Vector Space Models

Vector semantics
[based on the slides of Jurafsky and Mooney]

Why semantic translation may not be enough

Consider answering question “Did Google buy YouTube” from the following sentences:

- 1. Google purchased YouTube
- 2. Google’s acquisition of YouTube
- 3. Google acquired every company
- 4. YouTube may be sold to Google
- 5. Google will buy YouTube or Microsoft
- 6. Google didn’t takeover YouTube

Why semantic translation may not be enough

Noun

- S: (n) bargain, **buy**, steal (an advantageous purchase) *"she got a bargain at the auction"; "the stock was a real buy at that price"*

Verb

- S: (v) **buy**, purchase (obtain by purchase; acquire by means of a financial transaction) *"The family purchased a new car"; "The conglomerate acquired a new company"; "She buys for the big department store"*
- S: (v) bribe, corrupt, **buy**, grease one's palms (make illegal payments to in exchange for favors or influence) *"This judge can be bought"*
- S: (v) **buy** (be worth or be capable of buying) *"This sum will buy you a ride on the train"*
- S: (v) **buy** (acquire by trade or sacrifice or exchange) *"She wanted to buy his love with her dedication to him and his work"*
- S: (v) **buy** (accept as true) *"I can't buy this story"*

Buy and purchase are related however there is no entry for the relation between **buy and acquire**

Why semantic translation may not be enough

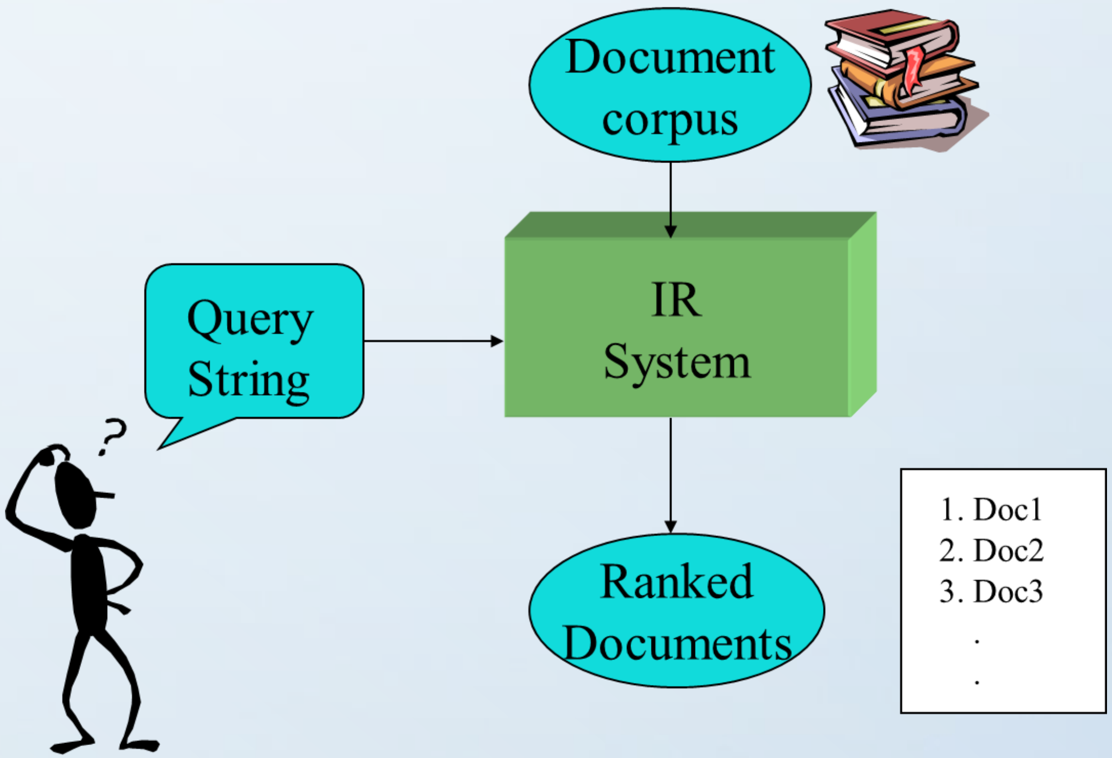
- All of these require knowledge of lexical semantics (e.g. that buy and purchase are synonyms),
- But some also need interpretation of quantifiers, negatives, modals and disjunction.
- It seems unlikely that distributional or formal approaches can accomplish the task alone.

Introduction

- Computers understand very little of the meaning of human language.
- This profoundly limits our ability to give instructions to computers, the ability of computers to explain their actions to us, and the ability of computers to analyze and process text.
- Vector space models (VSMs) of semantics are beginning to address these limits.

Vector Space Model(VSM)

- VSM was developed for the SMART information retrieval system.



Basic Idea

1. Represent each document as a point in a space i.e a vector in a vector space.

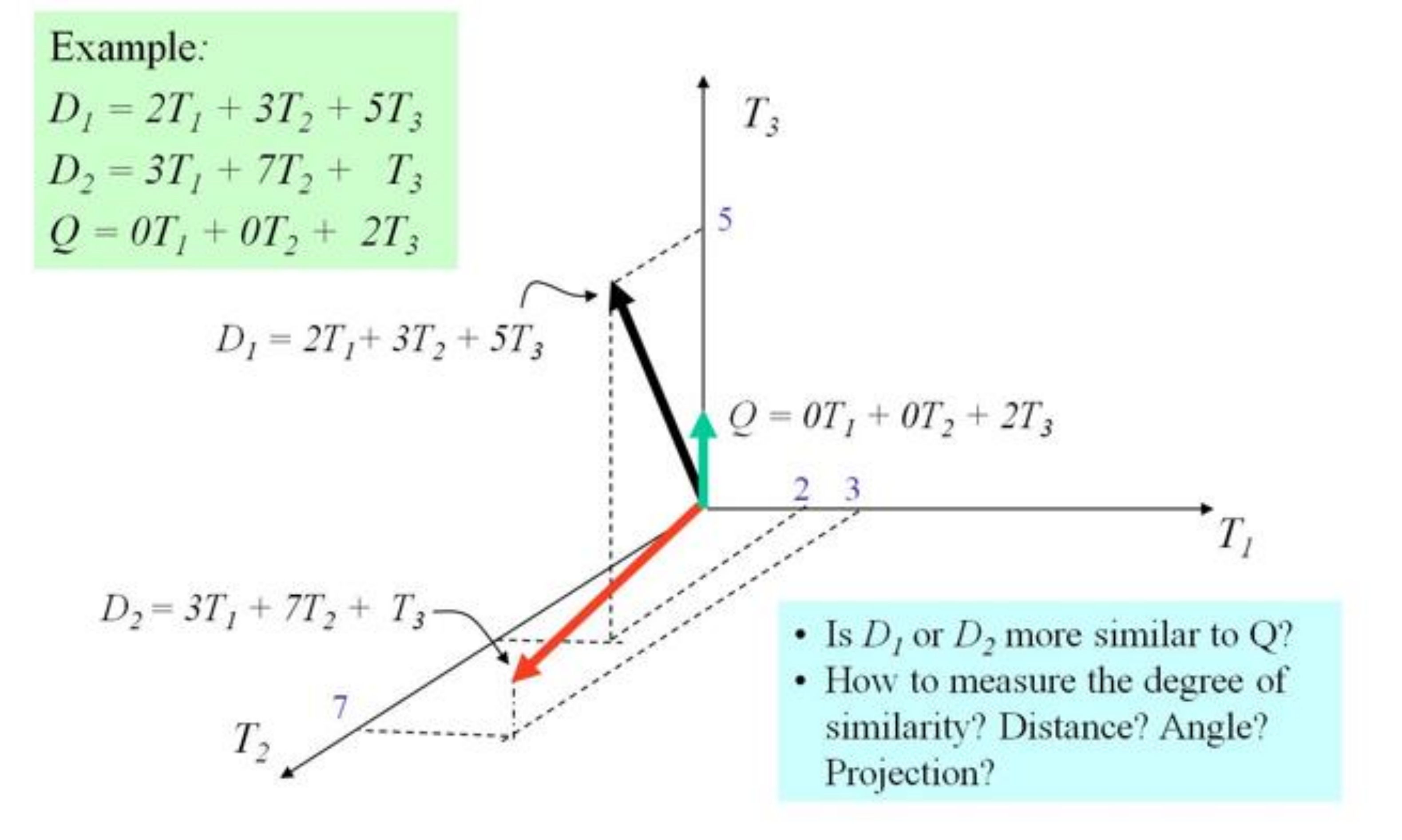
Basic Idea

1. Represent each document as a point in a space i.e. a vector in a vector space.
2. User's query is also represented as a point in the same space.

Basic Idea

1. Represent each document as a point in a space i.e. a vector in a vector space.
2. User's query is also represented as a point in the same space.
3. Sort the documents according to their distance from the query and return.

Graphic Representation



Outline

- The success of the VSM for information retrieval has inspired researchers to extend the VSM to other semantic tasks in natural language processing.
- If we have a large collection of documents, it is convenient to organize the vectors into a matrix. It turns out that VSMs can be employed in different NLP tasks by creating the matrix in different ways.
- Three different VSM Models
  - Similarity of Documents: The Term-Document Matrix
  - Similarity of Words: The Word-Context Matrix
  - Similarity of Relations: The Pair-Pattern Matrix
- Linguistic Processing for Vector Space Models
- Mathematical Processing for Vector Space Models
- Applications

Similarity of Documents: The Term-Document Matrix

$$
\begin{array}{c c}
 & \begin{array}{cccc} D_1 & D_2 & \cdots & D_n \end{array} \\
\begin{array}{c} T_1 \\ T_2 \\ \vdots \\ T_m \end{array} &
\begin{pmatrix}
f_{11} & f_{12} & \cdots & f_{1n} \\
f_{21} & f_{22} & \cdots & f_{2n} \\
\vdots & \ddots & \vdots & \\
f_{m1} & f_{m2} & \cdots & f_{mn}
\end{pmatrix}
\end{array}
$$

f_{ij} is the frequency of term T_i in document D_j

- The row vectors of the matrix correspond to terms (usually terms are words).
- The column vectors correspond to documents (web pages, for example)
- In general, the value of most of the elements in the Matrix will be 0 for obvious reasons.

Similarity of Documents: The Term-Document Matrix

$D_1 =$ “I like databases.”

$D_2 =$ “I hate databases “

	D1	D2
I	1	1
like	1	0
hate	0	1
databases	1	1

Similarity of Documents: The Term-Document Matrix

- The count vector may seem to be a rather crude representation of the document D_j.
- The sequential order of the words is lost. The vector does not attempt to capture the structure in the phrases, sentences, paragraphs, and chapters of the document.
- However, in spite of this crudeness, search engines work surprisingly well; Why?

Similarity of Documents: The Term-Document Matrix

- An intuitive justification for the *term-document* matrix is that
  - The topic of a document will probabilistically influence the author's choice of words when writing the document.
  - If two documents have similar topics, then the two corresponding column vectors will tend to have similar patterns of numbers.

Similarity of Words: The Word-Context Matrix

- Deerwester et al. (1990) observed that we can shift the focus to measuring word similarity, instead of document similarity, by looking at row vectors in the term-document matrix, instead of column vectors.
- However, document is not necessarily the optimal length of text for measuring word similarity.
- In general, we may have a *word-context* matrix, in which the context is given by words, phrases, sentences, paragraphs, chapters, documents, or more exotic possibilities, such as sequences of characters or patterns.

Similarity of Words: The Word-Context Matrix (Example)

$D_1 =$ "John went ~~to the~~ playground."

$D_2 =$ "John travelled ~~to the~~ playground."

Context is "words".

Window size = 1 (i.e. 1 word to the left and 1 word to the right)

	John	went	travelled	playground
John	0	1	1	0
went	1	0	0	1
travelled	1	0	0	1
playground	0	1	1	0

Similarity of Words: The Word-Context Matrix (Example)

- Use syntax to move beyond simple bag-of-words features.
- Parse each sentence in a large corpus.
- For each target word, produce features for it having specific dependency links to specific other words.

	nsubj =John	nsubj-of=go	nsub-of=travel	nmod=playground	nmod-of=go	nmod-of=travel
John	0	1	1	0	0	0
go	1	0	0	1	0	0
travel	1	0	0	1	0	0
playground	0	0	0	0	1	1

Similarity of Words: The Word-Context Matrix

- The distributional hypothesis in linguistics is that words that occur in similar contexts tend to have similar meanings (Harris, 1954).
- Word-Context matrix shows how that intuition could be used in a practical algorithm.

Similarity of Relations: The Pair-Pattern Matrix

- In a pair-pattern matrix,
  - row vectors correspond to pairs of words, such as <mason , stone> and <carpenter , wood>.
  - column vectors correspond to the patterns in which the pairs co-occur, such as “X cuts Y " and “X works with Y ".

Similarity of Relations: The Pair-Pattern Matrix

- Lin and Pantel (2001) introduced the pair-pattern matrix for the purpose of measuring the semantic similarity of patterns; that is, the similarity of column vectors.
- Given a pattern such as “X solves Y ", their algorithm was able to find similar patterns, such as “Y is solved by X", “Y is resolved in X", and “X resolves Y ".
- The patterns “X solves Y “and “Y is solved by X" tend to co-occur with similar X: Y pairs, which suggests that these patterns have similar meanings.

Similarity of Relations: The Pair-Pattern Matrix

- We can also use of the pair-pattern matrix for measuring the semantic similarity of relations between word pairs, that is, the similarity of row vectors.
- For example, the pairs
  - *mason : stone,*
  - *carpenter : wood,*
  - *potter : clay,*

  share the semantic relation artisan : material.
- The above pairs tend to co-occur in similar patterns, such as “the X used the Y to" and “the X shaped the Y into".

Linguistic Processing for Vector Space Models

Linguistic Processing for Vector Space Models

Before we generate a term-document, word-context, or pair-pattern matrix, it can be useful to apply some linguistic processing to the raw text. The types of processing that are used can be grouped into three classes.

- First, we need to ***tokenize*** the raw text; that is, we need to decide what constitutes a term and how to extract terms from raw text.
- Second, we may want to ***normalize*** the raw text, to convert superficially different strings of characters to the same form (e.g., car, Car, cars, and Cars could all be normalized to car).
- Third, we may want to ***annotate*** the raw text, to mark identical strings of characters as being different (e.g., fly as a verb could be annotated as fly/VB and fly as a noun could be annotated as fly/NN).

Tokenization

- Tokenization of English seems simple at first glance however this assumption is approximately true.
- An accurate English tokenizer must know how to handle
  - punctuation (e.g., don't, Jane's)
  - hyphenation (e.g., state-of-the-art versus state of the art, bat-and-ball), and
  - recognize multi-word terms (e.g., Barack Obama and ice hockey)
- We may also wish to ignore stop words, high-frequency words with relatively low information content, such as function words (e.g., of, the, and) and pronouns (e.g., them, who, that).

Normalization

- The most common types of normalization are
  - **case folding** : converting all words to lower case,
  - **stemming, lemmatization:** often a word is composed of a stem (root) with added affixes (inflections), such as plural forms and past tenses Stemming /Lemmatization, is the process of reducing inflected words to their stems.

    am, are, is → be

    car, cars, car's, cars' → car
- Case folding can be problematic. For example,
  - SMART is an information retrieval system, whereas smart is a common adjective;
  - Bush may be a surname, whereas bush is a kind of plant.

Annotation

- Annotation is the inverse of normalization.
- Just as different strings of characters may have the same meaning, it also happens that identical strings of characters may have different meanings, depending on the context.
- Common forms of annotation include
  - part-of-speech, tagging (marking words according to their parts of speech),
  - word sense tagging (marking ambiguous words according to their intended meanings), and
  - parsing (analyzing the grammatical structure of sentences and marking the words in the sentences according to their grammatical roles)

Example

D_1 = “John went to the playground.”

D_2 = “John travelled to the playground “

<u>Do</u>: tokenize, stop word removal, case folding, lemmatization, pos-tagging

<u>Result:</u>

D_1 = “john-nnp go-vbd playground-nn”

D_2 = “john-nnp travel-vbd playground-nn “

Mathematical Processing for Vector Space Models

Mathematical Processing for Vector Space Models

- After the text has been tokenized and (optionally) normalized and annotated, the first step is to generate a matrix of frequencies.
- However, we may want to adjust the weights of the elements in the matrix, because common words will have high frequencies, yet they are less informative than rare words.
- Third, we may want to smooth the matrix, to reduce the amount of random noise and to fill in some of the zero elements in a sparse matrix.
- Fourth, there are many different ways to measure the similarity of two vectors.

Weighting the Elements

- An element in a frequency matrix corresponds to an event: a certain item (term, word, word pair) occurred in a certain situation (document, context, pattern) a certain number of times (frequency).
- The idea of weighting is to give more weight to surprising events and less weight to expected events.
- The hypothesis is that surprising events, if shared by two vectors, are more discriminative of the similarity between the vectors than less surprising events. For example,
  - in measuring the semantic similarity between the words mouse and rat, the contexts *dissect* and *exterminate* are more discriminative of their similarity than the contexts *have* and *like*.

Term frequency

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

Log-frequency weighting

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1+\log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Score for a document-query pair: sum over terms *t* in both *q* and *d*:
- Score $= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Document frequency

- Rare terms are more informative than frequent terms
- Consider a term in the query that is rare in the collection (e.g., arachnocentric)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like arachnocentric.

idf weight

- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $\text{df}_t \leq N$ (N is the total number of documents)
- We define the idf (inverse document frequency) of *t* by

$$\text{idf}_t = \log_{10} (N/\text{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

idf example, suppose N = 1 million

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

There is one idf value for each term t in a collection.

Effect of idf on ranking

- Does idf have an effect on ranking for one-term queries, like
  - iPhone
- idf has no effect on ranking one term queries
  - idf affects the ranking of documents for queries with at least two terms
  - For the query capricious person, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

tf-idf

- The tf-idf weight of a term is **the product of its tf weight and its idf weight**.

$$w_{t,d} = \log(1+\text{tf}_{t,d}) \times \log_{10}(N/\text{df}_t)$$

- Best known weighting scheme in information retrieval
- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Weighting the Elements

- Another kind of weighting, often combined with tf-idf weighting, is **length normalization**.
- In information retrieval, if document length is ignored, search engines tend to have a bias in favor of longer documents. Length
- Normalization corrects for this bias.

Pointwise Mutual Information

- An alternative to tf-idf is Pointwise Mutual Information (PMI) which works well for both word-context matrices and term-document matrices.
- A variation of PMI is Positive PMI (PPMI), in which all PMI values that are less than zero are replaced with zero.
- Here f_{ij} is the (i, j)-th element in the raw frequency matrix (either term-document matrix, word-context matrix or the pair-pattern matrix).

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}}$$

$$p_{i*} = \frac{\sum_{j=1}^{n_c} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}}$$

$$p_{*j} = \frac{\sum_{i=1}^{n_r} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}}$$

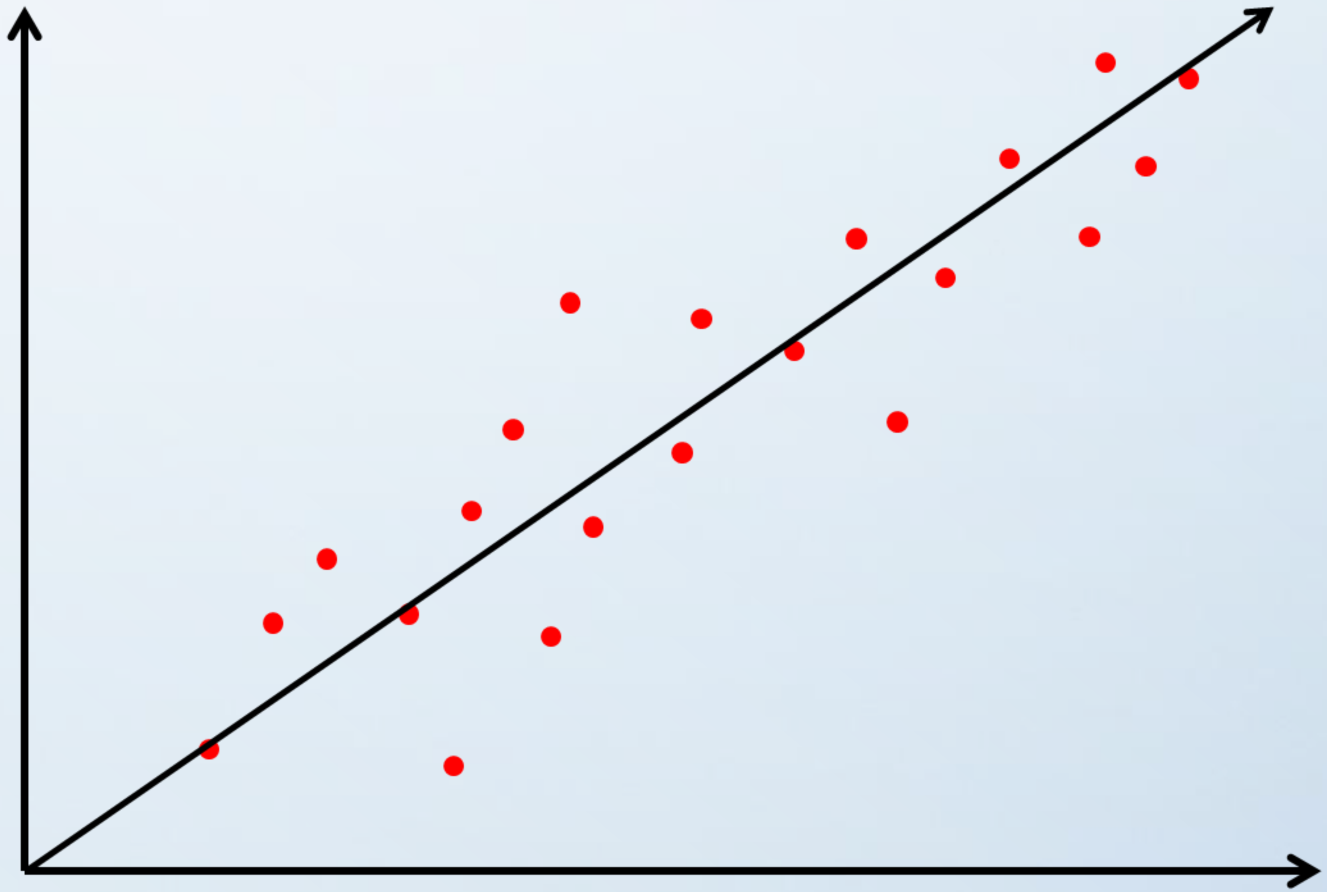
$$\mathrm{pmi}_{ij} = \log\left(\frac{p_{ij}}{p_{i*}p_{*j}}\right)$$

$$x_{ij} = \begin{cases} \mathrm{pmi}_{ij} & \text{if } \mathrm{pmi}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

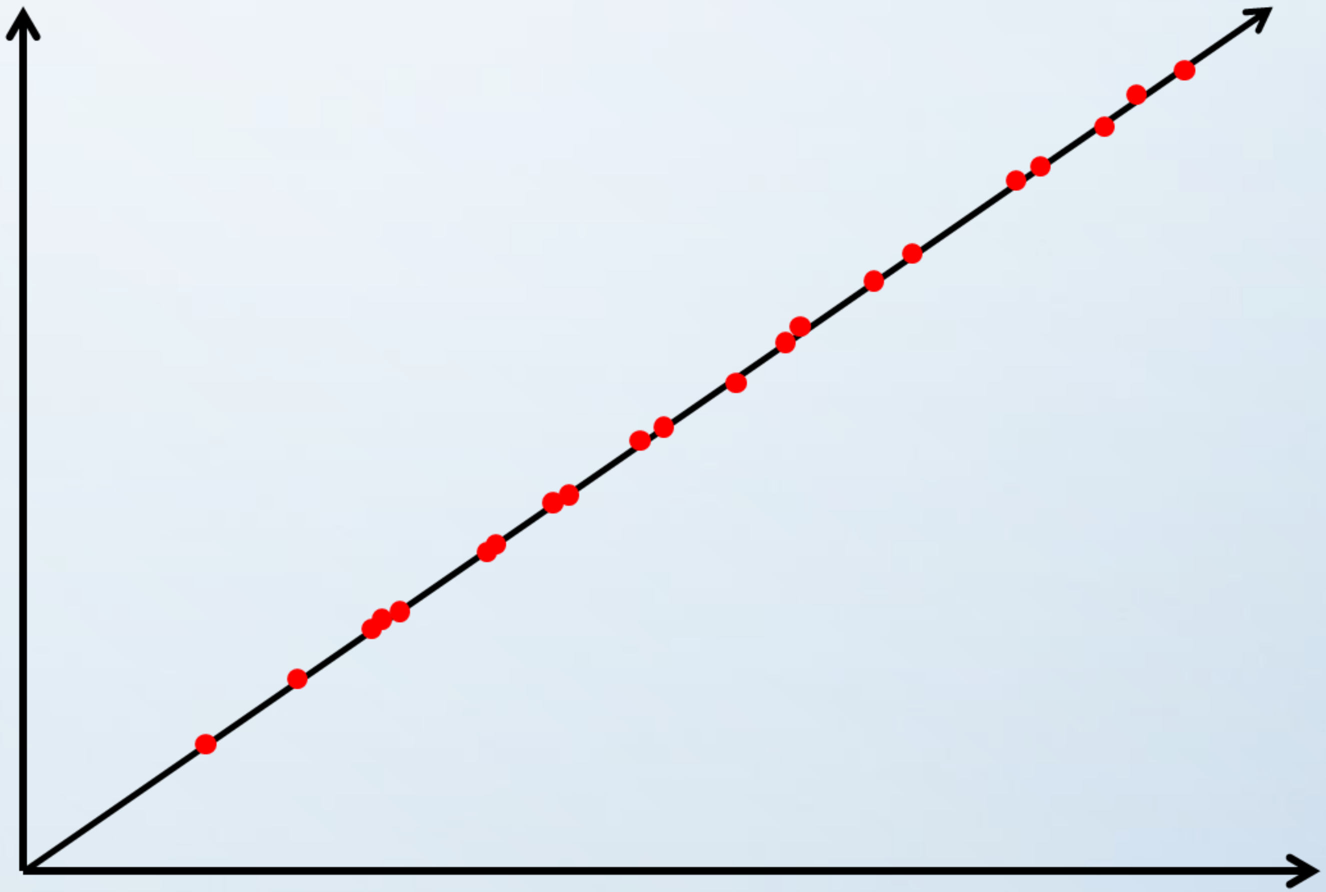
Dimensionality reduction

- Word-based features result in extremely high-dimensional spaces that can easily result in over-fitting.
- Reduce the dimensionality of the space by using various mathematical techniques to create a smaller set of k new dimensions that most account for the variance in the data.
  - Singular Value Decomposition (SVD) used in Latent Semantic Analysis (LSA)
  - Principle Component Analysis (PCA)

Simple Dimensionality reduction



Simple Dimensionality Reduction



High-order co-occurrence

- **Direct co-occurrence** (first-order co-occurrence) is when two words appear in identical contexts.
- **Indirect co-occurrence** (high-order co-occurrence) is when two words appear in similar contexts. Similarity of contexts may be defined recursively in terms of lower-order co-occurrence.
- It was demonstrated that (truncated) SVD can discover high-order co-occurrence.

Comparing the Vectors

- The most popular way to measure the similarity of two frequency vectors (raw or weighted) is to take their cosine. Let x and y be two vectors, each with n elements.

$$\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$$
$$\mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle$$

- The cosine of the angle between x and y can be calculated as follows:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}}$$

Comparing the Vectors

- A measure of distance between vectors can easily be converted to a measure of similarity by inversion or subtraction .

$$\mathrm{sim}(\mathbf{x},\mathbf{y}) = 1/\mathrm{dist}(\mathbf{x},\mathbf{y})$$
$$\mathrm{sim}(\mathbf{x},\mathbf{y}) = 1 - \mathrm{dist}(\mathbf{x},\mathbf{y})$$

- Many similarity measures have been proposed in both and lexical semantics circles.
- It is commonly said in IR that, properly normalized, the difference in retrieval performance using different measures is insignificant.

Applications: Term-Document Matrices

- **Document retrieval**
- **Document clustering**: Given a measure of document similarity, we can cluster the documents into groups, such that similarity tends to be high within a group, but low across groups.
- **Essay grading**: Student essays may be automatically graded by comparing them to one or more high-quality reference essays on the given essay topic.
- **Document segmentation**: The task of document segmentation is to partition a document into sections, where each section focuses on a different subtopic of the document. We may treat the document as a series of blocks, where a block is a sentence or a paragraph. The problem is to detect a topic shift from one block to the next.
- **Call routing:** "How may I direct your call?"

Applications: Word-Context Matrices

- Word classification (e.g. classify words as positive (honest, intrepid) or negative (disturbing, superfluous).
- Automatic thesaurus generation
- **Context-sensitive spelling correction**: People frequently confuse certain sets of words, such as *there*, *they're*, and *their*. These confusions cannot be detected by a simple dictionary-based spelling checker.

Applications: Word-Context Matrices

- Word Sense Induction(How?)
  - Create a context-vector for ***each individual occurrence*** of the target word, *w*.
  - Cluster these vectors into *k* groups.
  - Assume each group represents a "sense" of the word and compute a vector for this sense by taking the mean of each cluster

Applications: Pair-Pattern Matrices

- **SAT analogy questions**

Directions: In the following question, a related pair of words or phrases is followed by five pairs of words or phrases. Choose the pair that best expresses a relationship similar to that in the original pair

MEDICINE : ILLNESS ::

1. law : anarchy
2. hunger : thirst
3. etiquette : discipline
4. love : treason
5. stimulant : sensitivity

Turney (2006) evaluated this approach to relational similarity with 374 multiple-choice analogy questions from the SAT college entrance test, achieving human-level performance (**56% correct for the pair-pattern matrix and 57% correct for the average US college applicant**). The best non-VSM algorithm achieves 43%.

Note: VSM means Vector Space Model

Applications: Pair-Pattern Matrices

- **Automatic thesaurus generation**: Snow, Jurafsky, and Ng (2006) used a pair-pattern matrix to build a hypernym-hyponym taxonomy, whereas Pennacchiotti and Pantel (2006) built a meronymy and causation taxonomy.

Note: Meronymy refers to part-of relationships

Detecting Antonyms

- Lin et al. (2003) distinguish synonyms from antonyms using two patterns, "from X to Y " and "either X or Y ". When X and Y are antonyms, they occasionally appear in a large corpus in one of these two patterns, but it is very rare for synonyms to appear in these patterns.
- Using a Pair-Pattern Matrix one can automatically discover these patterns and more to solve this task.

References

- **From Frequency to Meaning, Turney & Pantel** [https://www.jair.org/media/2934/live-2934-4846-jair.pdf]

End