Lecture 12

Natural Language Processing

Some of contents are adapted from A. Ng

Natural Language Processing

- A sub-field of Artificial Intelligence (AI)
- An inter disciplinary subject
- To build intelligent computers that can interact with human
- Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts/speech at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.
- The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language

https://en.wikipedia.org/wiki/Natural_Language_Toolkit

Term-document count matrices

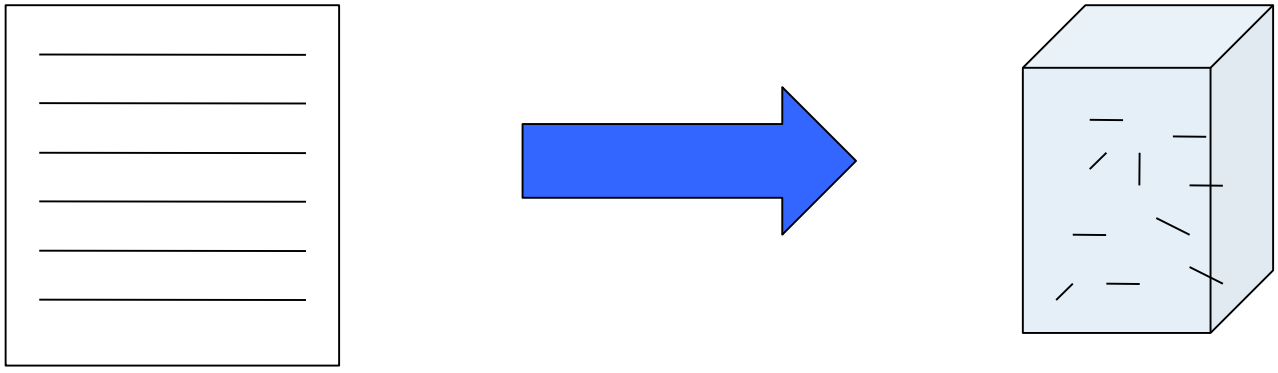
- Consider the number of occurrences of a term in a document:
  - Each document is a ***count vector*** in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

What information is lost with this representation?

Bag of words representation

- Represent a document by the occurrence counts of each word
- **Ordering** of words is lost
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors



CountVectorizer: count the number of times a word appears in the document

https://medium.com/swlh/sentiment-classification-with-bow-202c53dac154

Document frequency

- We will use document frequency (df) to capture this in the score

- Terms that occur in many documents are weighted less, since overlapping with these terms is very likely
  - In the extreme case, take a word like the that occurs in EVERY document

- Terms that occur in only a few documents are weighted more

Term Frequency

https://en.wikipedia.org/wiki/Tf-idf

Term frequency, tf(t,d), is the frequency of term t,

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

the raw count itself: tf(t,d) = $f_{t,d}$

Inverse document frequency

$$\mathrm{idf}_t = \log N/\mathrm{df}_t$$

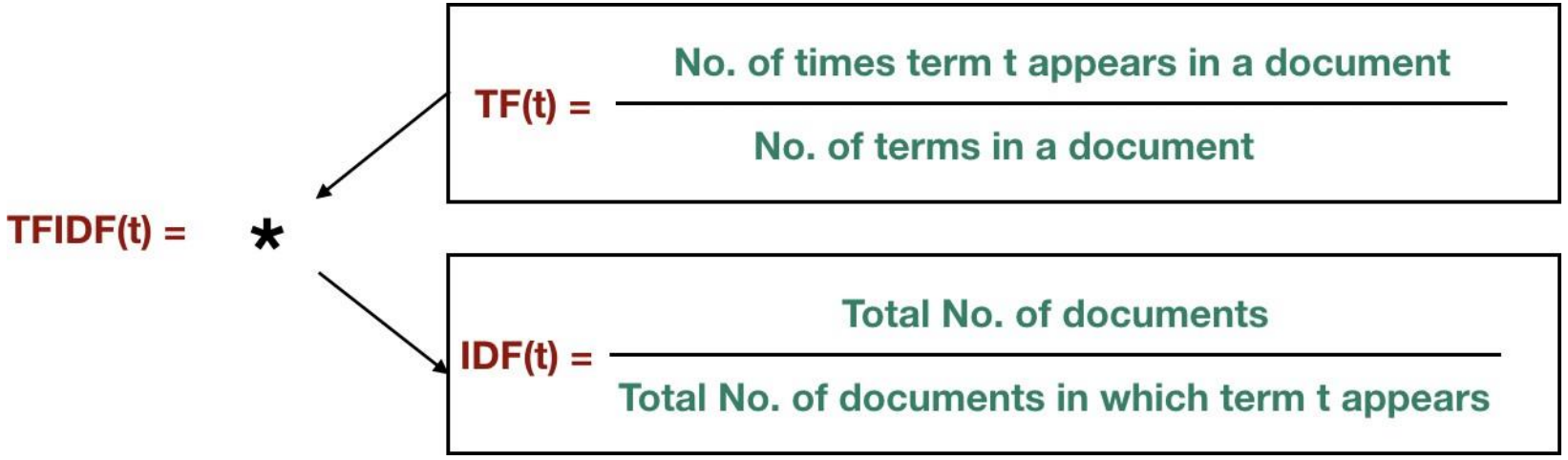
N: No of sentences

df: No of sentences containing words

Term frequency–Inverse document frequency (tf-idf)

Example of tf–idf : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

- Can we put all of these together?
  - Define a weighting for each term
  - The tf-idf weight of a term is the product of its tf weight and its idf weight



TfidfVectorizer: overall document weightage of a word. It helps us in dealing with most frequent words

https://medium.com/swlh/sentiment-classification-for-restaurant-reviews-using-tf-idf-42f707bfe4_4d

https://en.wikipedia.org/wiki/Word2vec

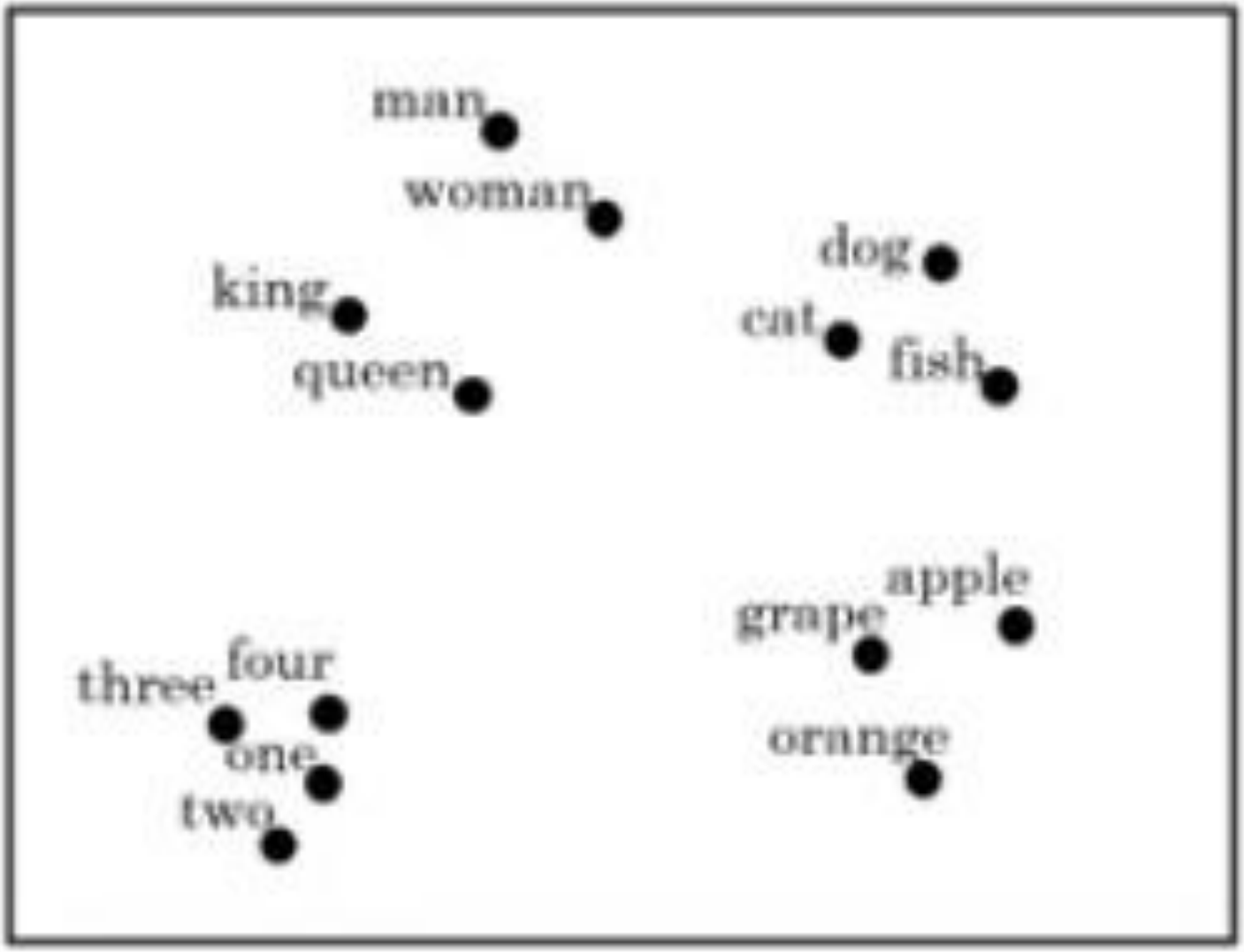
Word2vec (wiki)

https://en.wikipedia.org/wiki/Gensim

- **Word2vec** is a group of related models that are used to produce word embeddings.
- **Word embedding** is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers
- Methods to generate this mapping include neural networks, dimensionality reduction on the word co-occurrence matrix, probabilistic models.
- Word2vec was created and published in 2013 by a team of researchers led by ***Tomas Mikolov*** at Google and patented.
- Efficient Estimation of Word Representations in Vector Space https://arxiv.org/pdf/1301.3781.pdf

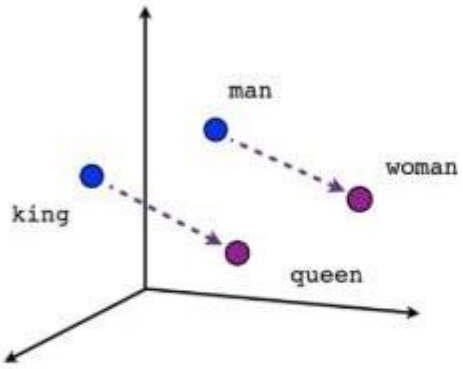
https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca

Visualizing Word Embedding

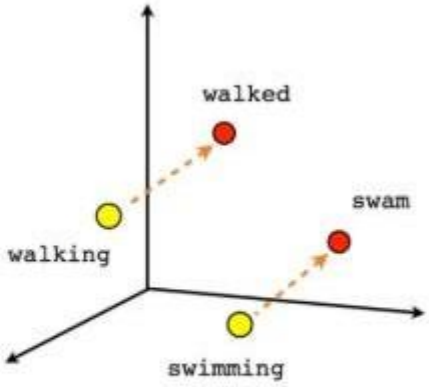


t-SNE

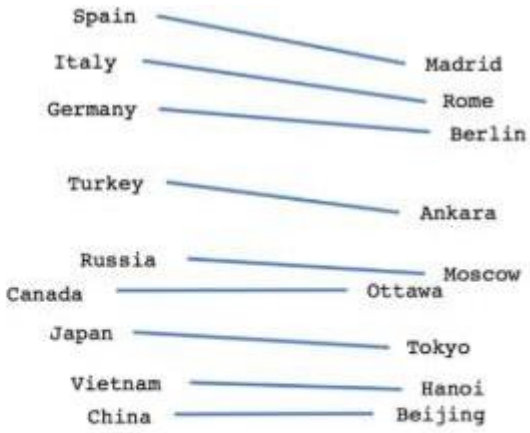
- https://www.tensorflow.org/tutorials/text/word2vec



Male-Female



Verb tense



Country-Capital

NN: Featured Representation: Word Embedding

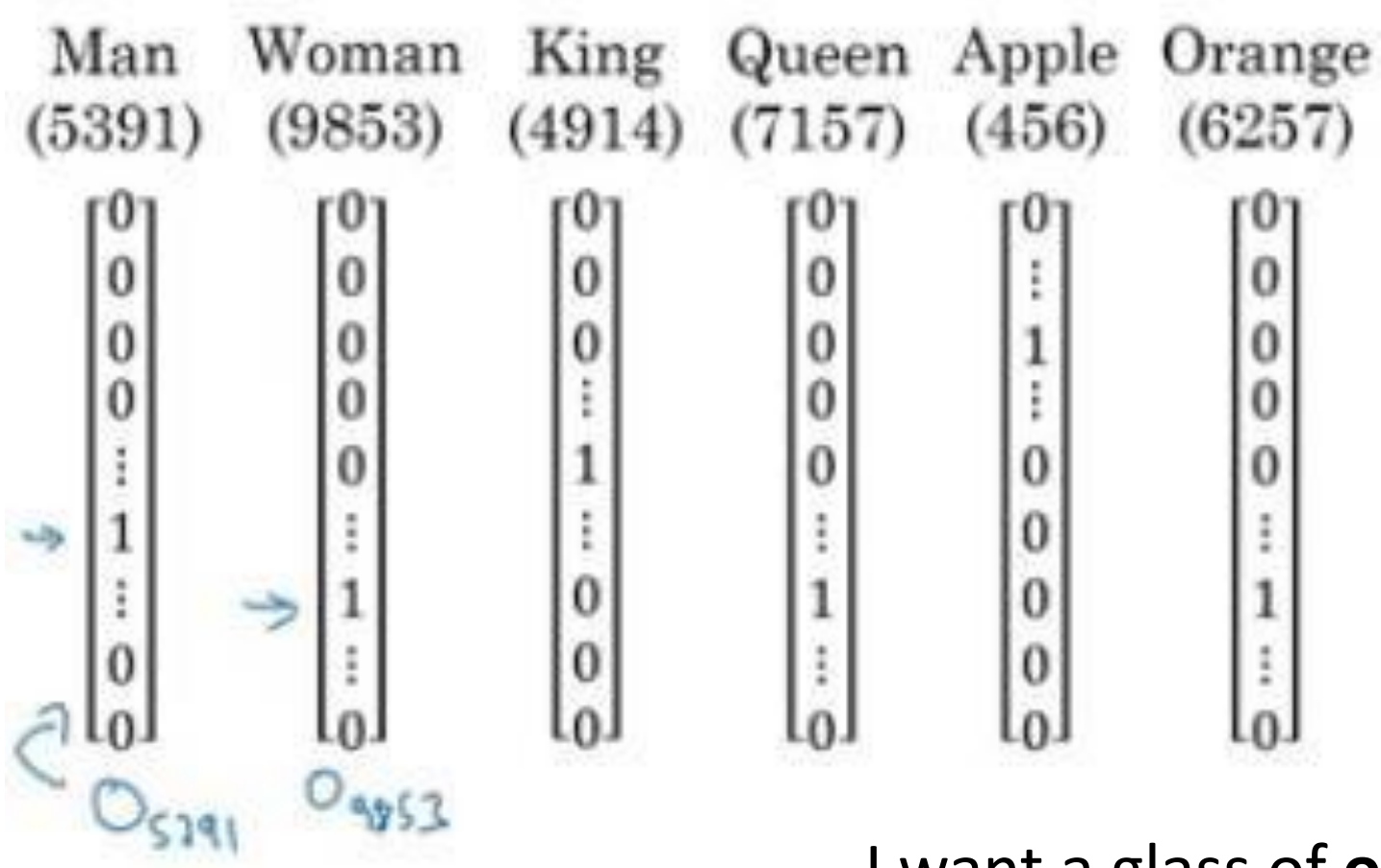
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
...						
size						
cost						
alive						
verb						

300

"I want a glass of **orange** _____"

I want a glass of **apple** _____

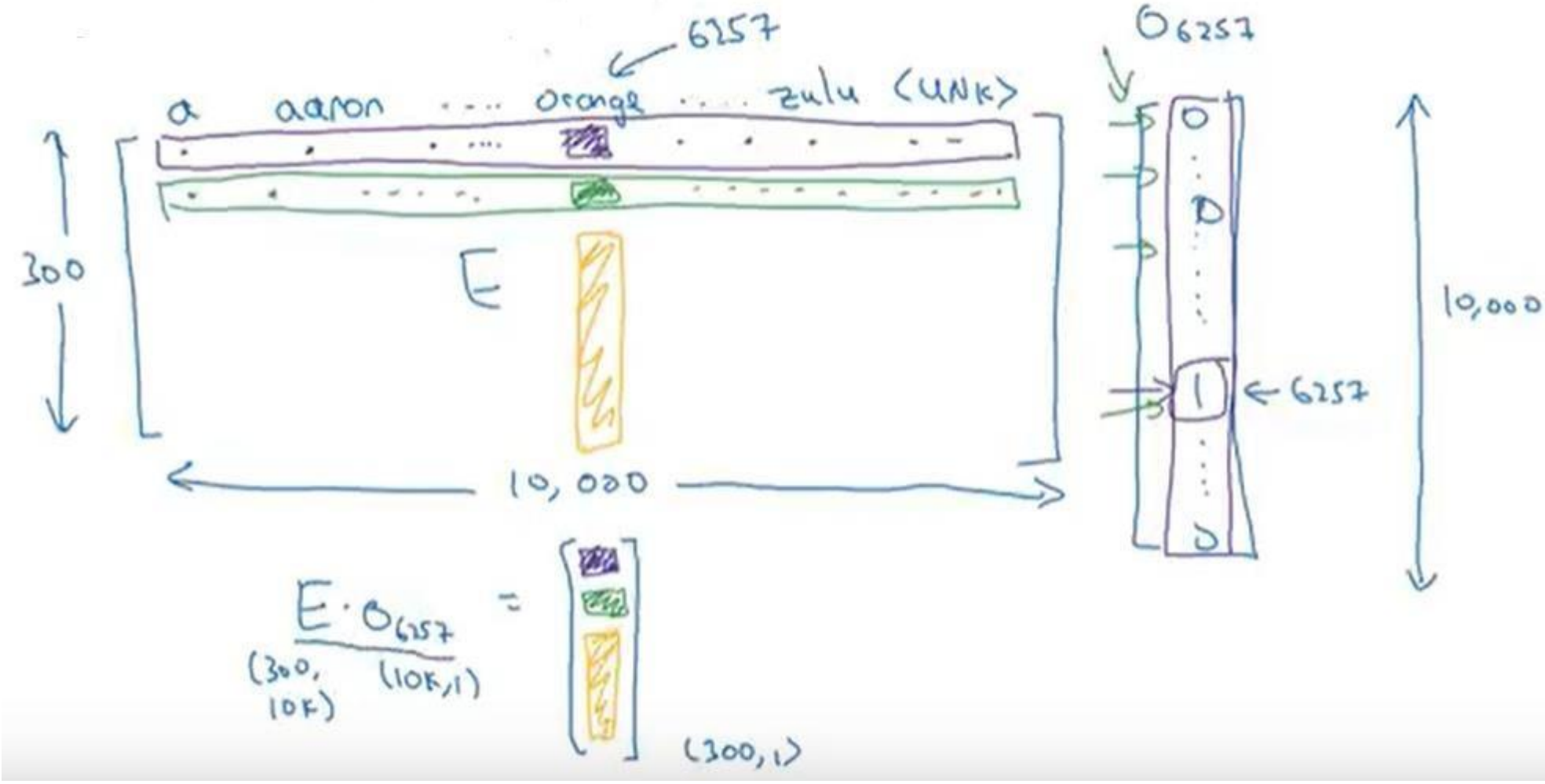
Word Representation



I want a glass of **orange** _____

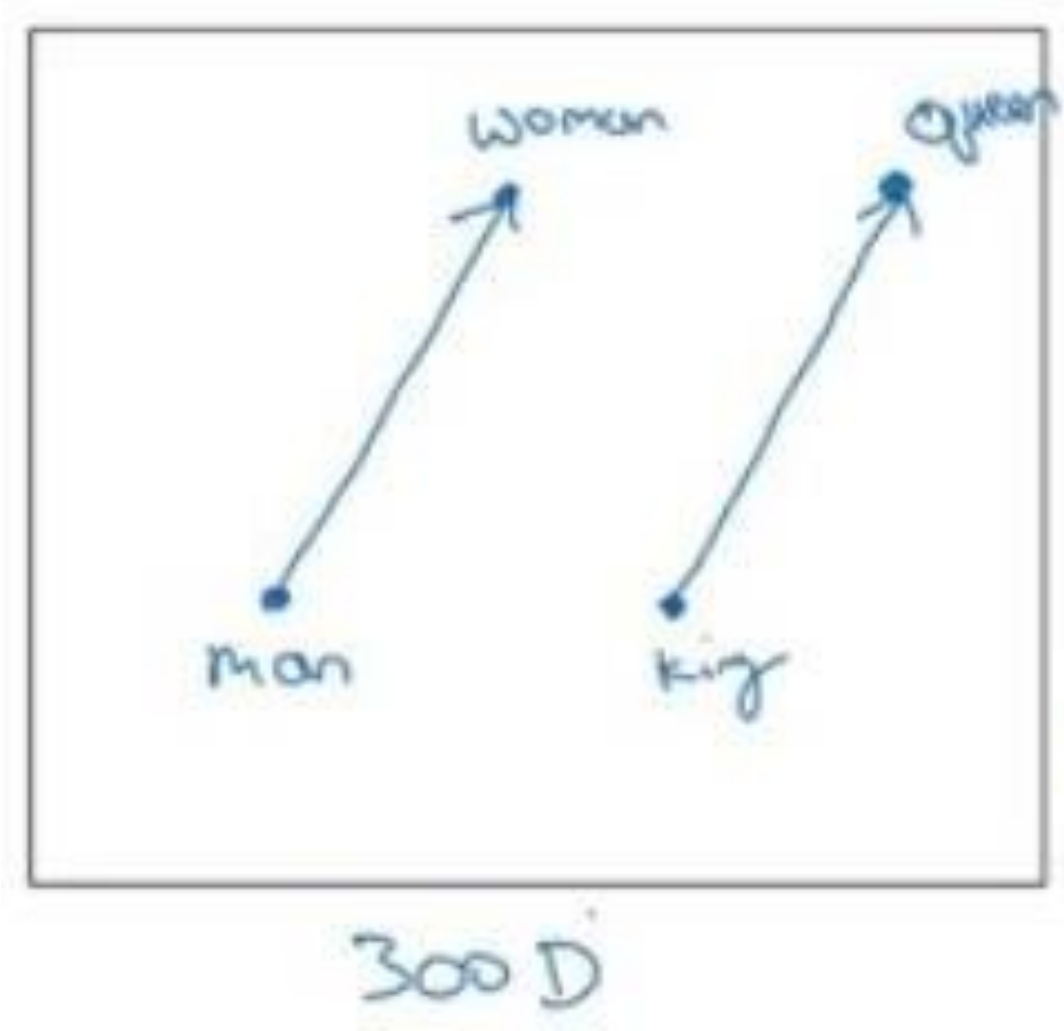
I want a glass of **apple** _____

Embedding Matrix



https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Analogies Using Word Vectors

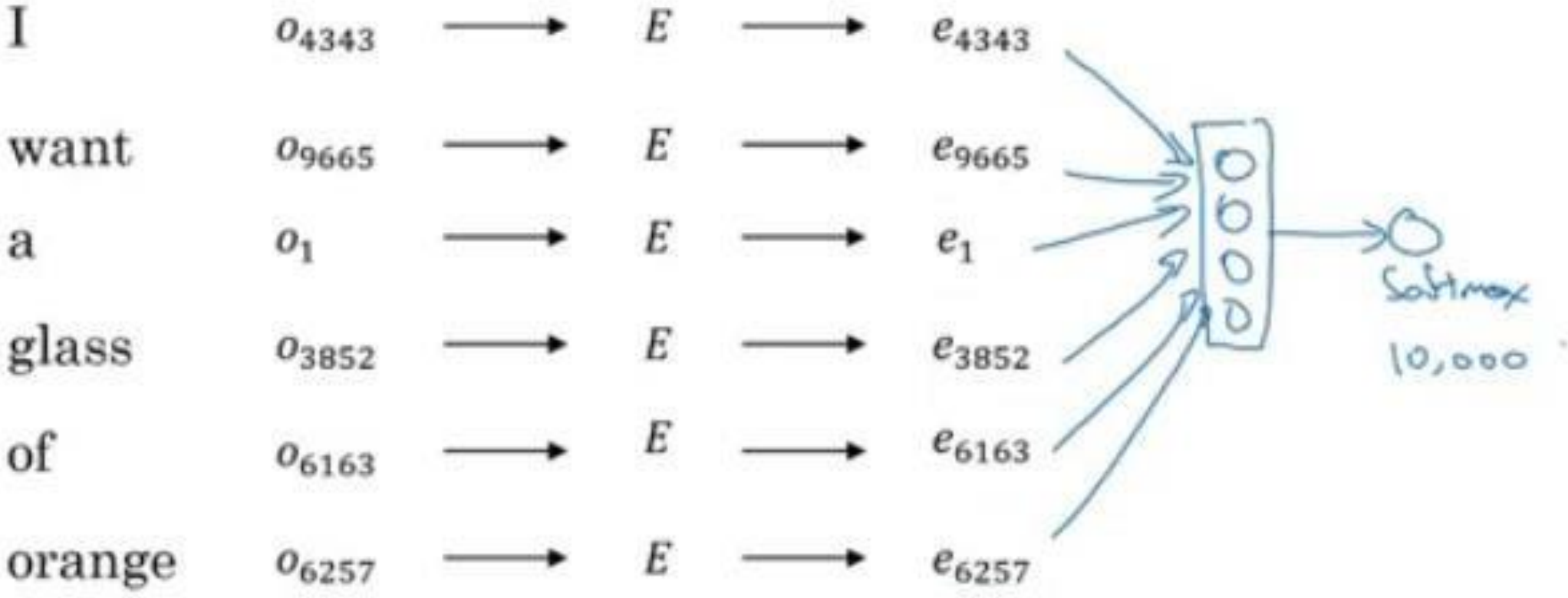


- $e_{Man} - e_{Woman} \approx e_{King} - e_{??}$
- $sim(e_w, e_{king} - e_{man} + e_{woman})$

$\arg\max_w \; sim(e_w, e_{king} - e_{man} + e_{woman})$

Neural language model

I	want	a	glass	of	orange	_____.
4343	9665	1	3852	6163	6257	



Andrew Ng

GloVe (global vectors for word representation)

- GloVe is another algorithm for learning the word embedding. It's the simplest of them.
- This is not used as much as word2vec or skip-gram models, but it has some enthusiasts because of its simplicity.
- Let's use our previous example: "**I want a glass of orange juice to go along with my cereal**".
- We will choose a context and a target from the choices we have mentioned in the previous sections.
- Then we will calculate this for every pair: X_{ct} = # times t appears in context of c
- $X_{ct} = X_{tc}$ if we choose a window pair, In GloVe they use a window which means they are equal

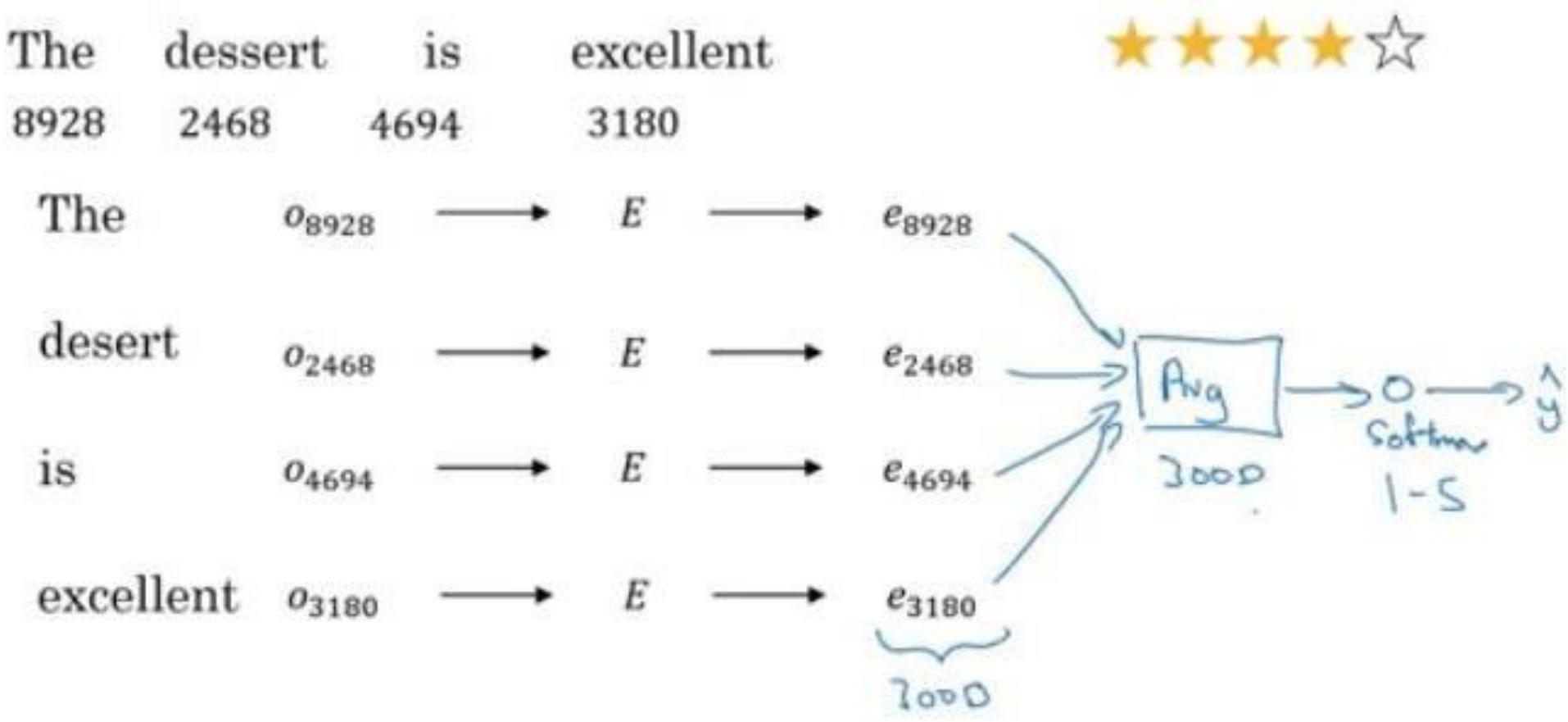
https://en.wikipedia.org/wiki/GloVe_(machine_learning)

Application using Word Embedding
Sentiment Classification

The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆

Andrew Ng

Simple sentiment classification model



Completely lacking in **good** taste, **good** service, and **good** ambience

Andrew Ng

What is sentiment analysis?

In a nutshell: extracting attitudes toward something from human language

SA aims to map *qualitative* data to a *quantitative* output(s)

This might be the most exciting NLP thing I've seen in awhile: honnibal.github.io/spaCy/. Fast, advanced NLP in Python.

=> Positive (?)

Data science can't be point and click buff.ly/1CzHbtd #DataScience #BigData

=> Negative (?)

(Or something else entirely?)

Example: Tweets

Why does averaging tweets work?

```
thanks for the great offer!! love your pizzas!!
```

```
model.most_similar(positive=positive)
```

```
[(u'great', 0.5085294246673584),
 (u'thanks', 0.4804477095603943),
 (u'your', 0.43128225207328796),
 (u'love', 0.3773646056652069),
 (u'for', 0.33436131477355957),
 (u'appreciate', 0.26042619347572327),
 (u'homie!!!!!!', 0.24800431728363037),
 (u'thank', 0.24796560406684875),
 (u'appreciated', 0.23802441358566284),
 (u'follow!', 0.23777809739112854)]
```

```
im so angry at myself
```

```
model.most_similar(positive=negative)
```

```
[(u'angry', 0.6087321043014526),
 (u'myself', 0.5472909808158875),
 (u'im', 0.4853784143924713),
 (u'so', 0.3929111957550049),
 (u'at', 0.3157612979412079),
 (u'hopeless', 0.30349820852279663),
 (u'tired', 0.2769174873828888),
 (u'sick', 0.2688058316707611),
 (u'fml', 0.2618246376514435),
 (u'cuz', 0.2534986436367035)]
```

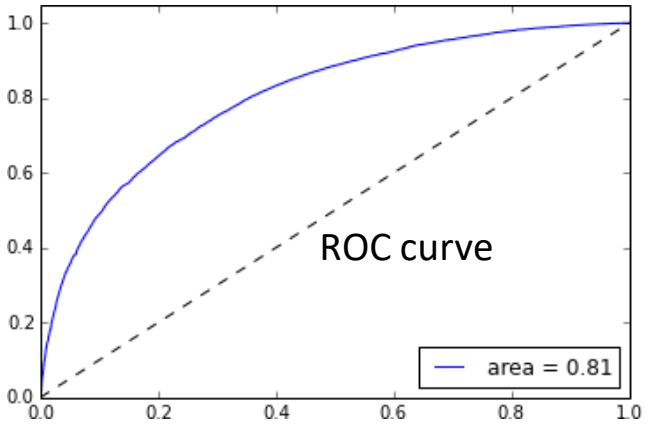
Example: Tweets

Tutorial at: http://districtdatalabs.silvrback.com/modern-methods-for-sentiment-analysis

Word2Vec trained on ~400,000 tweets gives us 73% classification accuracy

- Gensim word2vec implementation
- Sklearn Logit SGD classifier
- Improves to 77% using ANN classifier

Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**



Word2Vec Drawbacks

- Quality depends on input data, number of samples, and size of vectors (possibly long computation time!)

But Google released 3 million word vecs trained on 100 billion words!

> It can be seen that after some point, adding more dimensions or adding more training data provides diminishing improvements. So, we have to increase both vector dimensionality and the amount of the training data together.

Efficient Implementation of Word Representations in Vector Space, *T. Mikolov, K. Chen, G. Corrado, and J. Dean,* 2013.
http://arxiv.org/pdf/1301.3781.pdf