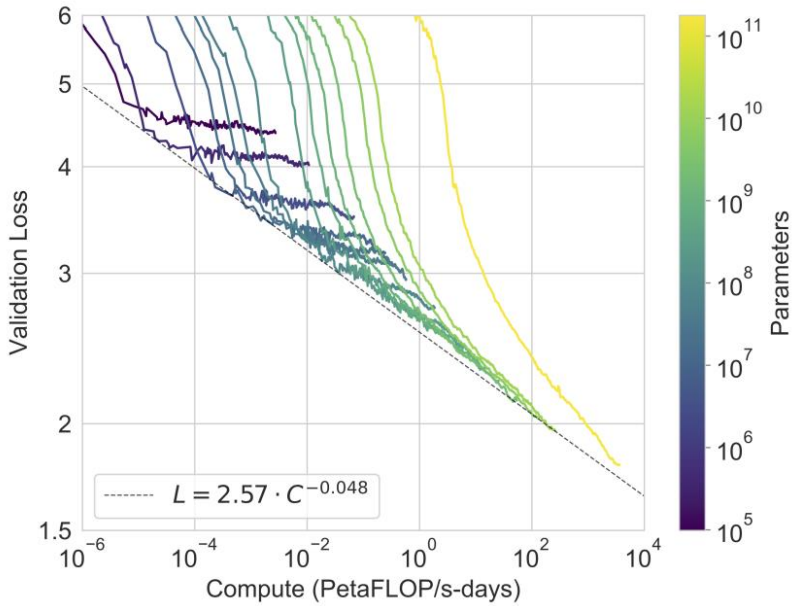
GPT3

- Brown et al (2020). Language Models are Few-Shot Learners

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```



Alice was friends with Bob. Alice went to visit her friend _____. → Bob

George bought some baseball equipment, a ball, a glove, and a _____. →

"Q: What is 48 plus 76? A: 124."

Questions?

THANK YOU