Tech Talks

String processing

NLP, biology, finance

Plan

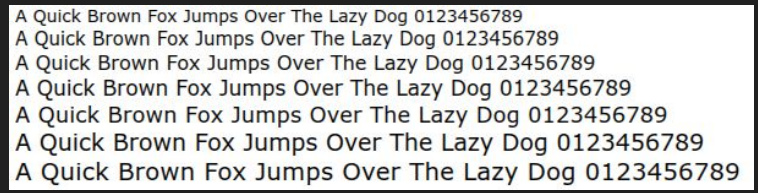
- Introduction to string processing
- String indexing. Suffix trees
- Sequence alignment
- Word embeddings. Word2Vec

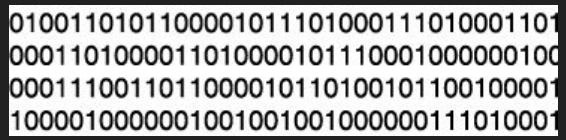
Introduction to String Processing

Strings

- **String** = *an ordered list of characters written contiguously from left to right*

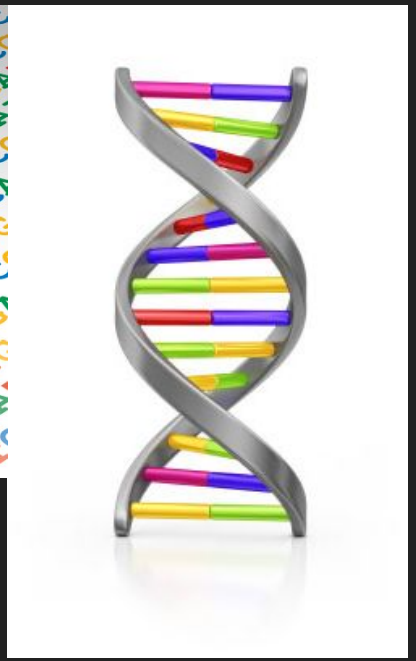
- **Characters** come from an alphabet:
  - english (A, B, C, …)
  - binary (0, 1)





More alphabets ⇨ more interesting strings!

- 4 nitrogen-containing nucleobases:
  - cytosine (**C**)
  - guanine (**G**)
  - adenine (**A**)
  - thymine (**T**)



⇨ **Genome**

More alphabets ⇨ more interesting strings!

- Let each letter be a tuple containing:
  - id
  - timestamp
  - current balance
  - ...

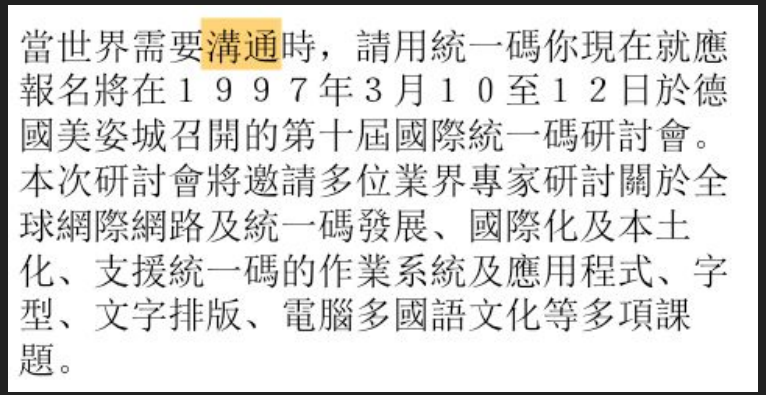
⇨ **Transactions**

	A	B	C	D	E	F
1	Table: Transaction				+: Credit, -: Debit	
2	Transaction ID	Transaction Date Time	User ID	Account ID	Amount	Account Balance
3	1000001	01/04/2012 09:10:19	2	1	3100.00	4,300.21
4	1000002	01/04/2012 11:10:19	4	3	5800.00	6,412.44
5	1000003	01/04/2012 12:10:19	3	4	1200.00	307.85
6	1000004	01/04/2012 13:10:19	1	5	2500.00	229.87
7	1000005	02/04/2012 09:10:19	5	1	-50.00	4,250.21
8	1000006	02/04/2012 11:10:19	3	3	-100.00	612.44
9	1000007	02/04/2012 14:10:19	1	6	810.00	-99,119.91
10	1000008	03/04/2012 09:10:19	3	1	-50.00	4,200.21
11	1000009	03/04/2012 11:10:19	1	3	-100.00	512.44
12	1000010	03/04/2012 14:10:19	5	6	810.00	-98,309.91

Stringology

- Science of **algorithms and data structures** on strings
- Many common problems across different fields

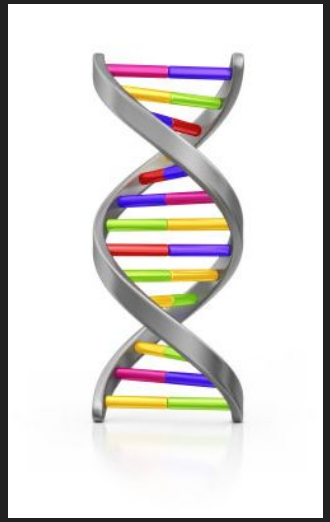
Example: **word separation** in natural language processing...



Stringology

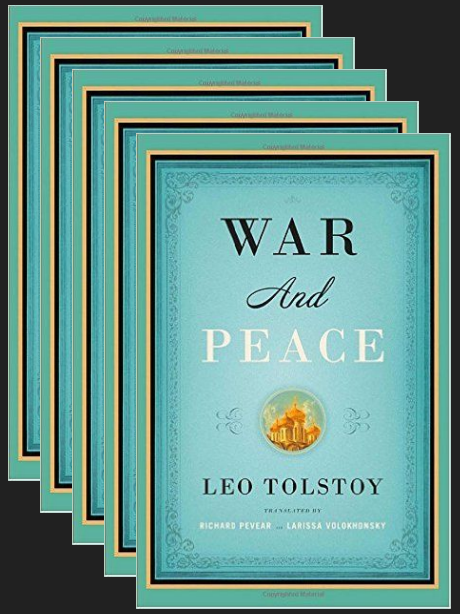
… and **identification of genes** in computational biology

Massive strings



Human genome
~ 3 billion letters

=



5 * ”War and Peace”

String Indexing

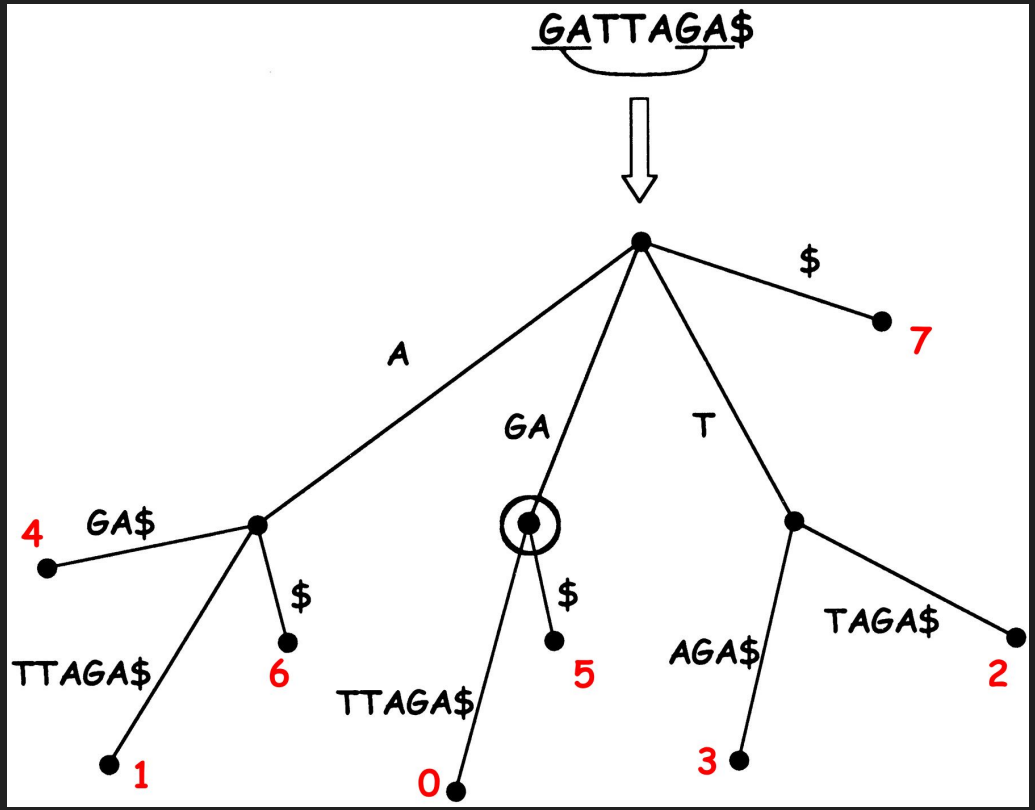
String indexing

- **Problem:** fast searches in big texts
- **Idea:** if the text is static, we can try to index it

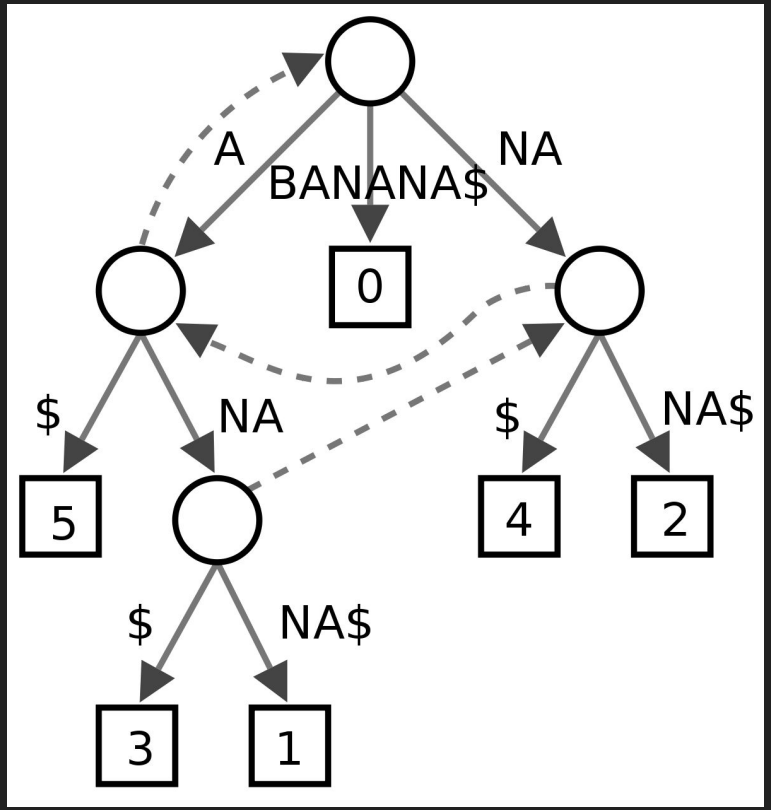
Suffix trees: major breakthrough in the 1970s

- O(n) indexing (n = |Text|)
- O(m) queries (m = |Query|)

Suffix trees in biology



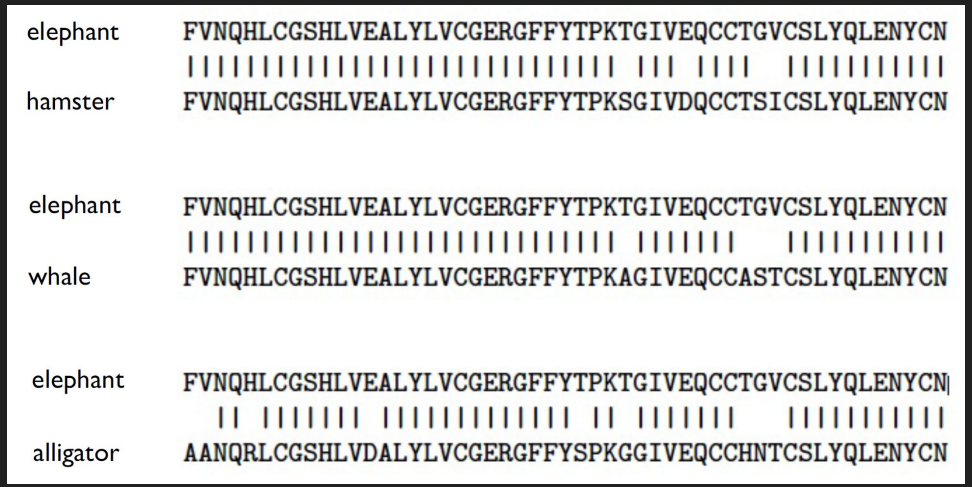
Suffix trees in NLP



Sequence alignment

Sequence alignment

- Tool for **comparing genomes** and **finding similarities**
- One of the most important and well-studied problems in computational biology



Sequence alignment

```
R D I S L V - - - K N A G I
|   |   | |       | |   | |
R N I - L V S D A K N V G I
```

- 3 types of columns corresponding to 3 elementary evolutionary events:
  a. match
  b. substitution (mismatch)
  c. Insertion / deletion

Sequence alignment

```
R D I S L V - - - K N A G I
|   |   | |       | |   | |
R N I - L V S D A K N V G I
```

- Assign a score (positive or negative) to each event
- Alignment score = sum (scores over all columns)
- Optimal alignment = one that maximizes the score

Sequence alignment

- Scoring function:

Mismatch :		Match :		Indel :
		G, N	: 6	
DN	: 1	R, K	: 5	−5
AV, LD	: 0	A, I, L, S, V	: 4	

- Optimal scores:

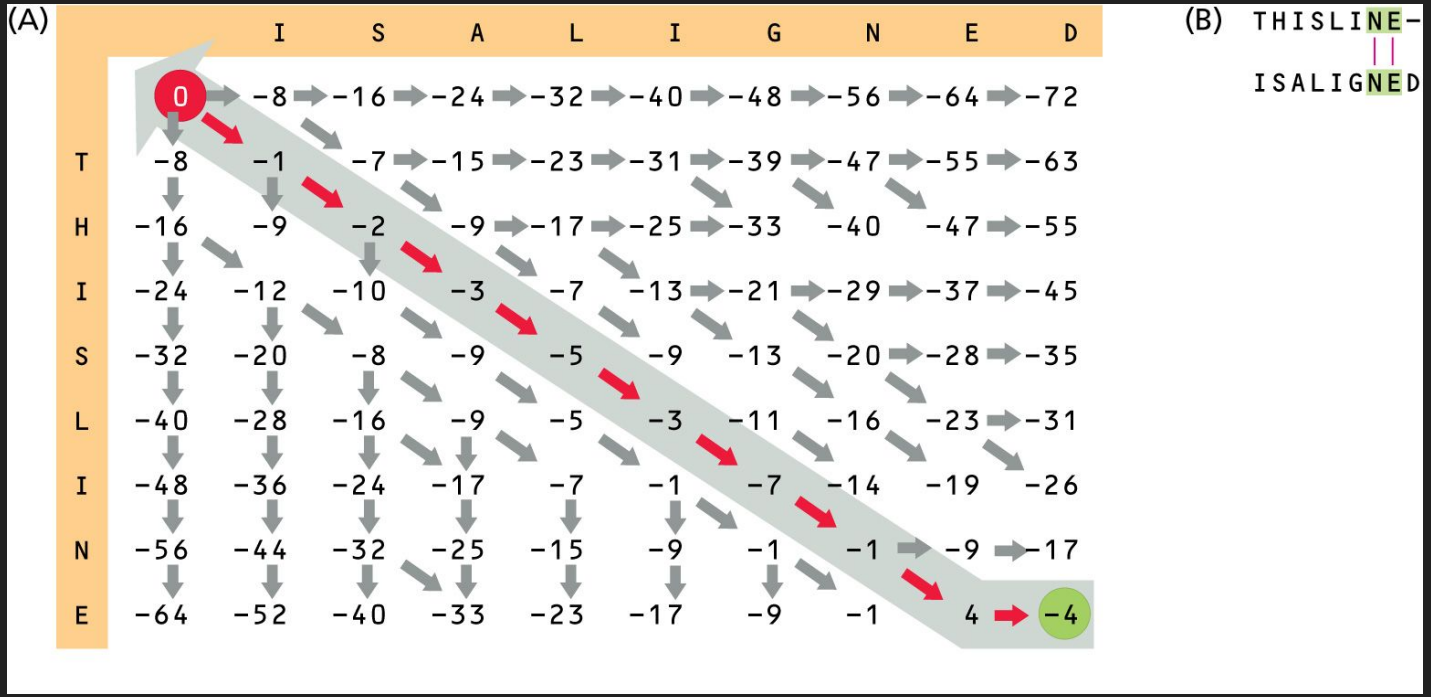
```
R D I S L V - - - K N A G I      R D I - - S L V K N A - - - G I
|   |   | |       | |   | |      |   |     |         |       | |
R N I - L V S D A K N V G I      R N I L V S - - - D A K N V G I
```

Score=19 Score=-11

```
          R D I - - S L V K N A G I
          |   |     |     | |   | |
Score=25  R N I L V S D A K N V G I
```

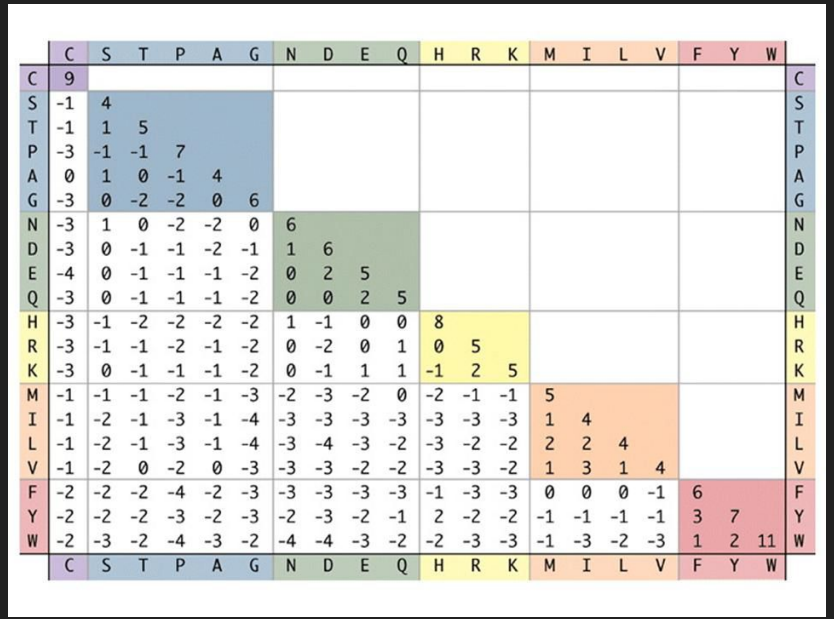
Sequence alignment

- Can be solved with **dynamic programming**



Sequence alignment

- Bioinformaticians come up with special matrices for scoring functions
- E.g. BLOSUM62 for protein sequences:



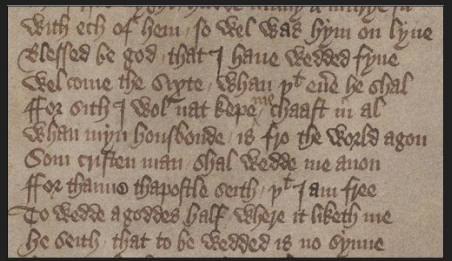
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

How can all this be useful for NLP?

- Alignment of texts in natural languages

An	interesting event	occurs	at the Hôtel de Lauzun today
A	seminar	is held today	at the Hôtel Pimodan

- Digital humanities: analysis of historical texts
  - Old Texts have been evolving over time (copyists ...)
  - Again, **evolutionary events!**

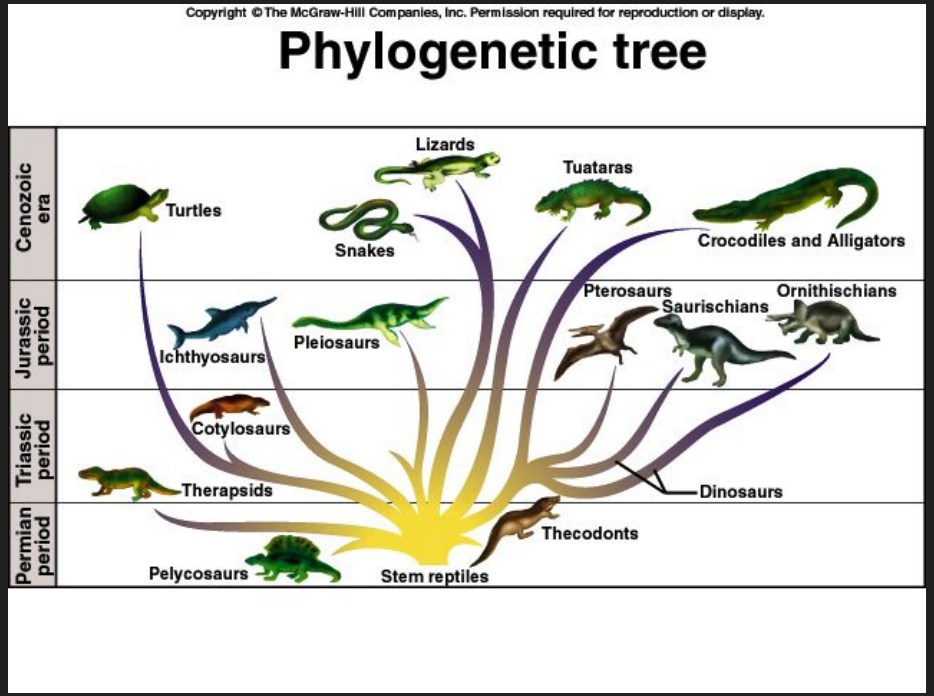


Historical text alignment

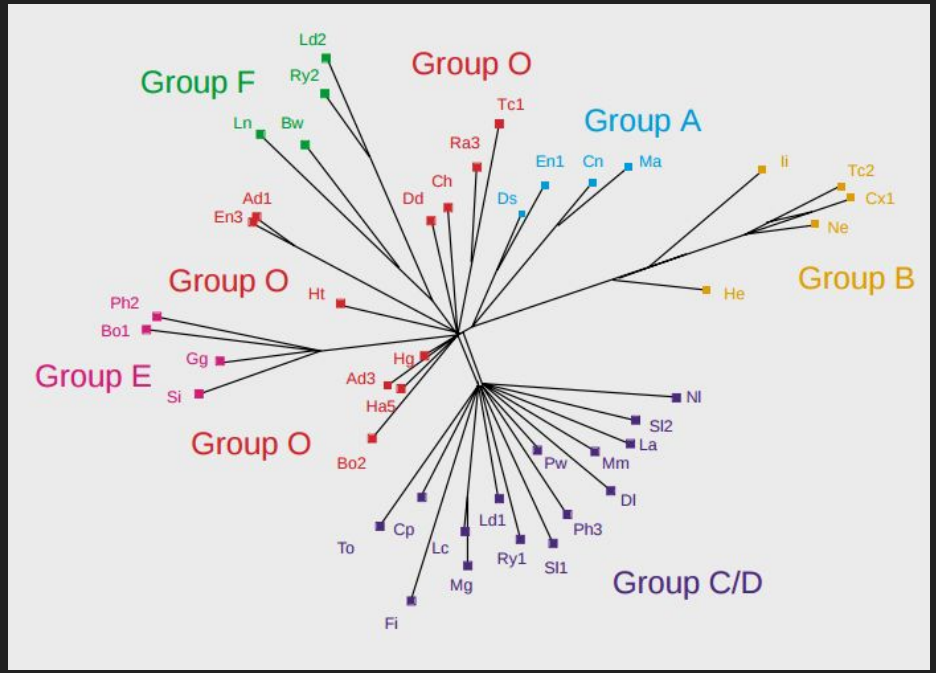
ffourtene	yeere	he	bare	his	crowne	l	reede
xiiij^{e}	yere	he	bare	his	crowne	in	dede
xiiij^{e}	yere		bare	his	corone	in	dede
ffourtene	yere	he	bare	his	croune	l	rede
ffourtene	yer^{e}	bare	he	his	crowne	in	dede
fortene			bare	hys	crown	in	dede
Bare th^{e} crowne	xij yere xi monthes & xvi dayes					in	dede

- John Lydgate, Kings of England, 15th cent.
- C.Howe, A.Barbrook, Manuscript evolution [2001]

Phylogenetic trees...



Phylogenetic trees… for texts!



Word2Vec

Word2Vec

- Most NLP algorithms require words and documents to be represented as vectors:

king = [1 0 0 0.. 0 0 0 0 0]

queen = [0 1 0 0 0 0 0 0 0]

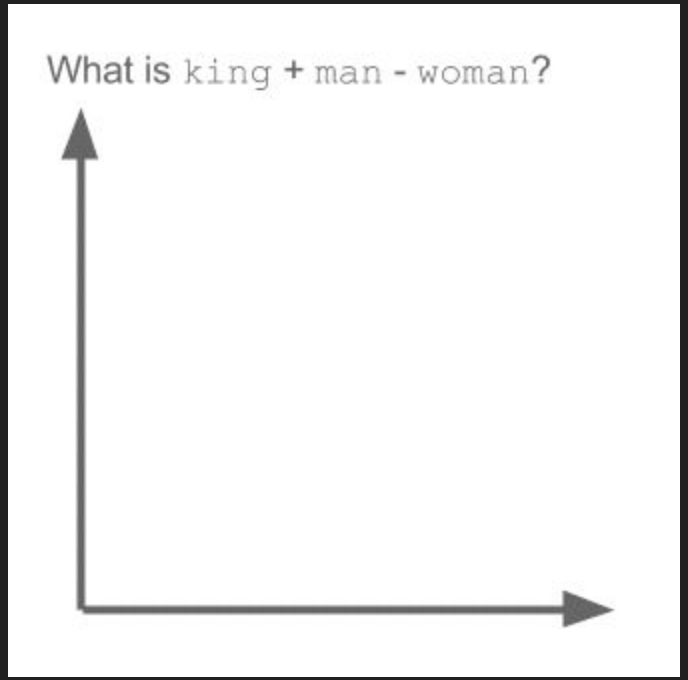
book = [0 0 1 0 0 0 0 0 0]

- This is a very high-dimensional representation. We would be much happier with something like this:

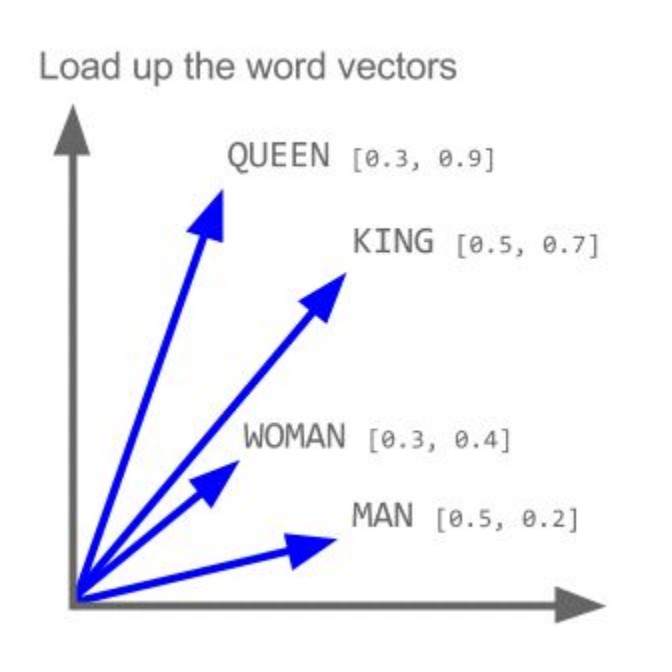
~~king = [1 0 0 0.. 0 0 0 0 0]~~

king = [0.9457, 0.5774, 0.2224]

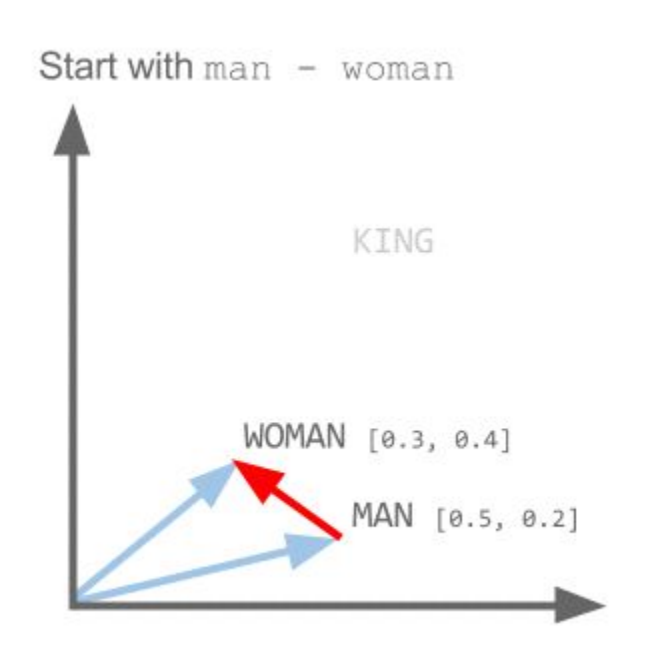
Word2Vec



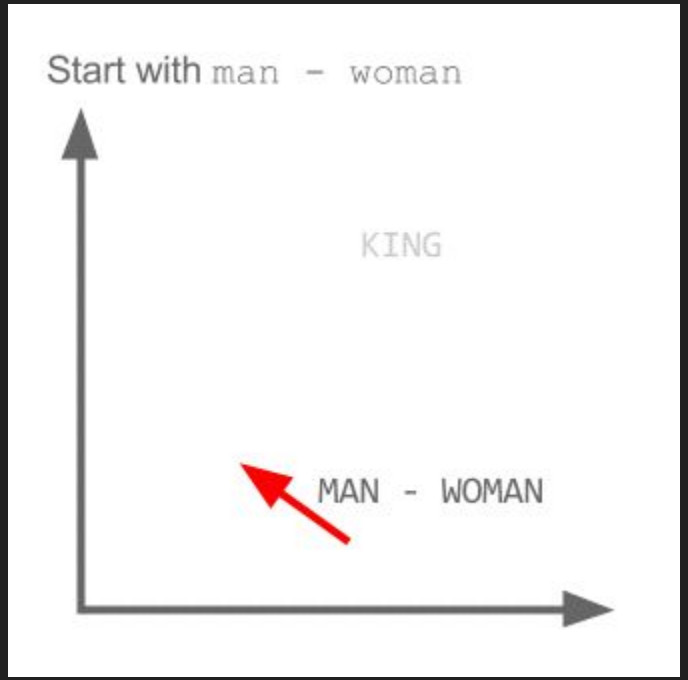
Word2Vec


Load up the word vectors
QUEEN [0.3, 0.9]
KING [0.5, 0.7]
WOMAN [0.3, 0.4]
MAN [0.5, 0.2]

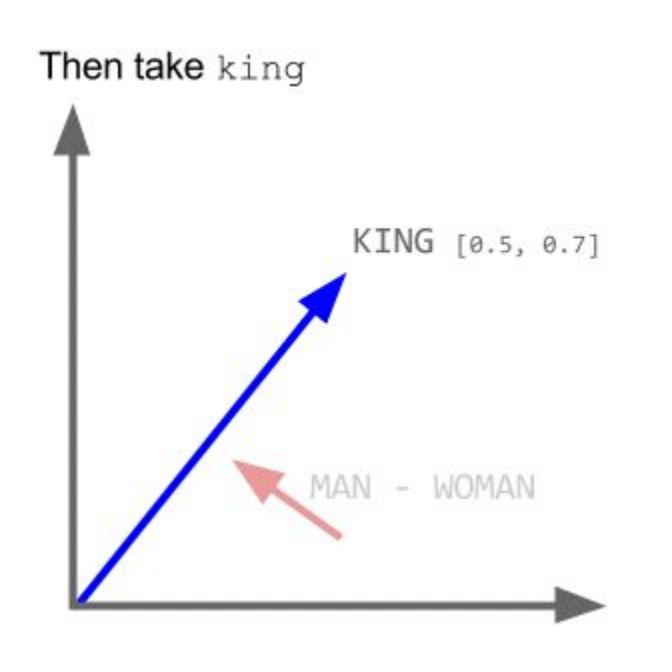
Word2Vec



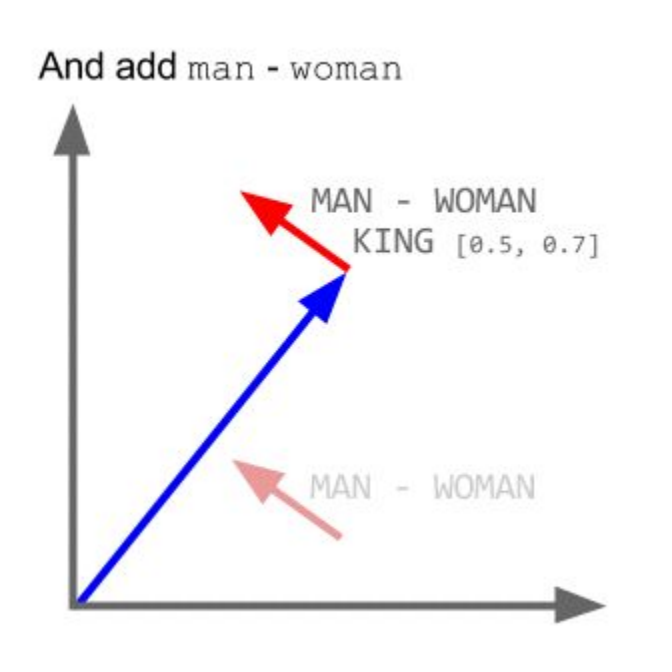
Word2Vec



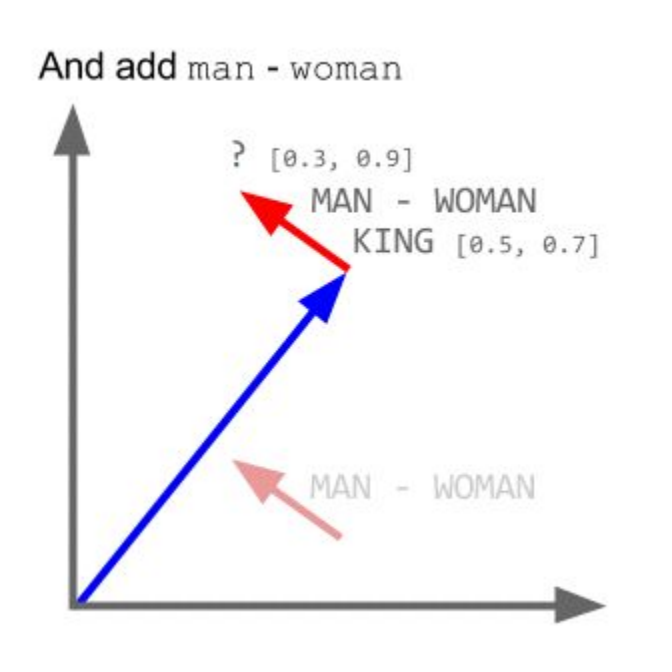
Word2Vec



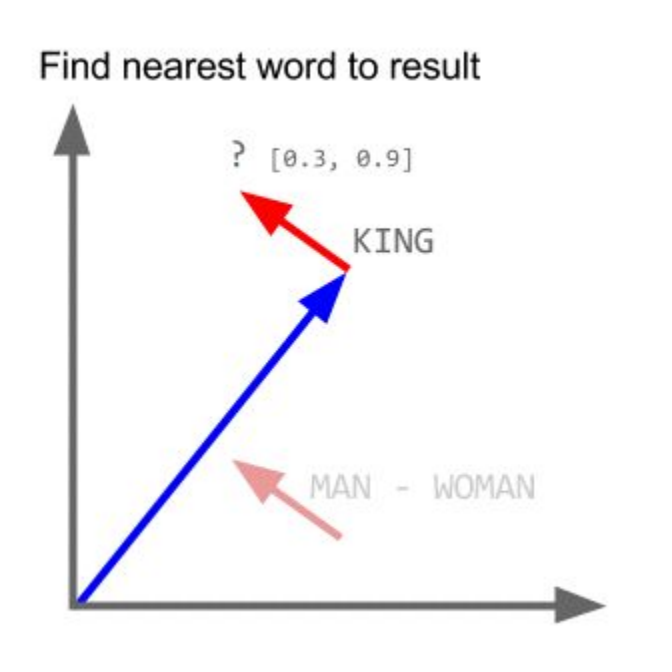
Word2Vec



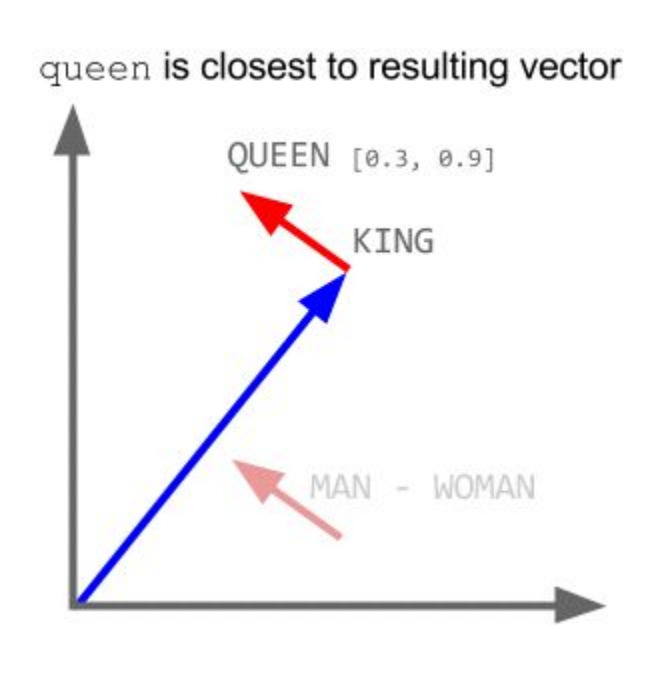
Word2Vec



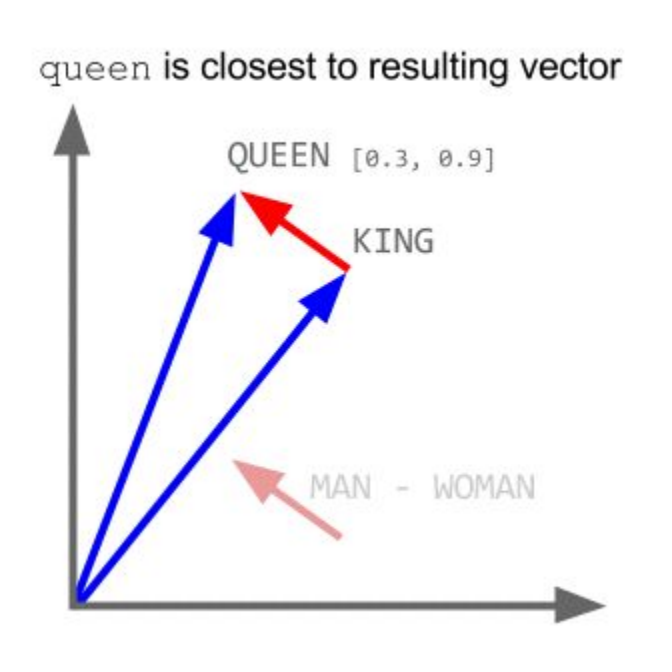
Word2Vec



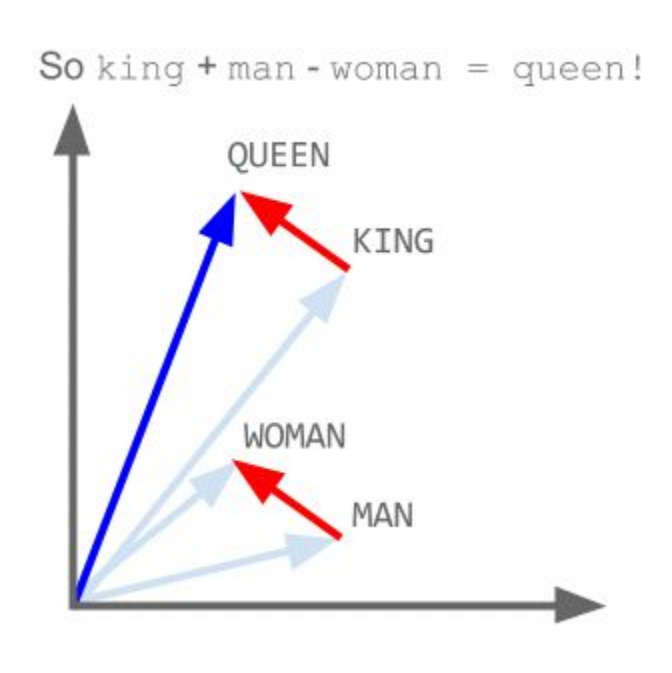
Word2Vec


queen is closest to resulting vector
QUEEN [0.3, 0.9]
KING
MAN - WOMAN

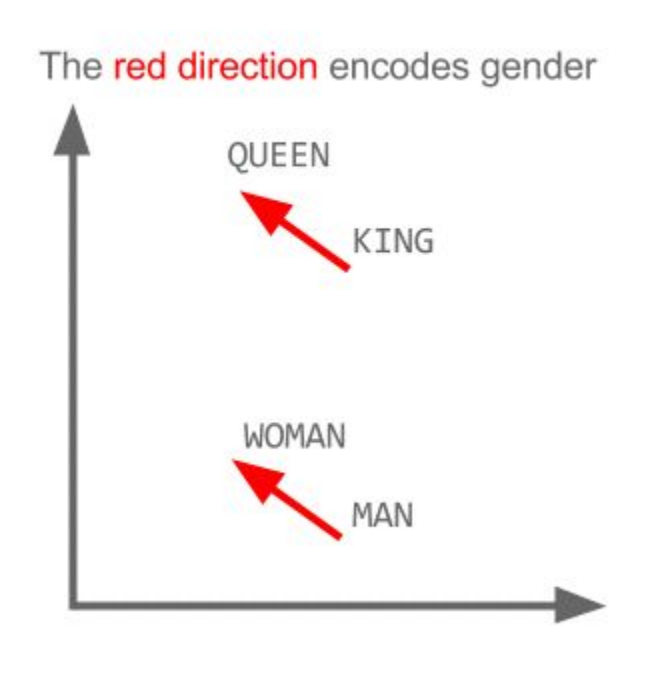
Word2Vec


queen is closest to resulting vector
QUEEN [0.3, 0.9]
KING
MAN - WOMAN

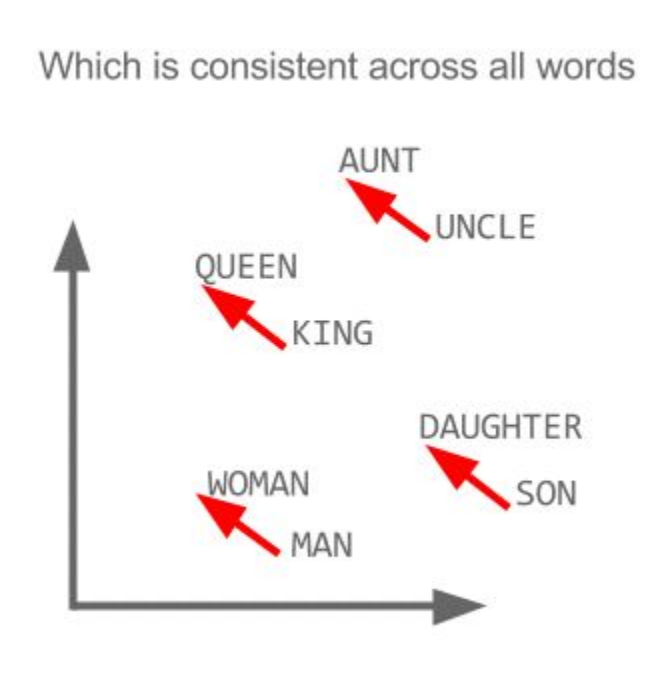
Word2Vec



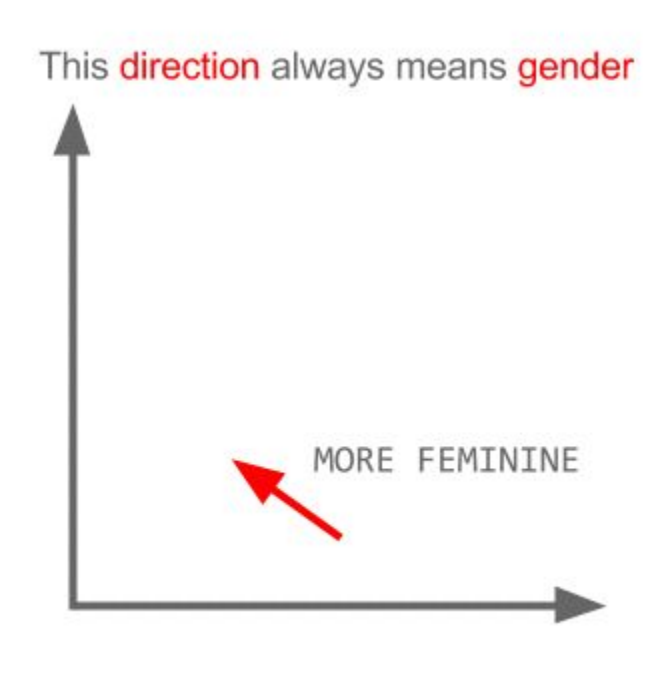
Word2Vec



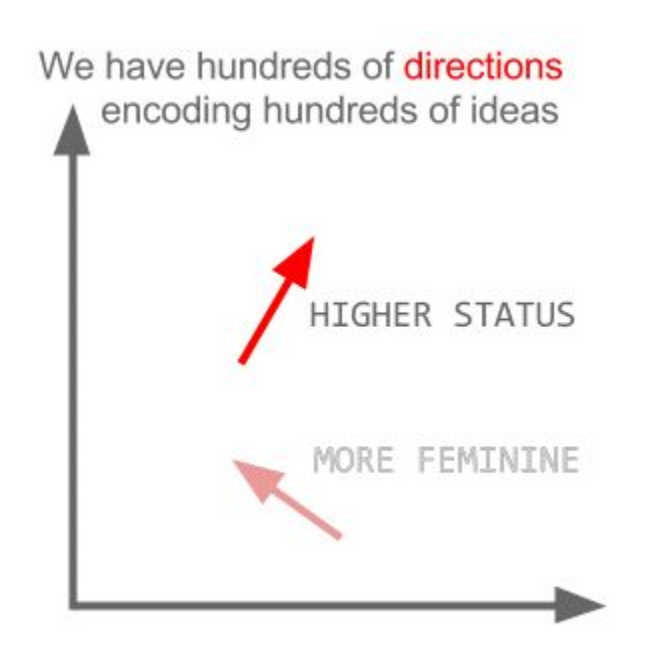
Word2Vec


Which is consistent across all words
AUNT
UNCLE
QUEEN
KING
DAUGHTER
SON
WOMAN
MAN

Word2Vec



Word2Vec


We have hundreds of directions
encoding hundreds of ideas
HIGHER STATUS
MORE FEMININE

Word2Vec

Averaging word vectors aka ‘Naive document vector’

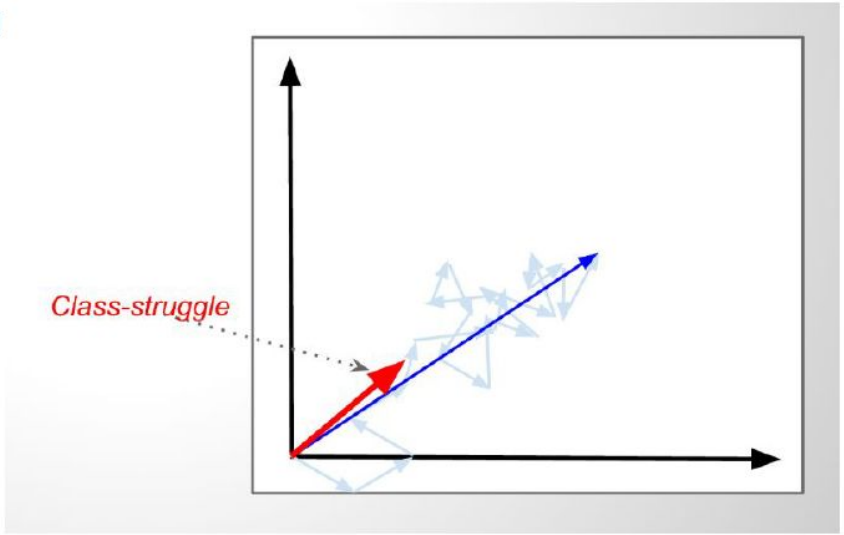
Just add word vectors together!

All words in a book

‘A tale of two cities’

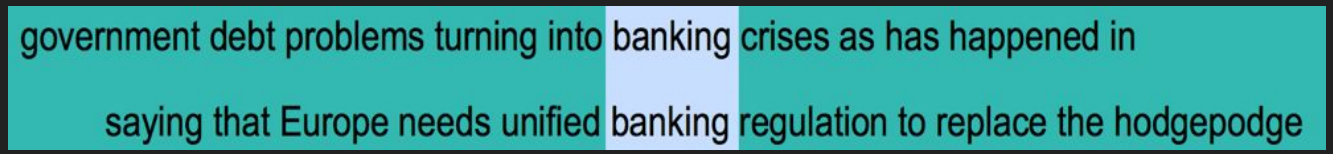
Should add up to

‘class-struggle’



Word2Vec

- Trained on texts as **sequences of words**, not letters
- **Alphabet** = all words in the corpus
- Fits vectors based on the context of each word:



What if we train Word2Vec on texts in some other alphabet?

Word2Vec - Financial data

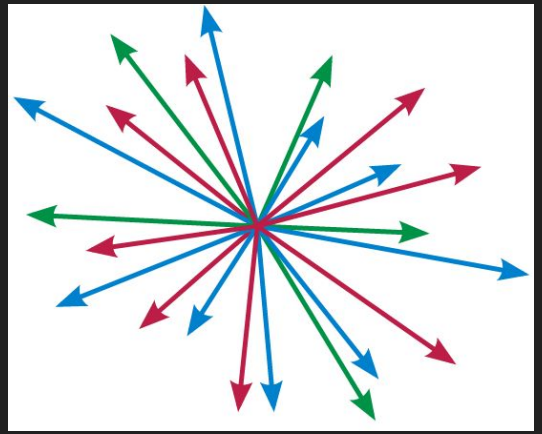
- Sberbank Data Science Hakhathon 2016

MCC Code	Transaction time	...
4814	2016-09-02 10:52:11	...
4814	2016-09-02 14:13:15	...
6010	2016-09-02 21:33:44	...
6011	2016-09-03 13:00:03	...
4814	2016-09-04 12:34:58	...
5003	2016-09-04 19:41:32	...

- MCC Code = type of transaction (ATM withdrawal / payment at a restaurant / …)

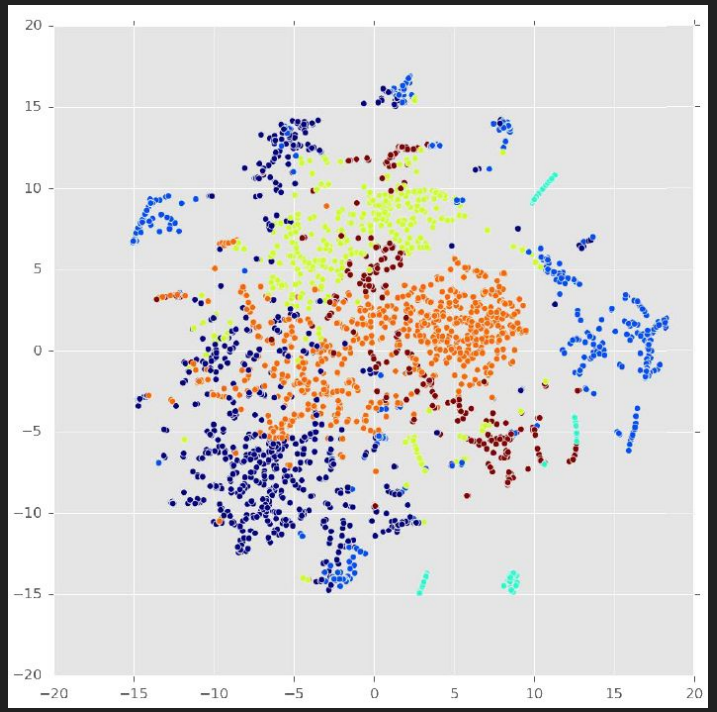
Step 1: Mcc2Vec

- Each transaction = a “word”
- Transactions form “sentences” groups of transactions split by < 12 hours
- Context for each transaction = its sentence
- Word2Vec on transactions ⇨ embedding MCC codes into a vector space

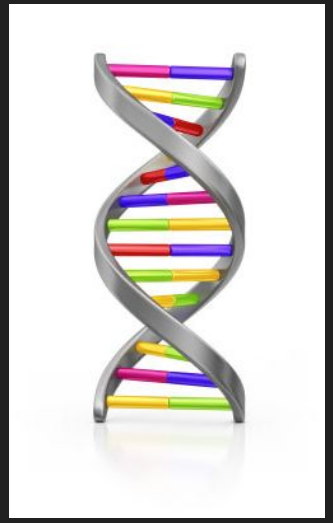


Step 2: Client2Vec

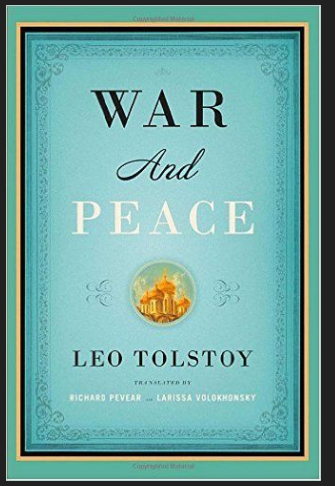
- Client vector = sum of MCC vectors (~100-dimensional)
- t-SNE on client vectors:



⇨ **customer segmentation!**



Thank you


WAR
And
PEACE
LEO TOLSTOY
RICHARD PEVEAR and LARISSA VOLOKHONSKY