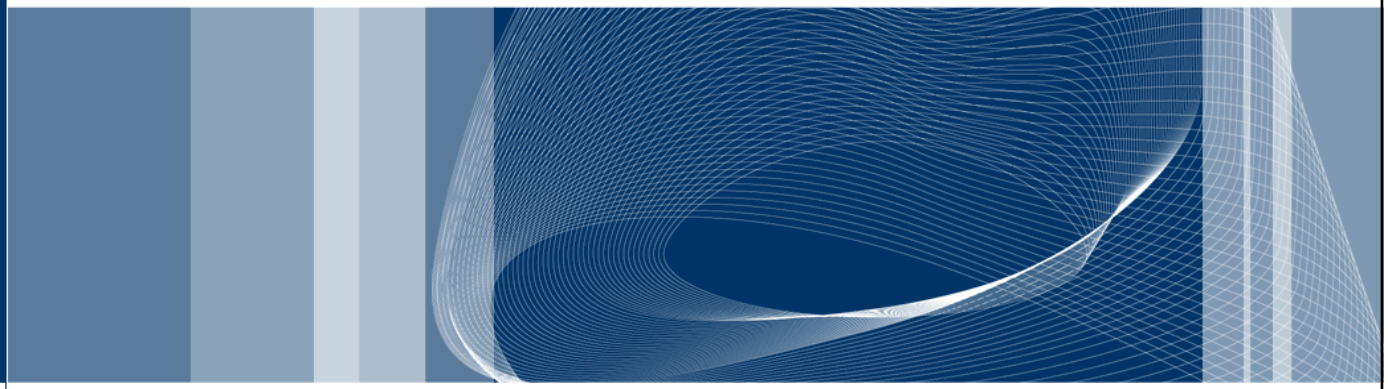
POLITECNICO DI MILANO



# Text Representation and Embeddings

## Data Mining and Text Mining

Prof. Pierluca Lanzi

How can we represent text data?

What about images?

Sound?

...

# Natural Language Processing (NLP)

## Gov. Schwarzenegger helps inaugurate pricey new bridge approach

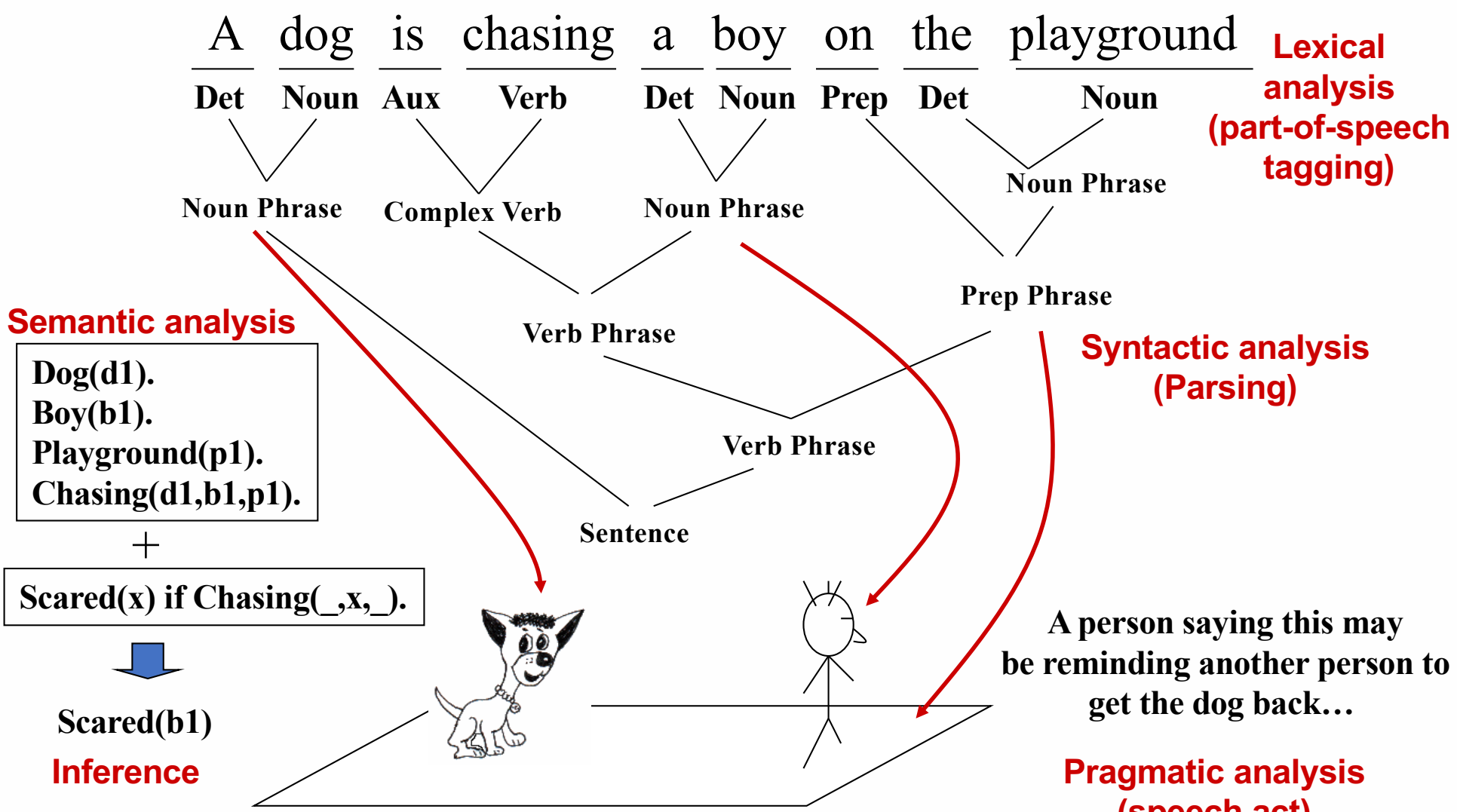
The Associated Press

Article Launched: 04/11/2008 01:40:31 PM PDT

SAN FRANCISCO—It briefly looked like a scene out of a "Terminator" movie, with Governor Arnold Schwarzenegger standing in the middle of San Francisco wielding a blow-torch in his hands. Actually, the governor was just helping to inaugurate a new approach to the San Francisco-Oakland Bay Bridge.
Caltrans thinks the new approach will make it faster for commuters to get on the bridge from the San Francisco side.
The new section of the highway is scheduled to open tomorrow morning and cost 429 million dollars to construct.

- Is this article talks about entertainment? Or politics?
- Using just the words has severe limitations

- Schwarzenegger
- Bridge
- Caltrans
- Governor
- Scene
- Terminator




(Taken from ChengXiang Zhai, CS 397cxz – Fall 2003)

# Natural Language Processing is Difficult!



- Natural language is designed for efficient human communication
- As a result,
  - We omit a lot of common-sense knowledge, which we assume the hearer/reader possesses.
  - We keep a lot of ambiguities, which we assume the hearer/reader knows how to resolve.
- This makes every step of natural language processing very difficult
  - Ambiguity
  - Common sense reasoning

# What the Difficulties?



- Word-level Ambiguity
  - "design" can be a verb or a noun
  - "root" has multiple meaning
- Syntactic Ambiguity
  - A man saw a boy with a telescope
- Presupposition
  - "He has quit smoking" implies he smoked
- Text Mining NLP Approach
  - Locate promising fragments using fast methods
  - Only apply slow NLP techniques to promising fragments

# What the Open Problems?



- 100% POS tagging
  - "he turned off the highway" vs "he turned off the light"
- General complete parsing
  - "a man saw a boy with a telescope"
- Precise deep semantic analysis
- Robust and general NLP methods tend to be shallow while deep understanding does not scale up easily


JEOPARDY!
JEOPARDY!

# Information Retrieval

# Information Retrieval Systems



- Information retrieval deals with the problem of locating relevant documents with respect to the user input or preference

- Typical systems
  - Online library catalogs
  - Online document management systems

- Typical issues
  - Management of unstructured documents
  - Approximate search
  - Relevance

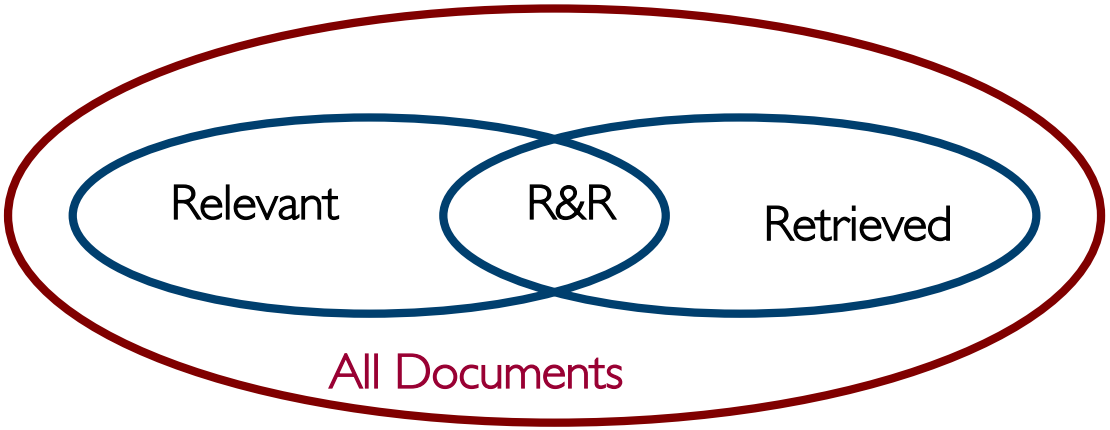
# Two Modes of Text Access



- Pull Mode (search engines)
  - Users take initiative
  - Ad hoc information need
- Push Mode (recommender systems)
  - Systems take initiative
  - Stable information need or system has good knowledge about a user's need

# Text Retrieval Methods

- Document Selection (keyword-based retrieval)
  - Query defines a set of requisites
  - Only the documents that satisfy the query are returned
  - A typical approach is the Boolean Retrieval Model
- Document Ranking (similarity-based retrieval)
  - Documents are ranked on the basis of their relevance with respect to the user query
  - For each document a "degree of relevance" is computed with respect to the query
  - A typical approach is the Vector Space Model




$$\text{precision} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{\{\text{Retrieved}\}}$$

$$\text{recall} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{\{\text{Relevant}\}}$$

$$\text{F-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

# Document Similarity

# Vector Space Model

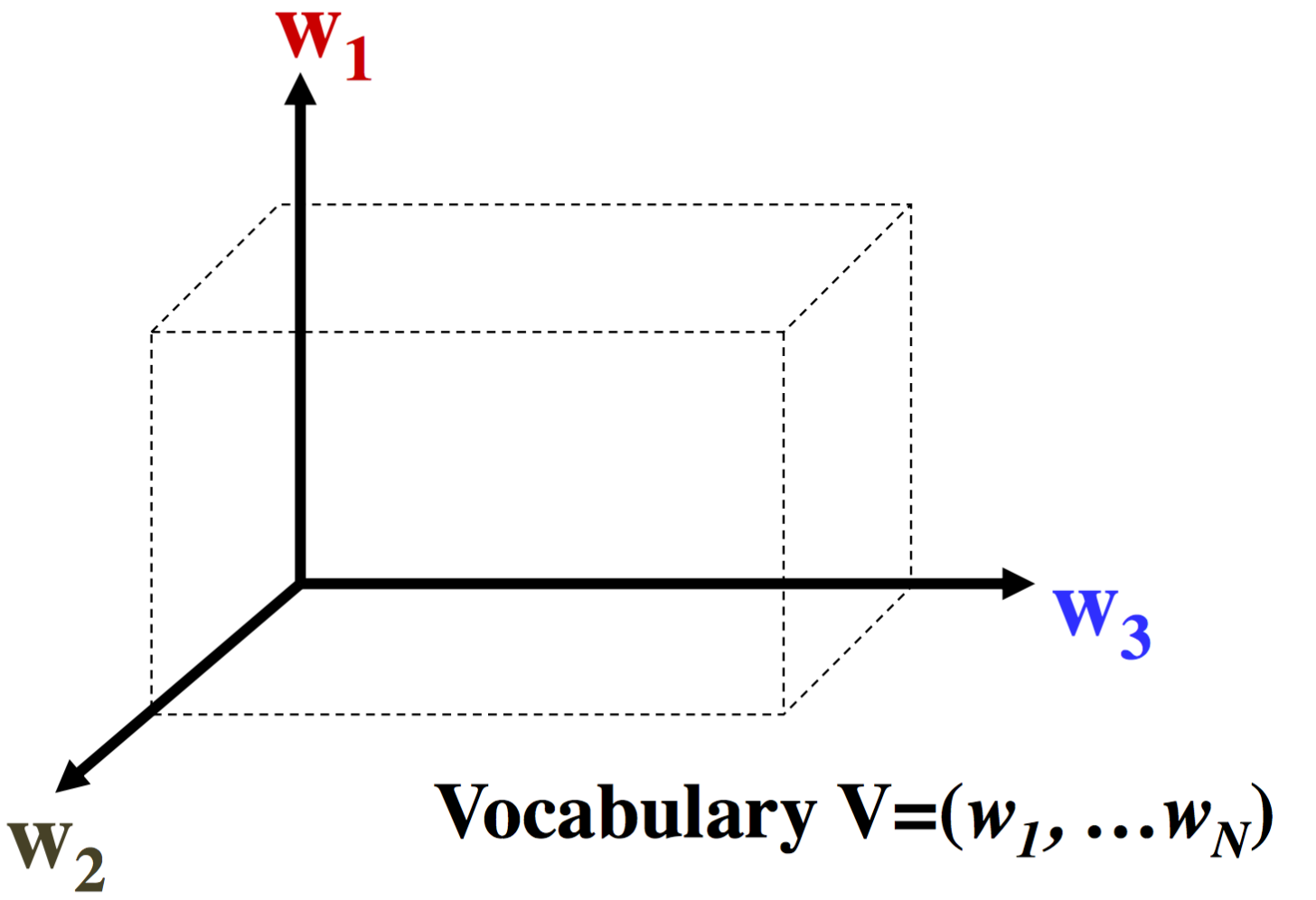


- A document is vectors in high-dimensional space corresponding to all the keywords
- Relevance is measured with an appropriate similarity measure defined over the vector space
- Issues
  - How to select keywords to capture "basic concepts" ?
  - How to assign weights to each term?
  - How to measure the similarity?

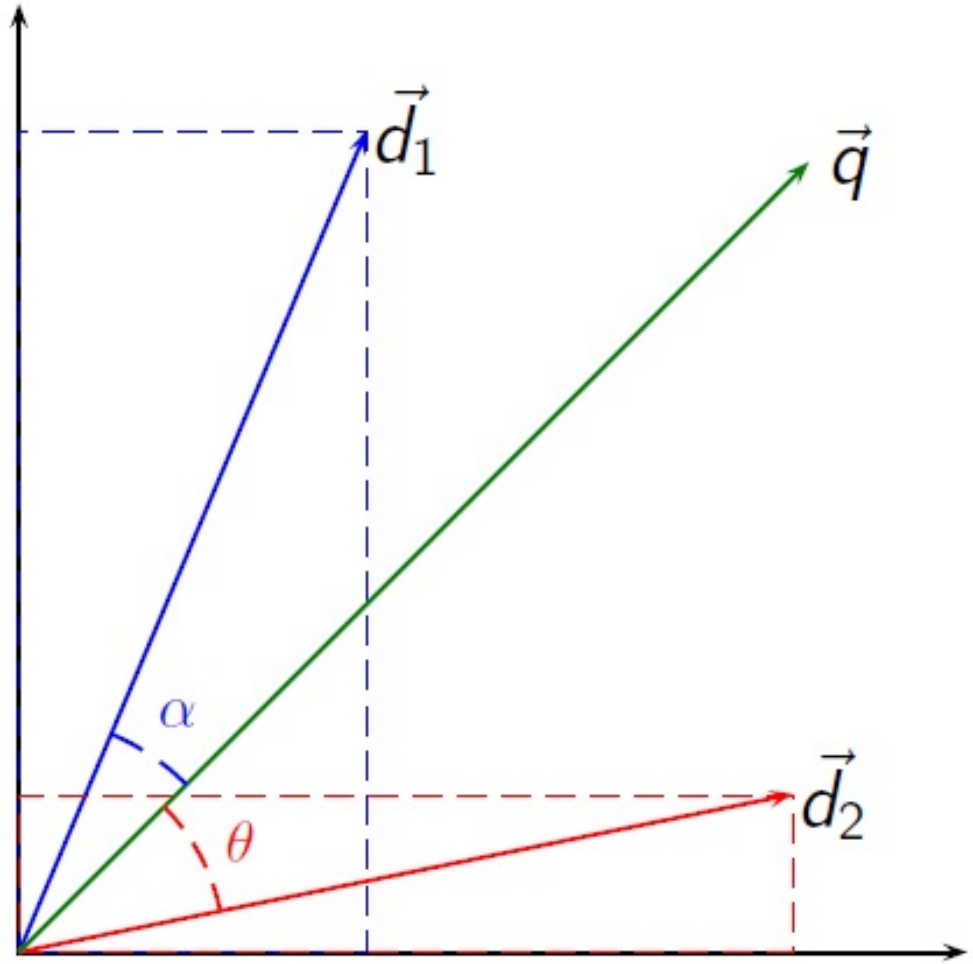
# Bag-of-Words Representation



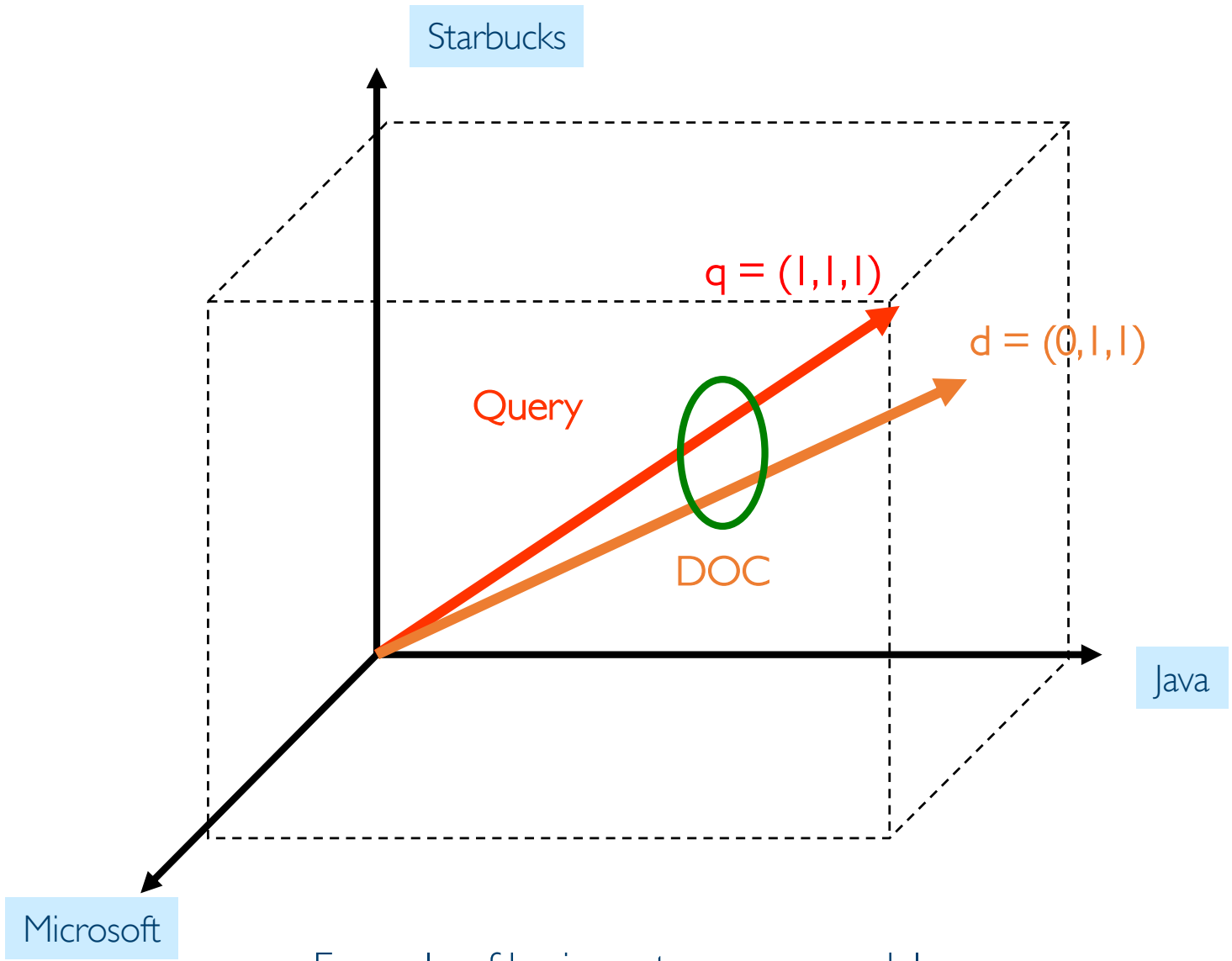
- Text is represented as a table
  - Columns (features/variables) identify vocabulary words
  - Rows (examples/data points) identify documents
- The values can represent
  - The presence of the word
  - The frequency of the word inside the document
  - …
- It is a sparse representation since a document usually contains much fewer words than the entire vocabulary so that most values in the vector are zeros
- Vector space model
  - Documents thus become vectors in the feature space




Bag-of-Words Representation




Vector Space Model
https://en.wikipedia.org/wiki/Vector_space_model#/media/File:Vector_space_model.jpg




Example of basic vector space model

# Representations of Documents

- Given two documents $q = q_1, \ldots, q_m$ and $d = d_1, \ldots, d_n$ where $q_i$ and $d_i$ represent words (bag of words model)
- We will use a similarity function sim(q, d), which returns a real value, to rank most similar documents on top of least similar ones
- There are two key challenges
  - How can we represent documents?
  - What similarity function should we use? (Typically we will use cosine similarity)

# Document Representation: Basic Vector Space Model



- Let's begin with a very simple representation ($d_i$ and $q_i$ represent the presence or absence of word $w_i$) and a simple similarity function f:

$$f(q,d) = q_1 d_1 + \cdots + q_N d_N$$

- Given the document q="news about presidential campaign" how would you rank the following documents?
  - d1 = "… news about …
  - d2 = "… news about … food campaign …"
  - d3 = "… news of … presidential campaign …"
  - d4 = "news of presidential campaign … presidential candidate …"
  - d5 = "… news of organic food campaign … campaign … campaign … campaign …"

| | | Ideal Ranking? |
|---|---|---|
| d1 | … **news about** … | d4 + |
| d2 | … **news about** organic food **campaign**… | d3 + |
| d3 | … **news** of **presidential campaign** … | |
| d4 | … **news** of **presidential campaign** …<br>… **presidential** candidate … | d1 - |
| d5 | … **news** of organic food **campaign**…<br>**campaign**…**campaign**…**campaign**… | d2 -<br>d5 - |

What might be the ideal ranking?

# q="news about presidential campaign"

d1 … **news about** …

d3 … **news** of **presidential campaign** …

vocabulary

V= {news, about, presidential, campaign, food …. }

**q= (1, 1, 1, 1, 0, …)**

**d1= (1, 1, 0, 0, 0, …)**

**f(q,d1)= 1*1+1*1 +1* 0+1*0+0*0 + …= 2**

**d3= (1, 0, 1, 1, 0, …)**

**f(q,d3)= 1*1+1*0 +1*1 +1*1+0*0+ … = 3**

# q="news about presidential campaign"

| | | |
|---|---|---|
| d1 | … **news about** … | f(q,d1)=2 |
| d2 | … **news about** organic food **campaign**… | f(q,d2)=3 |
| d3 | … **news** of **presidential campaign** … | f(q,d3)=3 |
| d4 | … **news** of **presidential campaign** …<br>… **presidential** candidate … | f(q,d4)=3 |
| d5 | … **news** of organic food **campaign**…<br>**campaign**…**campaign**…**campaign**… | f(q,d5)=2 |

This basic representation ranks the documents based on the number of distinct words that the documents share

q="news about presidential campaign"

| | | |
|---|---|---|
| d2 | … **news about** organic food **campaign**… | **f(q,d2)=3** |
| d3 | … **news** of **presidential campaign** … | **f(q,d3)=3** |
| d4 | … **news** of **presidential campaign** …<br>… **presidential** candidate … | **f(q,d4)=3** |

Matching "presidential" more times deserves more credit

Matching "presidential" is more important than matching "about"

# Document Representation: Term Frequency (TF)



- To take into account the number of times a word appears in the document $d_i$ and $q_i$ now represent the number of times word $w_i$ appears in the document.

- Note that, this is a simplification for the discussion. In fact, we will actually use the relative frequency of the words inside the documents when applying it.

# q="news about presidential campaign"

d2 … **news about** organic food **campaign**… f(q,d2)=3

**q= (1, 1, 1, 1, 0, …)**
**d2= (1, 1, 0, 1, 1, …)**

d3 … **news** of **presidential campaign** … f(q,d3)=3

**q= (1, 1, 1, 1, 0, …)**
**d3= (1, 0, 1, 1, 0, …)**

d4 … **news** of **presidential campaign** …
… **presidential** candidate … **f(q,d4)=4!**

**q= (1, 1, 1, 1, 0, …)**
**d4= (1, 0, 2, 1, 0, …)**

# "Presidential" vs "About"



- Matching "presidential" is more relevant than matching "about"
- However in our example the two words weight the same
- Moreover, "about" might weight much more than "presidential" since it appears more frequently in (all) documents.
- We need to penalize the words that are less informative (like "about") and the ones that are more informative and surprising to be found in a document (like "presidential")

# q="news about presidential campaign"

d2 … **news about** organic food **campaign**…

d3 … **news** of **presidential campaign** …

V= {news, about, presidential, campaign, food …. }

q= (1, 1, 1, 1, 0, …)
d2= (1, 1, 0, 1, 1, …)

f(q,d2)<3

q= (1, 1, 1, 1, 0, …)
d3= (1, 0, 1, 1, 0, …)

f(q,d3)>3

"presidential" and "about" weight the same in the document but the former is more relevant and surprising to be found in a document.

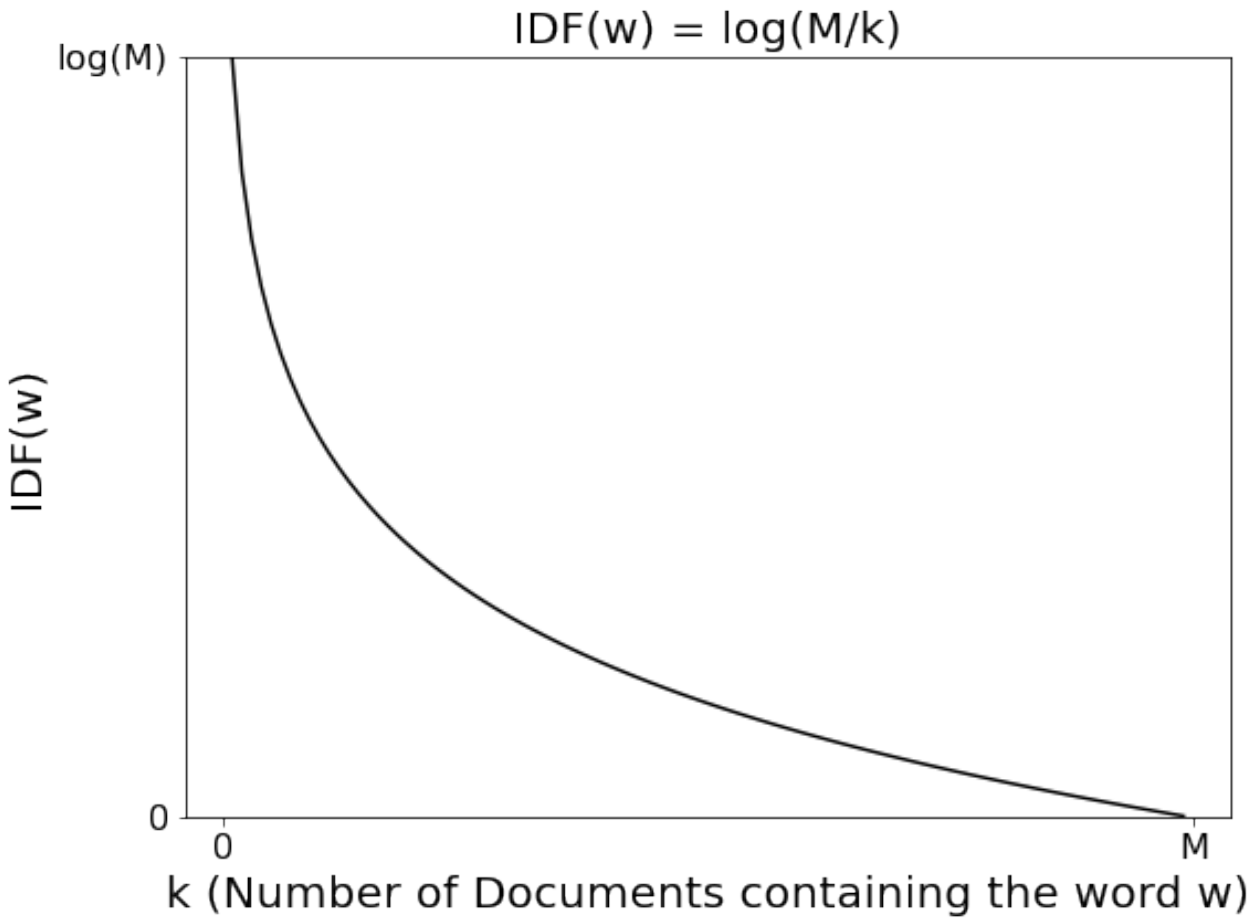
# Document Representation: Inverse Document Frequency (IDF)



- It measures how much information the word provides (how much it is surprising to have it in the document).
- Penalizes the words that frequently occur in many documents,

$$IDF(w) = \log \frac{M}{k}$$

- When w might is not in the corpus, the formula leads to a division-by-zero so the adjusted version with (k+1) at the denominator is used.




Inverse Document Frequency: M is the number of document in the corpus,
k is the number of documents containing word w.

# Combining Term Frequency (TF) with Inverse Document Frequency Ranking

$$d_i = c(w_i, d) \log \frac{M}{\mathrm{df}(w)}$$

$$q_i = c(w_i, q) \log \frac{M}{\mathrm{df}(w)}$$

$$f(q, d) = \sum_{i=1}^{N} q_i d_i$$

$$= \sum_{w \in q \cap d} c(w, q) \log \frac{M}{\mathrm{df(w)}} c(w, d) \log \frac{M}{\mathrm{df}(w)}$$

# q="news about presidential campaign"

d2 … **news about** organic food **campaign**…

d3 … **news** of **presidential campaign** …

V= {news, about, presidential, campaign, food …. }

**IDF(W)= 1.5 1.0 2.5 3.1 1.8**

q = ( 1x1.5, 1.0 x 1.0, 1 x 2.5, 1 x 3.1, 0)
d2 = ( 1x1.5, 1.0 x 1.0, 0, 1 x 3.1, 1 * 1.8)

q = ( 1x1.5, 1.0 x 1.0, 1 x 2.5, 1 x 3.1, 0)
d3 = ( 1x1.5, 0, 1 x 2.5, 1 x 3.1, 0)

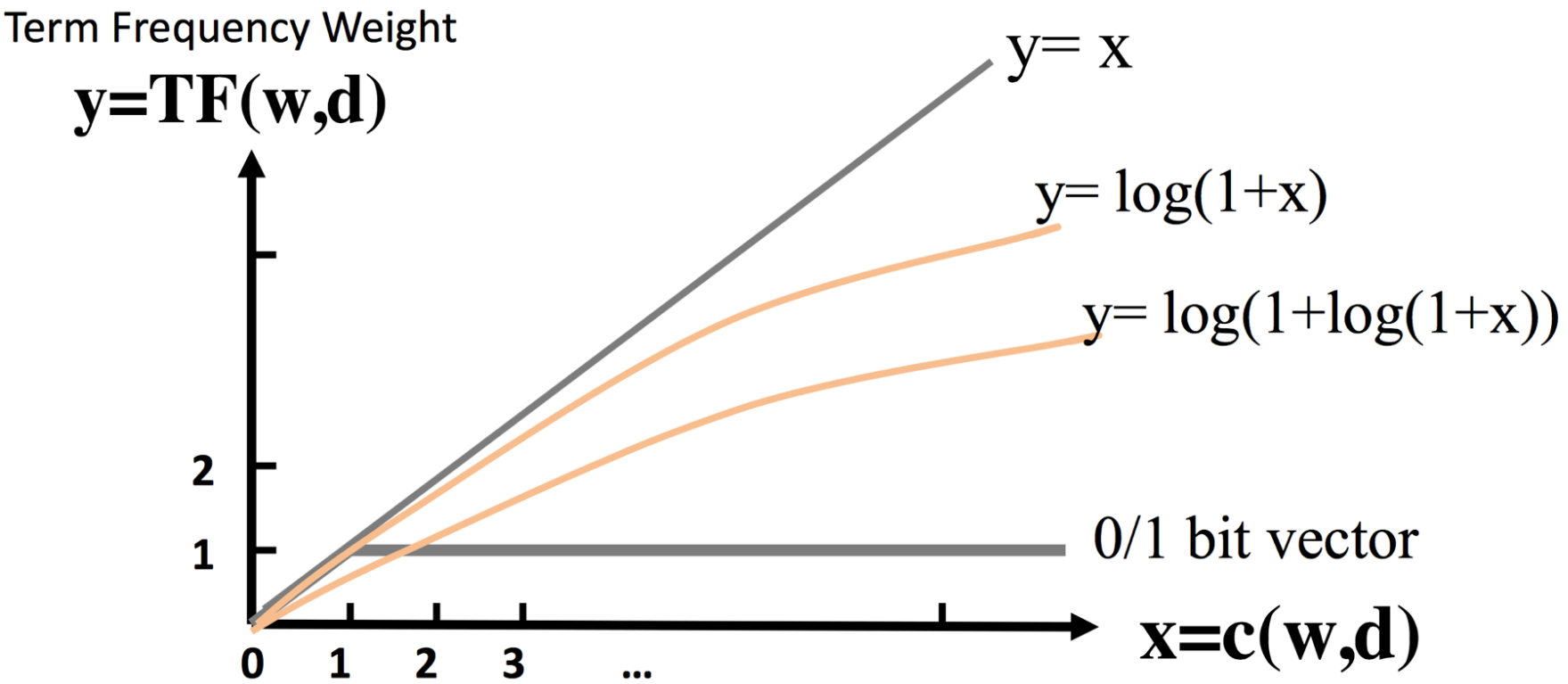
f(q,d2)=12.86 < f(q,d3)=18.11

Suppose we computed the IDF for the words in the vocabulary as IDF("news")=1.5, IDF("about")=1.0, IDF("presidential")=2.5, IDF("campaign")=3.1, IDF("food")=1.8

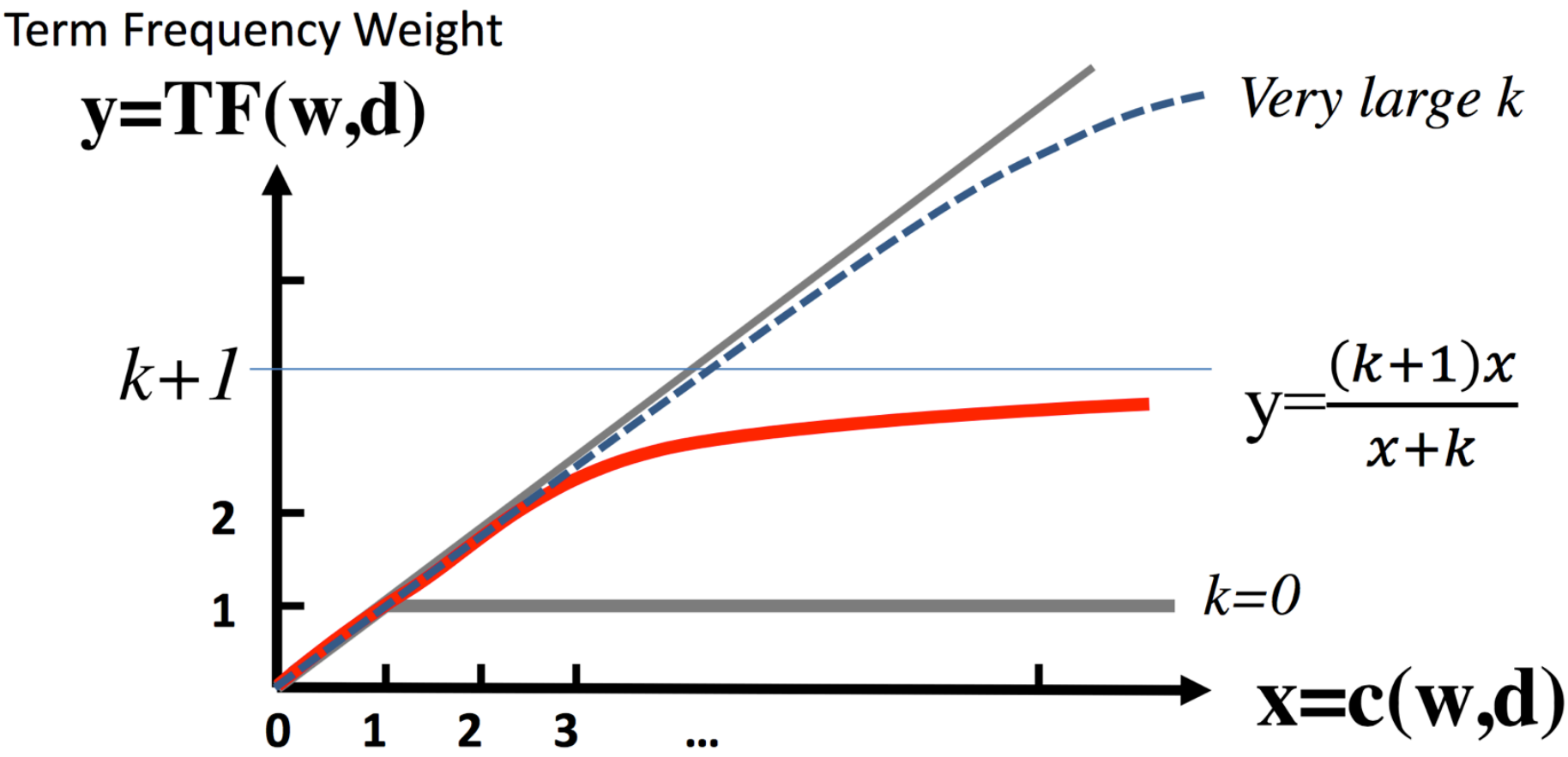
# q="news about presidential campaign"

| | | |
|---|---|---|
| d1 | … **news about** … | f(q,d1) = 3.25 |
| d2 | … **news about** organic food **campaign**… | f(q,d2) = 12.86 |
| d3 | … **news** of **presidential campaign** … | f(q,d3) = 18.11 |
| d4 | … **news** of **presidential campaign** …<br>… **presidential** candidate … | f(q,d4) = 24.36 |
| d5 | … **news** of organic food **campaign**…<br>**campaign**…**campaign**…**campaign**… | f(q,d5) = 40.69! |

Note however that word with high frequency in a document might return unexpected results.




We can limit the effect of term frequency.




BM25 Term Frequency Transformation

# What about Document Length?

d4

… **news** of **presidential campaign** …
… **presidential** candidate …

**100 words**

**d6 > d4?**

d6

… **campaign**…..... **campaign**..... **5000 words** .....
……
.....**news**.....
……
…… **news**…..
……
…… ……
…… **presidential** ……. **presidential**……

# Document Length Normalization

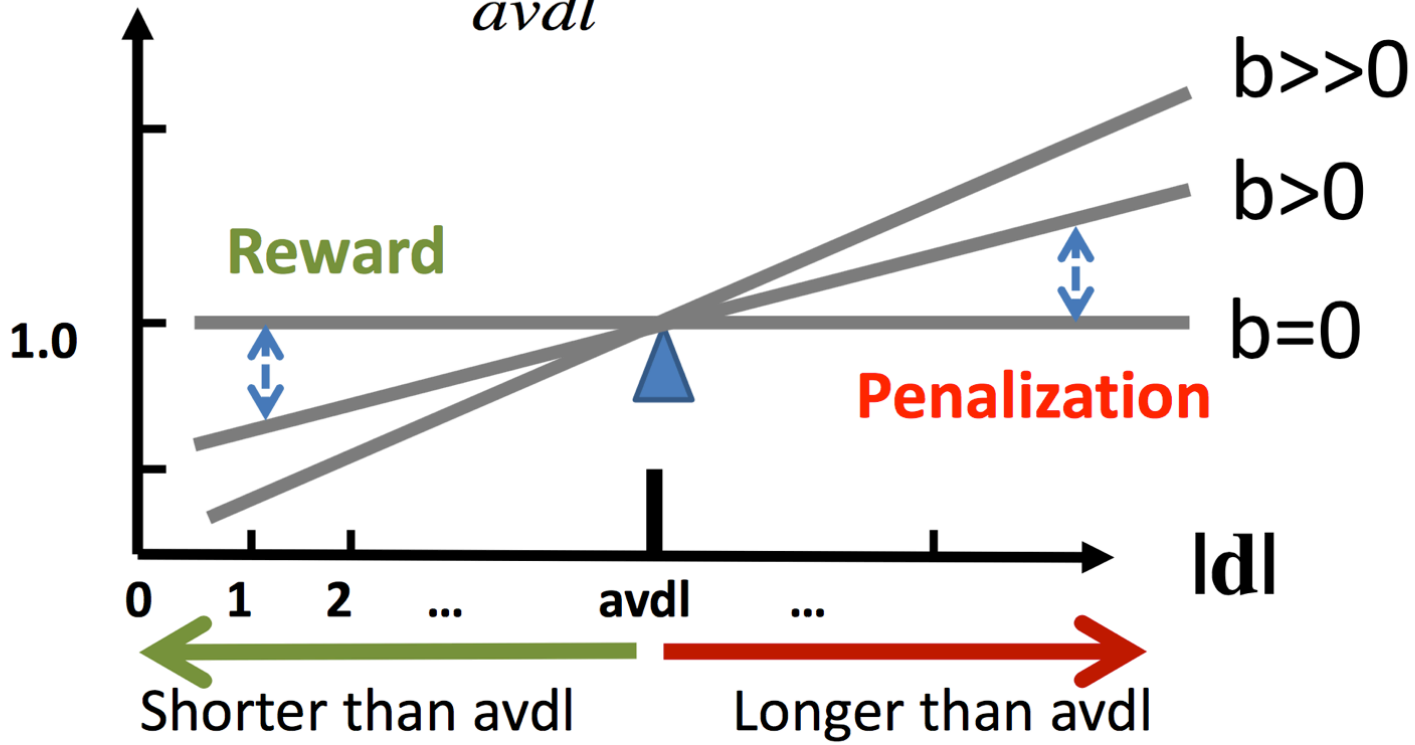


- Penalize a long doc with a doc length normalizer
  - Long doc has a better chance to match any query
  - Need to avoid over-penalization
- A document is long because
  - it uses more words, then it should be more penalized
  - it has more contents, then it should be penalized less
- Pivoted length normalizer
  - It uses average doc length as "pivot"
  - The normalizer is 1 if the doc length (|d|) is equal to the average doc length (avdl)

# Pivot Length Normalizer

$$normalizer = 1 - b + b\frac{|d|}{avdl}$$

$$b \in [0,1]$$



Pivoted Length Normalization VSM [Singhal et al 96]

$$d_i = \frac{\ln(1+\ln(1+c(w_i,d)))}{1-b+b\frac{|d|}{avdl}} \log \frac{M}{\mathrm{df}(w)}$$

BM25/Okapi [Robertson & Walker 94]

$$d_i = \frac{(k+1)c(w_i,d)}{c(w_i,d)+k\left(1-b+b\frac{|d|}{avdl}\right)} \log \frac{M}{\mathrm{df}(w)}$$

# Preprocessing Text Data

(From Text to Numerical Vectors)

# Text Data Preprocessing: Tokenization



- Preliminary Steps
  - Remove non-content related information, e.g. <HTML> tags
  - Lowercase the text (can lose information: e.g. 'WHO' vs 'who')
  - Remove punctuation
- Stop words elimination
  - Stop words are elements that are considered uninteresting with respect to the retrieval and thus are eliminated
  - For instance, "a", "the", "always", "along"
- Word stemming/Lemmatization
  - Different words that share a common prefix are simplified and replaced by their common prefix
  - For instance, "computer", "computing", "computerize" are replaced by "comput"; "fishing" with "fish", etc.

# Word Embeddings

- Bag of Words Model
  - Words are represented by one position in a huge vector (representing the vocabulary of words)
  - Context information is not used
- Word Embeddings
  - Dense representations of words rather than documents
  - Stores each word in as a point in continuous space
  - A word is represented by a vector of fixed number of dimensions (between 25 or 1000)
  - Generated from a huge corpus using supervised methods
  - Dimensions are basically projections along different axes

Word embeddings revolutionized NLP tasks

Performance improvements on
just about every task

"A word is known by
the company it keeps"

# Why Word Embeddings?

- Your task is to fill in the blank in the sentence:

  "Sure Sally, let's have a skype call at 3pm _____ the 3rd of June."

- What could fit?
  - Days of the week (like Monday, Tuesday, Wednesday, …)
  - Timezones (GMT, CET, EST, AEST, …)
  - Others on, by, before, around, near, …

- Embeddings are supervised models
  - Trained to predict missing word based on surrounding context
  - Note that not many words fit and those that do are semantically related to each other

# Word Embeddings as a Multi-Class Classification Problem



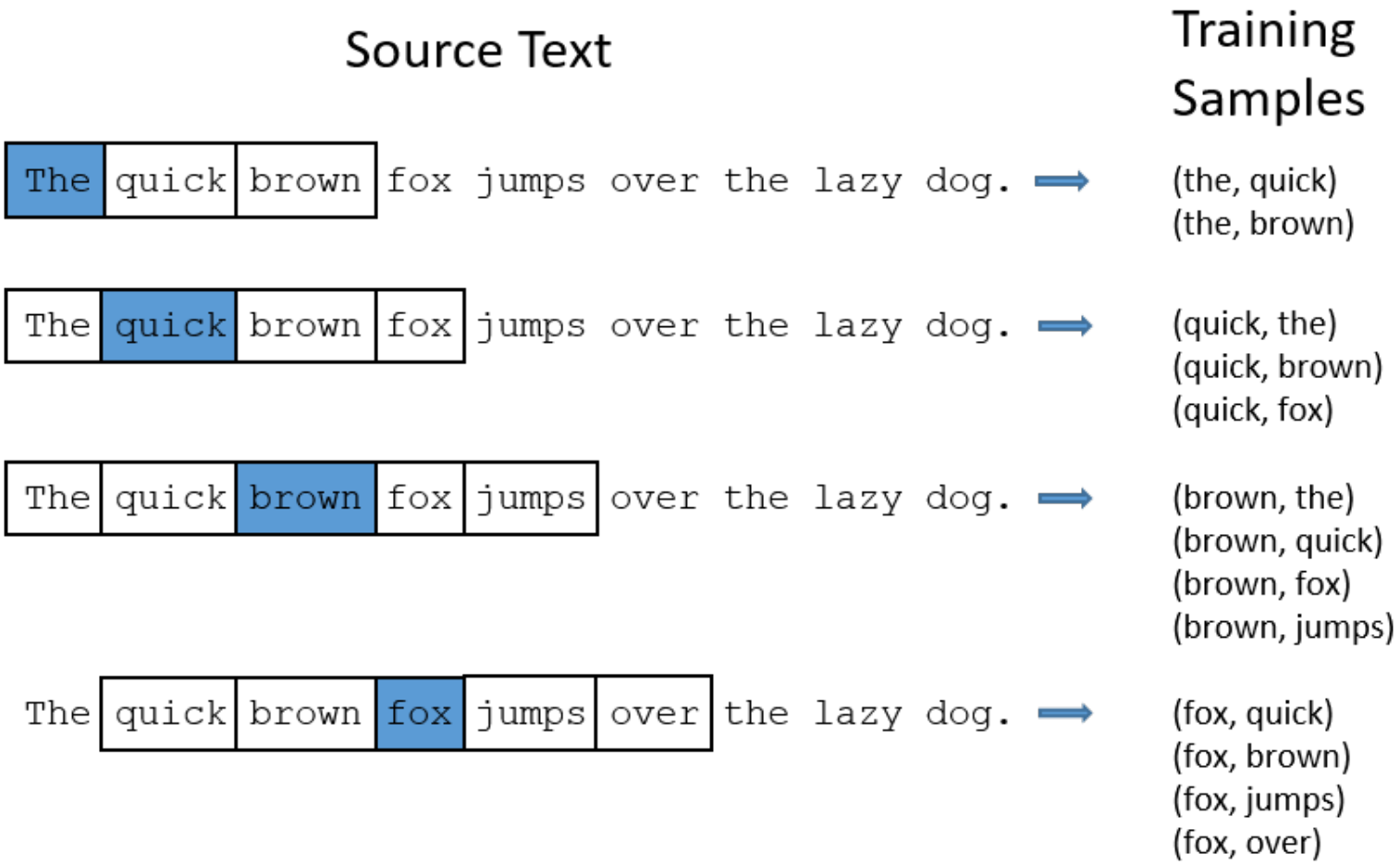
- To predicting missing word,
  - The input variables are the words in current context
  - The target is the word missing word from the sequence
- The problem has a massive parameter space.
- Consider for example a vocabulary of 20000 words and a context of 5 words, the parameter space is 20000x5x20000 that is |vocabulary| x |context| x |vocabulary|

# Word2Vec

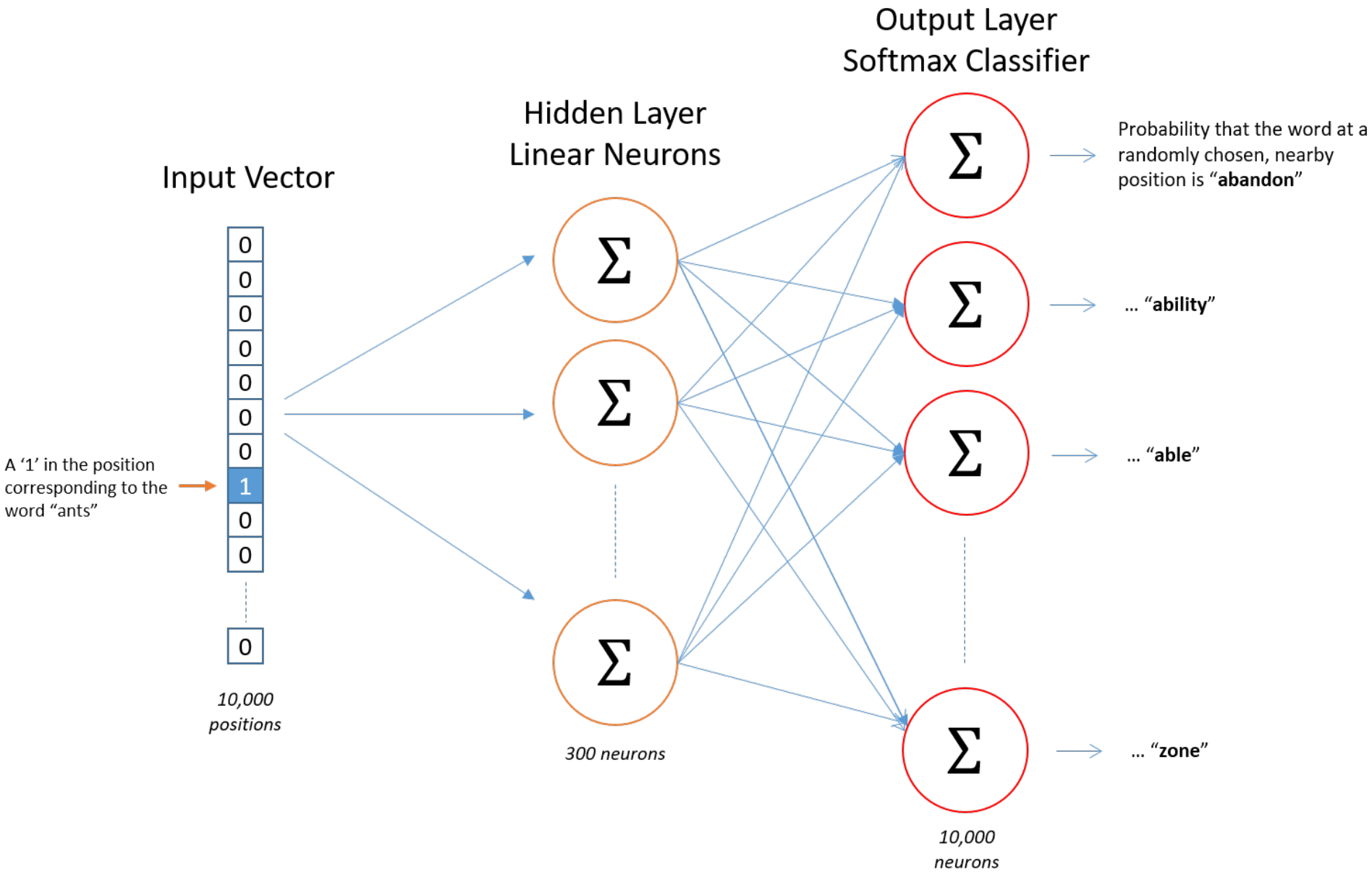
# Word2Vec (Google)



- It employs a skip gram neural network architecture
- It trains a simple neural network with a single hidden layer to, given a specific word in the middle of a sentence (the input word), predict the probability for every word in our vocabulary of being the "nearby word" that we chose.
- Note that we are not interested in the entire model and we are not going to use it for prediction, instead we are going to use the hidden layer as a representation of the input word
- The output vector is a probability distribution

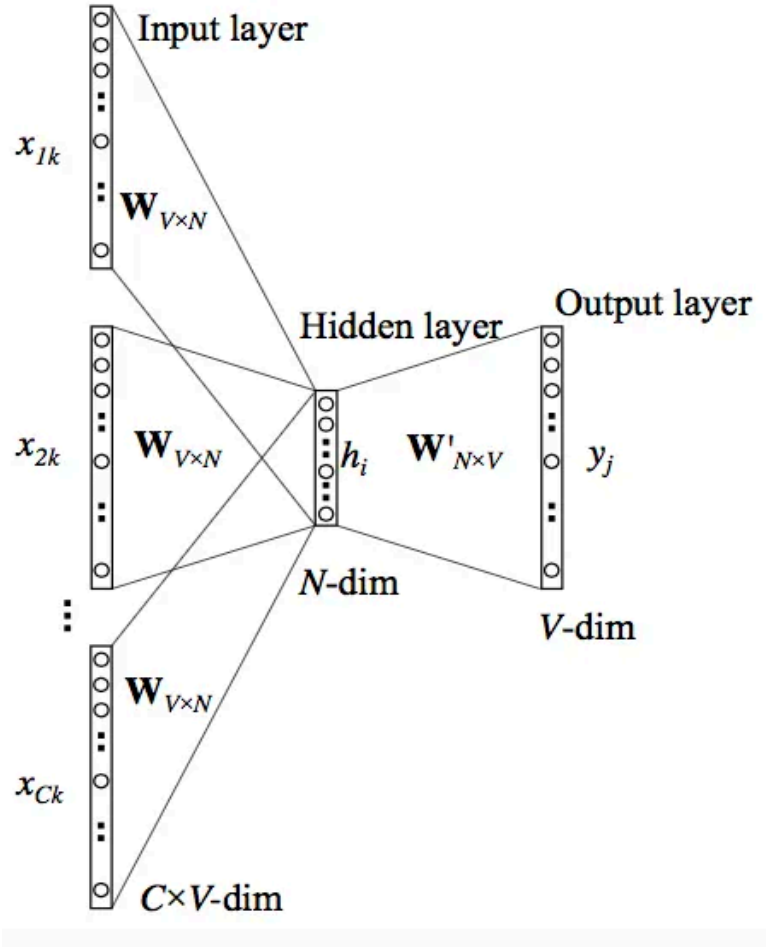



Training Examples for Word2Vec
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

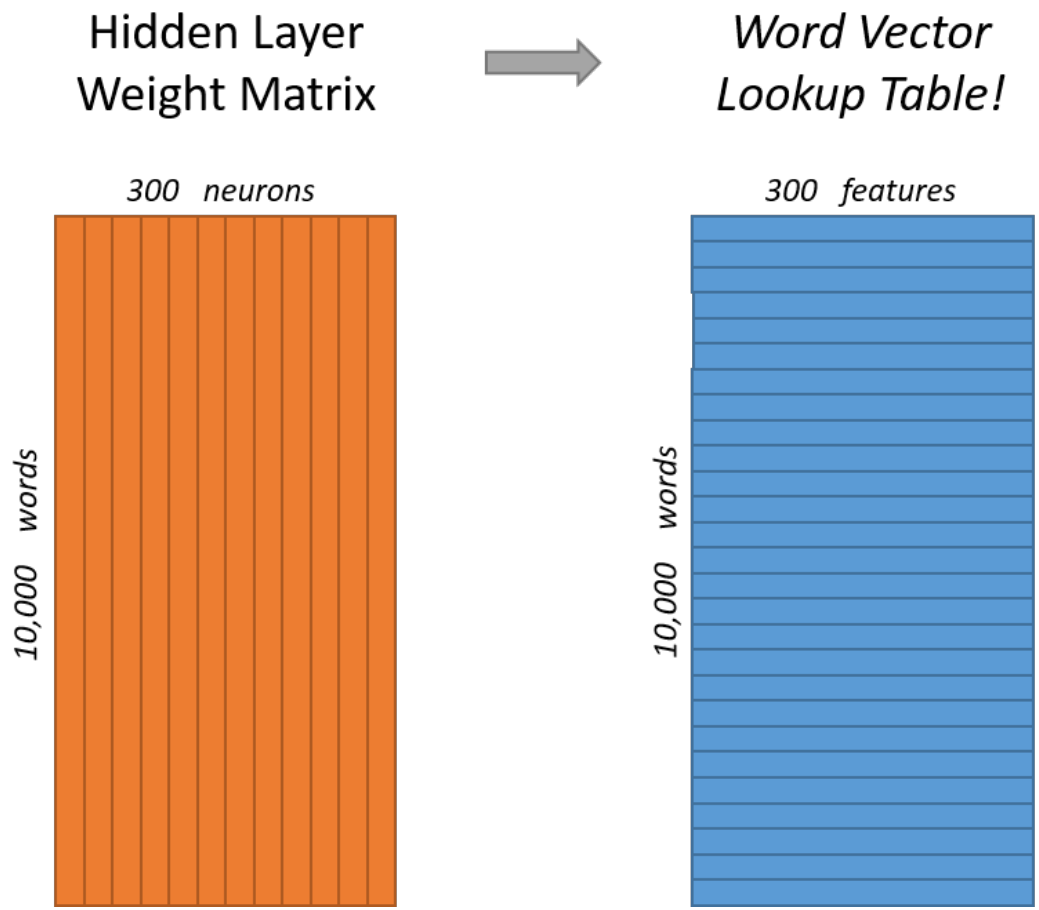



Word2Vec Skip Gram Architecture

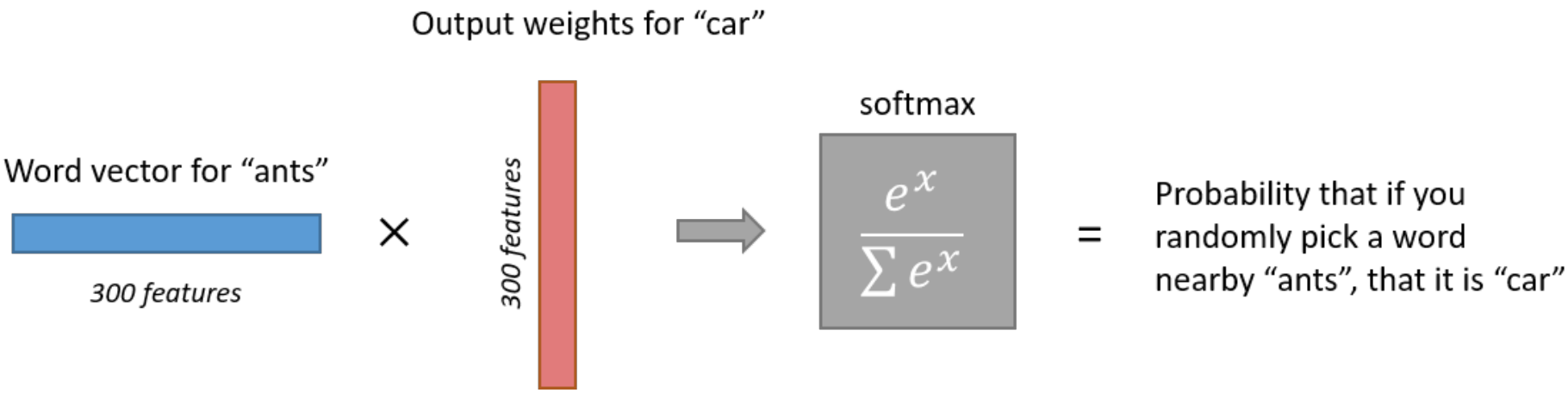
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/




http://www.stokastik.in/understanding-word-vectors-and-word2vec/




Word2Vec representation of word embeddings
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/




Word2Vec Skip Gram Architecture
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

# Word2Vec



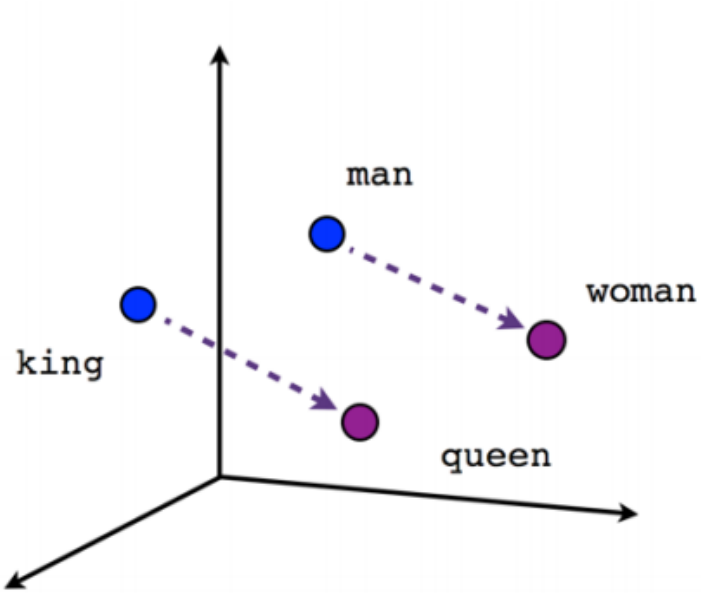
- Word2Vec developed in 2013 by Mikolov et al.
  - Following early work by Bengio et al. in 2003
  - Later GloVe in 2014 by Penington et al.

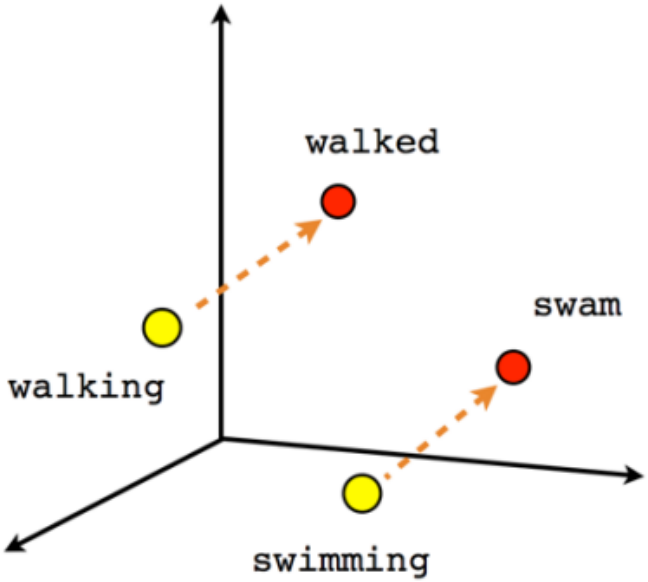
- Word2Vec solved the parameter space issue by using:
  - Bag-of-words representation
  - Neural network with single (linear) hidden layer

# Properties of Word Embeddings



- Translation in the space meaningful
- Semantics is additive
- Neighbors in space are semantically related
- Induces relationships between words such as part-of-speech, type-of and geographic relationships
- Embeddings also place similar concepts close together useful for discovering implied (but unknown) properties of them

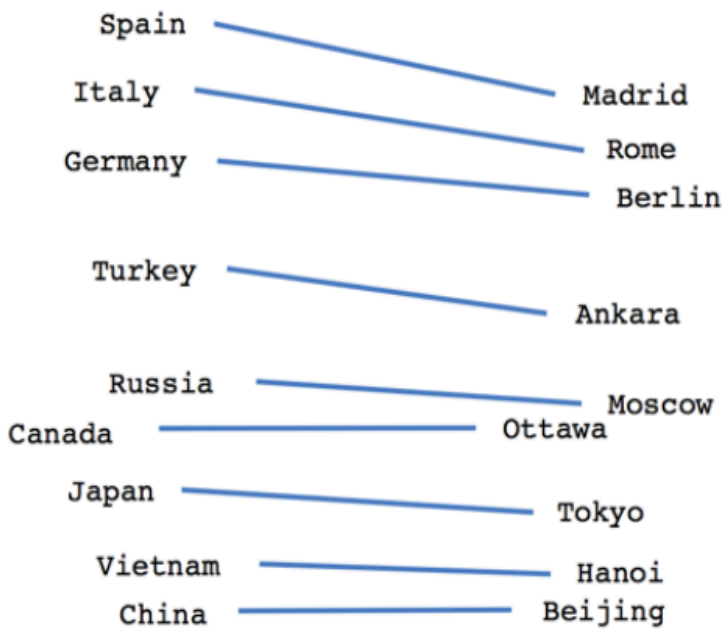



Male-Female

Verbe Tense

vector[Queen] = vector[King] - vector[Man] + vector[Woman]




Country-Capital

https://www.tensorflow.org/images/linear-relationships.png

# Sub-word embeddings

- Word embeddings work well when the vocabulary is fixed (no new words in test set).
- However, we don't have embedding for a new unseen word
- Fasttext (2016 Bojanowski et al.)
  - Splits words into character sequences
  - Learns embeddings for character n-grams
  - Combines the embeddings to form words
- Fasttext deals nicely with morphologically related terms

# Applications of Word Embeddings



- Word Similarity
  - Classic methods use  Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries
  - Word embeddings easily identifies similar words and synonyms since they occur in similar contexts
  - For example, (1) think, thought, ponder, pondering, (2) plane, aircraft, flight
- Machine Translation
- Part-of-Speech and Named Entity Recognition
- Relation Extraction (e.g., Paris-France, Rome-Italy)
- Sentiment Analysis
  - Classic methods are based on the classification of bag of word model vectors
  - Averaging the word embeddings possibly using TD/IDF approaches to weight the embeddings to classify sentences as positive and negative

# Applications of Word Embeddings



- Co-reference Resolution
  - Chaining entity mentions across multiple documents - can we find and unify the multiple contexts in which mentions occurs?
- Clustering
  - Words in the same class naturally occur in similar contexts, and have similar embeddings
  - Embeddings can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc)
- Semantic Analysis of Documents
  - Build word distributions for various topics, etc.

# Other Word Embeddings Approaches



- Latent Semantic Indexing (LSI)
  - Co-occurrence Matrix with Singular Value Decomposition)
- Fasttext - Facebook
- Word2vec (Google)
- Global Vector Representations (GloVe) (Stanford)