

Natural Language Processing

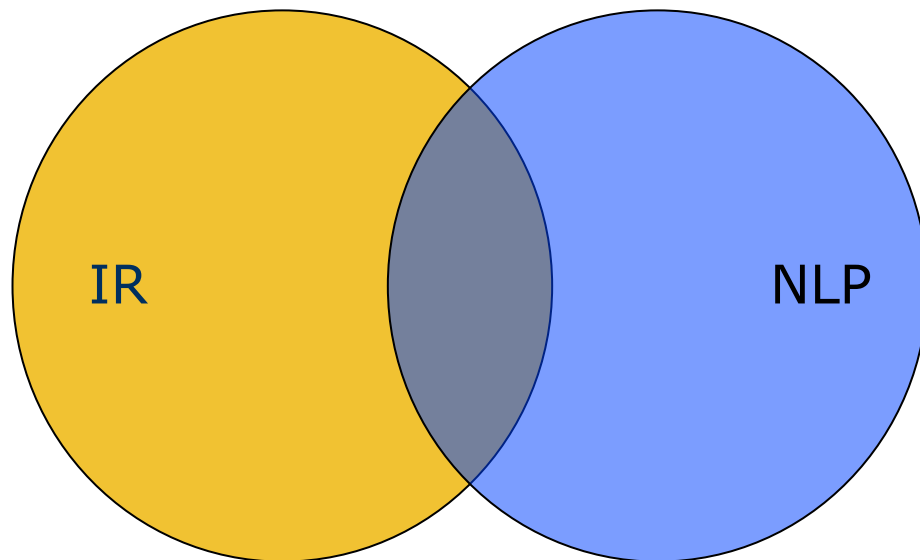
An introduction for Information Retrieval

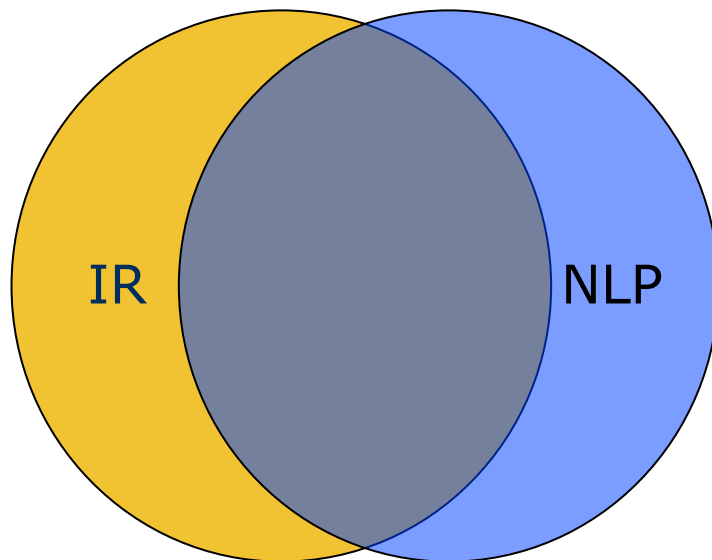
NLP – Natural Language Processing

- NLP combines *linguistics* and *computer science*
 - Sometimes referred to as *computational linguistics*
 - Also incorrectly confused with *AI* in general
- How do we *process* human language using computers
- One of the “holy grails” of computation

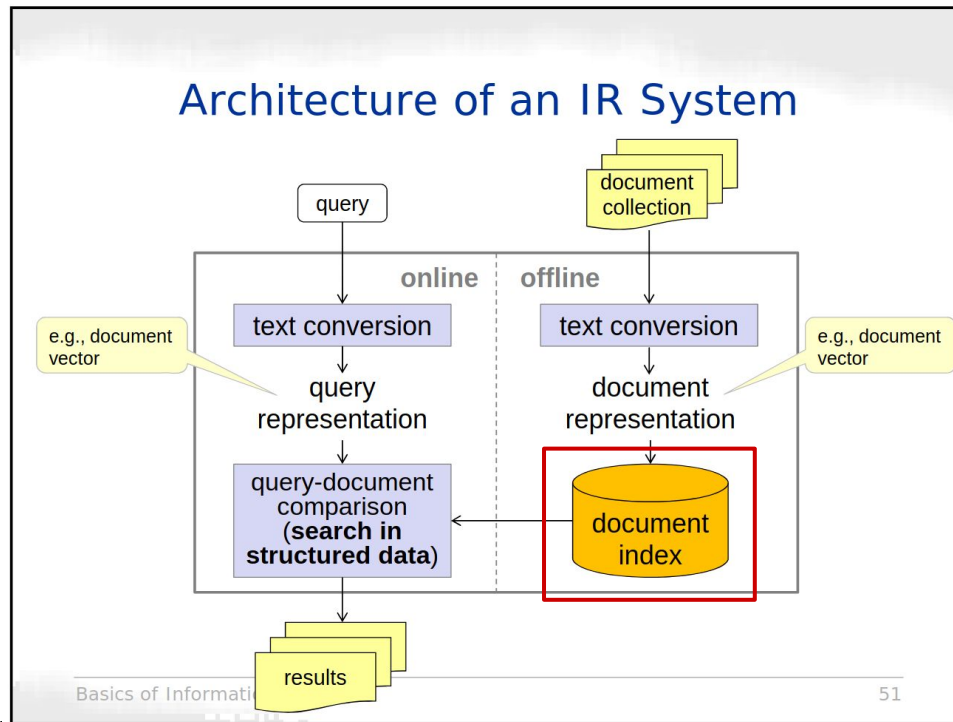
Examples of NLP applications

- Spell checking
- Machine translation
- Sentiment analysis and opinion mining
- Text summarization
- Chatbots
- *Search engines?*





From lecture 3



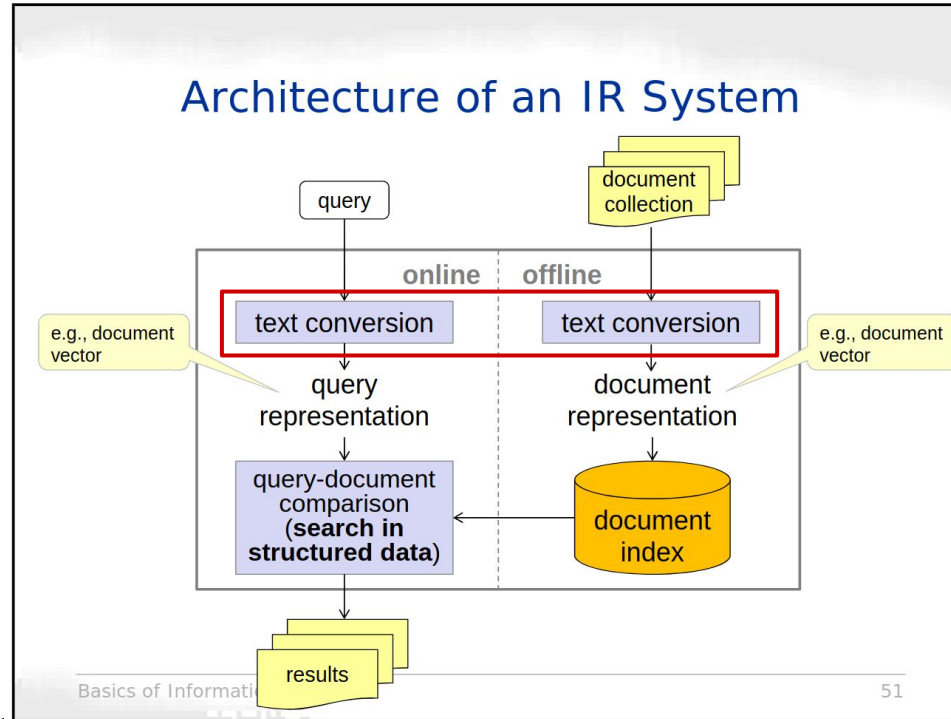
Document Index

- ◆ Index provides fast access to data given the key

Document	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

Word	(Document; Word)
cold	(1; 6), (4; 8)
days	(3; 2), (6; 2)
hot	(1; 3), (4; 4)
in	(2; 3), (5; 4)
it	(4; 3, 7), (5; 3)
like	(4; 2, 6), (5; 2)
nine	(3; 1), (6; 1)
old	(3; 3), (6; 3)
pease	(1; 1, 4), (2; 1)
porridge	(1; 2, 5), (2; 2)
pot	(2; 5), (5; 6)
some	(4; 1, 5), (5; 1)
the	(2; 4), (5; 5)

From lecture 3



Part 1: Pre-processing

Tokenization

Tokenization is converting a string of words into a sequence of discrete *tokens*

Tokenization

| Tokenization | is converting a string of
words into a sequence of discrete *tokens*

Tokenization

Tokenization is converting a string of words into a sequence of discrete *tokens*

Document representation

Tokenization is converting a string of
words into a sequence of discrete *tokens*



tokenization
is
converting
a
string
words
into
sequence
of
discrete
tokens

A good (?) query

“convert word strings”

tokenization
is
converting
a
string
of
words
into
sequence
discrete
tokens

A good (?) query

“convert word strings”

Will this query work?

tokenization
is
converting
a
string
of
words
into
sequence
discrete
tokens

Inflection in the document

“convert word strings”

tokenization
is
convert^{ing}
a
string
of
word^s
into
sequence
discrete
tokens

Inflections in the query

"convert word string^s"

tokenization
is
converting
a
string
of
words
into
sequence
discrete
tokens

Morphology

- The study of words and how they are formed
- *Morpheme*: the smallest **meaningful** unit of a word

m e a n i n g f u l

Morphology

- The study of words and how they are formed
- *Morpheme*: the smallest **meaningful** unit of a word

1	2	3	4	5	6	7	8	9
m	e	a	n	i	n	g	f	u
1	2	3	4	5	6	7	8	9

Morphology

- The study of words and how they are formed
- *Morpheme*: the smallest **meaningful** unit of a word

m e a n | i n g | f u l

4 7

4 7

Mean-ing-ful

Etymology [\[edit \]](#)

From *meaning* + *-ful*.

Suffix [\[edit \]](#)

-ful

1. Used to form *adjectives* from *nouns*, with the sense of being *full* of, *tending* to, or *thoroughly* possessing the *quality* expressed by the noun.

sin + *-ful* → *sinful*

Etymology 1 [\[edit \]](#)

From Middle English *mening*, *menyng*, equivalent to *mean* + *-ing*.

Suffix [\[edit \]](#)

-ing

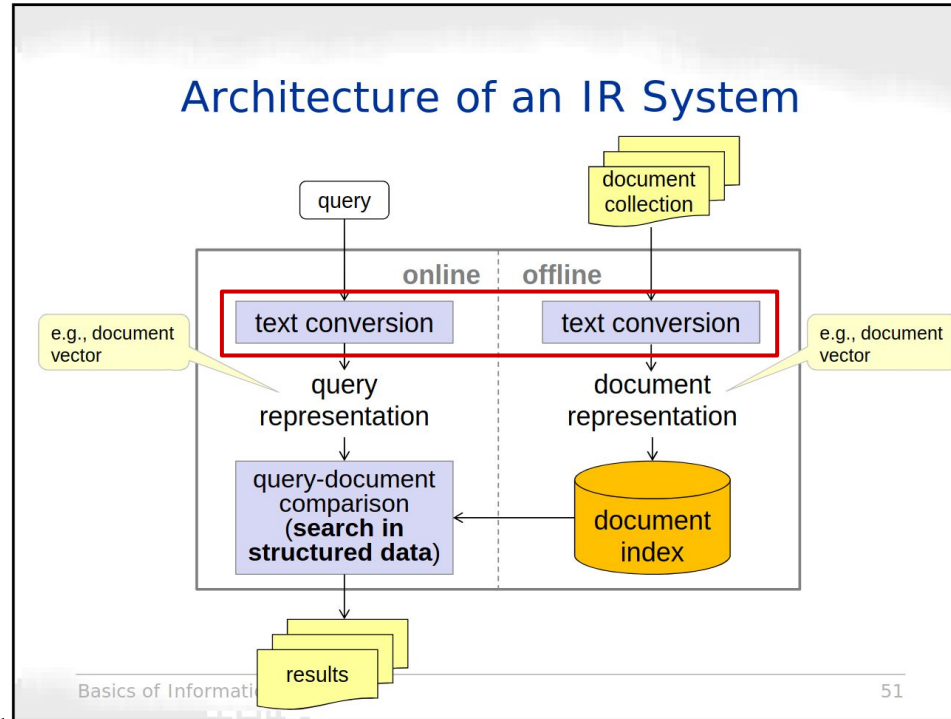
1. Used to form *nouns* or noun-like words (or elements of noun phrases) from verbs, denoting the act of doing something, an action, or the embodiment of an action.

<https://en.wiktionary.org/wiki/meaningful>
<https://en.wiktionary.org/wiki/meaning#English>
<https://en.wiktionary.org/wiki/-ing#English>
<https://en.wiktionary.org/wiki/-ful#English>

Morphology

- The study of words and how they are formed
- *Morpheme*: the smallest **meaningful** unit of a word
- Affixes
 - **Suffixes** after words: *amazing*
 - **Prefixes** before words: *undeniable*

From lecture 3



Inflections in the query

"convert word string^s"

tokenization
is
convert^{ing}
a
string
of
word^s
into
sequence
discrete
tokens

Stemming

- Words can have multiple affixes
- Stemming cuts these, leaving only the “root” or “stem”
- A popular English stemmer is the *Snowball stemmer*
 - Example: *meaningful* → *mean(ing)*
 - Example: *universities* → *univers*

Stemming to the rescue

"convert word strings"

tokenization

is

converting

a

string

of

words

into

sequence

discrete

tokens

Stemming

- Words can have multiple affixes
- Stemming cuts these, leaving only the “root” or “stem”
- A popular English stemmer is the *Snowball stemmer*
 - Example: *meaningful* → *mean(ing)*
 - Example: *universities* → *univers*

Stemming

- Words can have multiple affixes
- Stemming cuts these, leaving only the “root” or “stem”
- A popular English stemmer is the *Snowball stemmer*
 - Example: *meaningful* → *mean(ing)*
 - Example: *universities* → *univers*
 - **Example: *universal* → *univers***

Lemmatization

- An alternative to stemming is *lemmatization*
- The lemma of a word is the “dictionary form”
- Example: *universities* → *university* (not “univers”)
- Typically implemented with structured resources
 - In the lab: *WordNet Lemmatizer*

WordNet

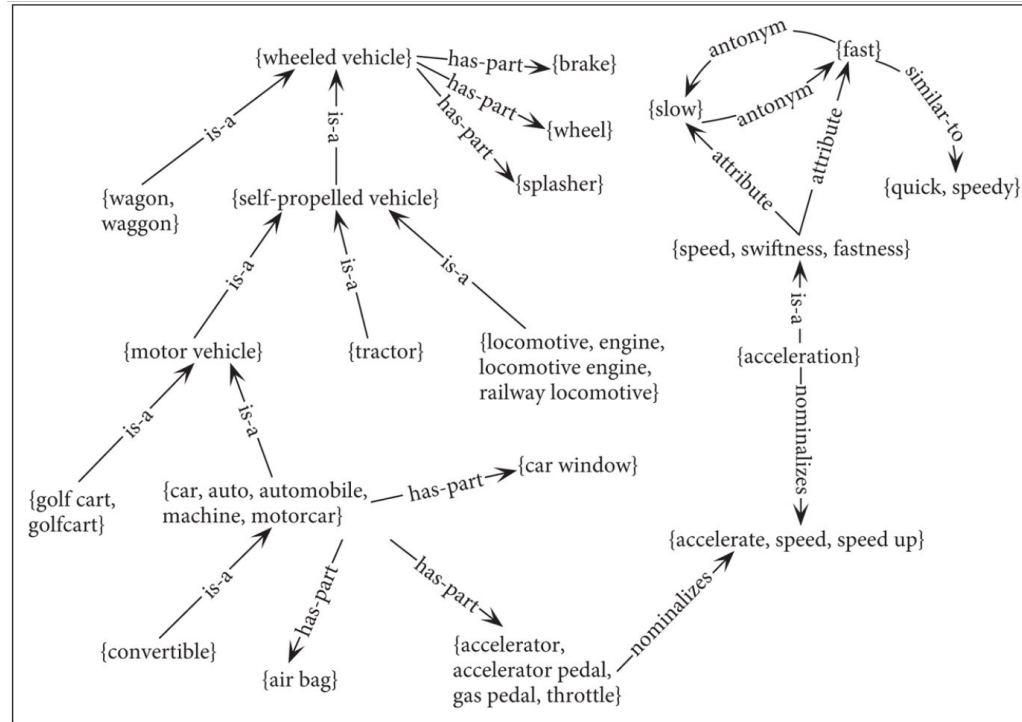


Figure G.6 WordNet viewed as a graph. Figure from Navigli (2016).

From [Jurafsky & Martin \(2024\)](#)

Compound splitting

- Words can be built by adding words together
- English: Dish-washer
- Swedish: Golv-slip-s-maskin-ut-hyrning
- German: Fuss-boden-schleif-maschinen-verleih



Agglutinative languages

- Languages are on a spectrum in terms of morphology
 - More analytic (e.g., English and Mandarin)
 - More agglutinative (e.g., Turkish and Finnish)
- Agglutinative languages can have very long words!
 - Finnish: *ylioppilastutkintolautakunta*
 - Turkish: *Muvaffakiyetsizleştiricileştiriveremeyebileceklerimiz*
-denmişsinizcesine

Morphology

- The study of words and how they are formed
- *Morpheme*: the smallest **meaningful** unit of a word
- Affixes
 - **Suffixes** after words: *amazing*
 - **Prefixes** before words: *undeniable*
 - **Infixes** inside words: ???

Infixes: Arabic

- Arabic words are defined by *root consonants*
- Example: k-t-b (ك-ت-ب)
 - **kitāb** = book
 - **kutub** = books (plural)
 - **kātib** = writer
- What is a stem here? A lemma?
 - In practice: root *or* base form
 - **kutub** → **k-t-b** *or* **kitāb**

Arabic derivatives [edit]

A full account of derivatives could be extensive, but some of them are these:

- kataba** كَتَبَ or كُتِبَ "he wrote" (masculine)
- katabat** كَتَبَتْ or كُتِبَتْ "she wrote" (feminine)
- katabtu** كَتَبْتُ or كُتِبْتُ "I wrote" (f and m)
- kutiba** كُتِبَ or كُتِبَ "it was written" (masculine)
- kutibat** كُتِبَتْ or كُتِبَتْ "it was written" (feminine)
- katabū** كَتَبُوا or كُتِبُوا "they wrote" (masculine)
- katabna** كَتَبْنَ or كُتِبْنَ "they wrote" (feminine)
- katabnā** كَتَبْنَا or كُتِبْنَا "we wrote" (f and m)
- yaktub(u)** يَكْتُبُ or يَكْتُبُ "he writes" (masculine)
- taktub(u)** تَكْتُبُ or تَكْتُبُ "she writes" (feminine)
- naktub(u)** نَكْتُبُ or نَكْتُبُ "we write" (f and m)
- aktub(u)** أَكْتُبُ or أَكْتُبُ "I write" (f and m)
- yuktub(u)** يُكْتُبُ or يُكْتُبُ "being written" (masculine)
- tuktub(u)** تُكْتُبُ or تُكْتُبُ "being written" (feminine)
- yaktubūn(a)** يَكْتُوبُونَ or يَكْتُوبْنَ "they write" (masculine)
- yaktubna** يَكْتُوبْنَ or يَكْتُوبْنَ "they write" (feminine)
- taktubna** تَكْتُوبْنَ or تَكْتُوبْنَ "you write" (feminine)
- yaktubān(i)** يَكْتُوبَانِ or يَكْتُوبَانِ "they both write" (masculine) (for 2 males)
- taktubān(i)** تَكْتُوبَانِ or تَكْتُوبَانِ "they both write" (feminine) (for 2 females)
- kātaba** كَاتَبَ or كَاتَبَ "he exchanged letters (with sb.)"
- yukātib(u)** يُكَاتِبُ "he exchanges (with sb.)"
- yatakātubūn(a)** يَتَكَاثَبُونَ or يَتَكَاثَبْنَ "they write to each other" (masculine)
- ikataba** اِكْتَبَ or اِكْتَبَ "he is registered" (intransitive) or "he contributed (a money quantity to sth.)" (ditransitive)
(the first t is part of a particular verbal [transfix](#), not part of the root)
- istaktaba** اِسْتَكْتَبَ "to cause to write (sth.)"
- kitāb** كِتَاب or كِتَاب "book" (the hyphen shows end of stem before various case endings)
- kutub** كُتُب or كُتُب "books" (plural)
- kutayyib** كُتَيْب or كُتَيْب "booklet" (diminutive)
- kitābat** كِتَابَةٌ or كِتَابَةٌ "writing"
- kātib** كَاتِب or كَاتِب "writer" (masculine)
- kātibat** كَاتِبَةٌ or كَاتِبَةٌ "writer" (feminine)
- kātibūn(a)** كَاتِبُونَ or كَاتِبُونَ "writers" (masculine)
- kātibāt** كَاتِبَات or كَاتِبَات "writers" (feminine)
- kuttāb** كُتَّاب or كُتَّاب "writers" (broken plural)
- katabat** كَاتِبَةٌ or كَاتِبَةٌ "clerks" (broken plural)
- maktab** مَكْتَب or مَكْتَب "desk" or "office"
- makātib** مَكَاتِب or مَكَاتِب "desks" or "offices"
- maktabat** مَكْتَبَةٌ or مَكْتَبَةٌ "library" or "bookshop"
- maktūb** مَكْتُوب or مَكْتُوب "written" (participle) or "postal letter" (noun)
- kātibat** كَاتِبَةٌ or كَاتِبَةٌ "squadron" or "document"
- kātā'ib** كَاتِبَات or كَاتِبَات "squadrons" or "documents"
- iktibā** اِكْتِبَاء or اِكْتِبَاء "registration" or "contribution of funds"
- muktabib** مَكْتَبِي or مَكْتَبِي "subscriber"
- muktabab** مَكْتَبَات or مَكْتَبَات "subscription"
- istiktab** اِسْتِكْتَاب or اِسْتِكْتَاب "causing to write"

Language diversity

- A lot of NLP research is done in English
- English is not a very “representative” language
 - In fact, it’s #33 on the [Language Weirdness Index](#)
- Make sure your algorithms work for your language!

Summary of Part 1

- *Tokenization* splits input into distinct words
- *Morphology* describes how words are inflected
 - *Affixes: suffixes, prefixes and infixes*
 - Languages vary a lot! E.g., *compound splitting*.
- *Stemming* algorithmically strips morphological nuance
- *Lemmatization* reduces words to their dictionary form
- Both can be useful *depending on your problem*

Part 2: Words as vectors

Example problem

Craftbeer bar SoFo

Beer pub Södermalm

No words in common, but very similar meaning.

Problem: computers have no idea what words *mean*!

Semantics

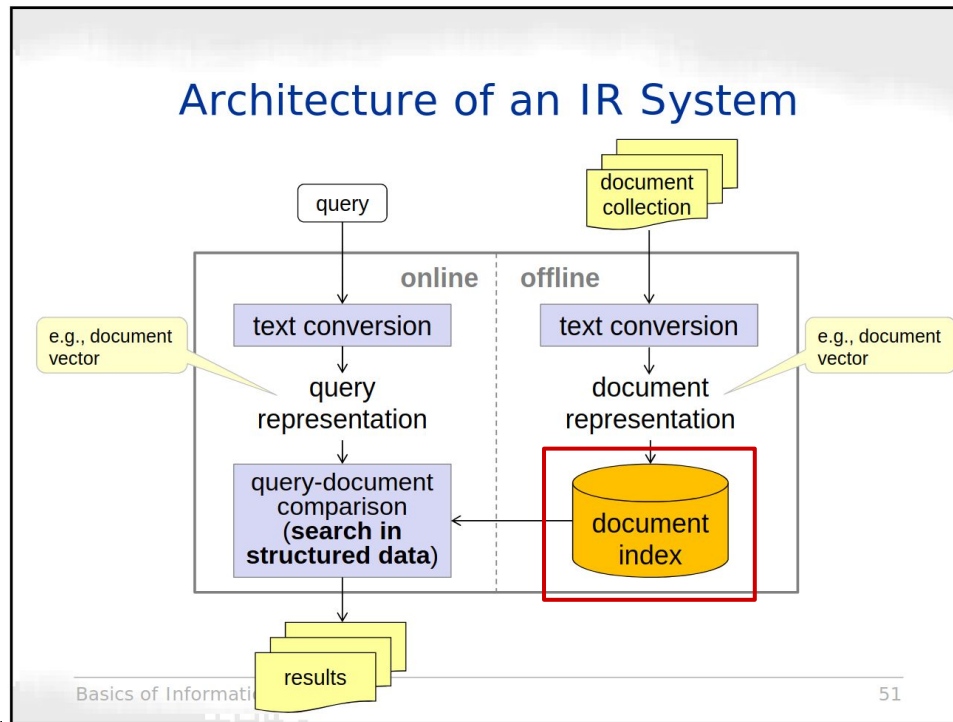
- What do words *mean*?
- One view: dictionary meaning (Oxford English Dictionary)

I. The animal.

I.1.a. A well-known carnivorous quadruped (*Felis domesticus*) which has long been domesticated, being kept to destroy mice, and as a house pet.

- How can this be represented computationally?

From lecture 3



Term-frequency index

	Basketball	Information retrieval	Baseball	Natural Language Processing	Graphics processing unit
<i>sport</i>	56	0	87	0	0
<i>computer</i>	0	14	0	16	41
<i>semantic</i>	0	3	0	46	0
<i>language</i>	0	7	2	123	2
<i>ball</i>	425	0	523	0	0
...					

Term-document matrix

	Basketball	Information retrieval	Baseball	Natural Language Processing	Graphics processing unit
<i>sport</i>	56	0	87	0	0
<i>computer</i>	0	14	0	16	41
<i>semantic</i>	0	3	0	46	0
<i>language</i>	0	7	2	123	2
<i>ball</i>	425	0	523	0	0
...					

Similar term-frequency vectors $\Rightarrow$ similar documents

Term-document matrix

	Basketball	Information retrieval	Baseball	Natural Language Processing	Graphics processing unit
<i>sport</i>	56	0	87	0	0
<i>computer</i>	0	14	0	16	41
<i>semantic</i>	0	3	0	46	0
<i>language</i>	0	7	2	123	2
<i>ball</i>	425	0	523	0	0
...					

Similar **document** vectors $\Rightarrow$ similar **words**

Distributional hypothesis

Ludwig Wittgenstein (1953):

"The meaning of a word is its use in the language"

John Rupert Firth (1957):

"You shall know a word by the company it keeps"

We learn the meaning based on its use in a **corpus**.

Co-occurrence matrix

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Co-occurrence matrix: vectors

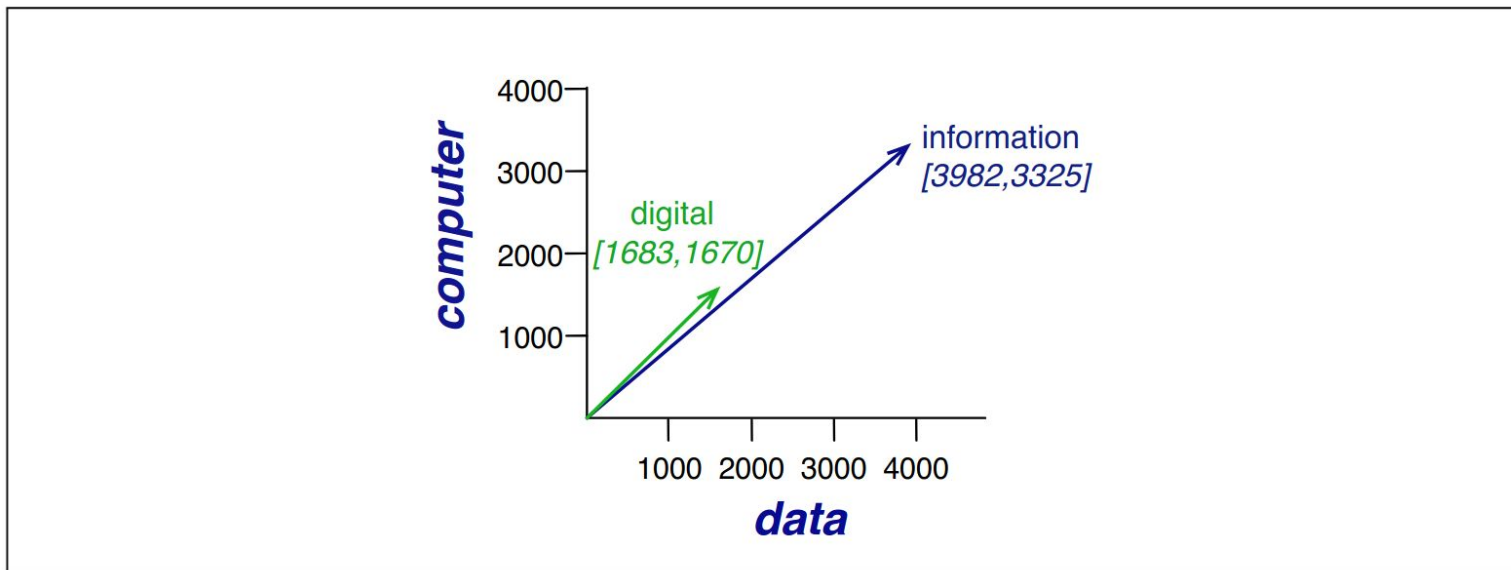


Figure 6.7 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

Word2Vec

- Word2Vec creates vectors that are *learned*
- Vectors are learned using the *skip-gram algorithm*
- Predict a *target word* from its *context*

All cats use whiskers to sense their surroundings

Word2Vec

- Word2Vec creates vectors that are *learned*
- Vectors are learned using the *skip-gram algorithm*
- Predict a *target word* from its *context*

Target word

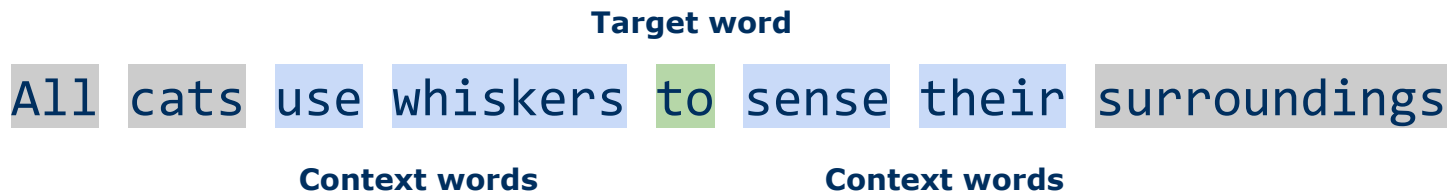
All cats use whiskers to sense their surroundings

Context words Context words

We process the corpus with a *sliding window*.

Word2Vec

- Word2Vec creates vectors that are *learned*
- Vectors are learned using the *skip-gram algorithm*
- Predict a *target word* from its *context*



We process the corpus with a *sliding window*.

Word2Vec

- Word2Vec creates vectors that are *learned*
- Vectors are learned using the *skip-gram algorithm*
- Predict a *target word* from its *context*

Target word

All cats use whiskers to sense their surroundings

Context words Context words

We process the corpus with a *sliding window*.

What does Word2Vec capture?

- Word2Vec effectively captures the semantic of words
- Allows for “semantic arithmetic”
 - Example: $\text{king} - \text{man} + \text{woman} \approx \text{queen}$
- Words with similar meaning are close in the space
 - Typically measured using the *cosine similarity*
- Other similar techniques: GloVe, FastText, etc

Demo: <http://projector.tensorflow.org/>

Harmful bias

- Word2Vec learns from corpora written by humans
- Humans have biases – these get reflected in vectors
- Bias in NLP and IR can lead to harmful systems

Example of bias from training on news texts:

king - man + woman $\approx$ queen

computer programmer - man + woman $\approx$ homemaker

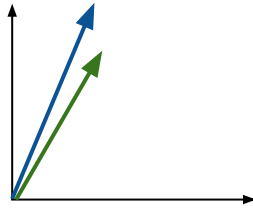
More examples in [Jurafsky & Martin \(2024\) section 6.11](#)

A final example

Thomas got a loan from the bank

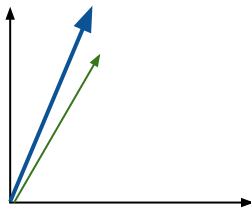
A final example

Thomas got a loan from the bank



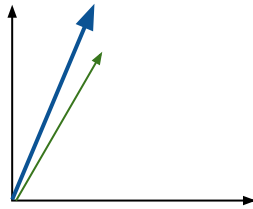
A final example

Thomas got a loan from the bank
His bank sits on



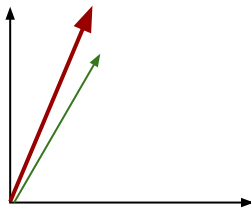
A final example

Thomas got a loan from the bank
His bank sits on the river bank

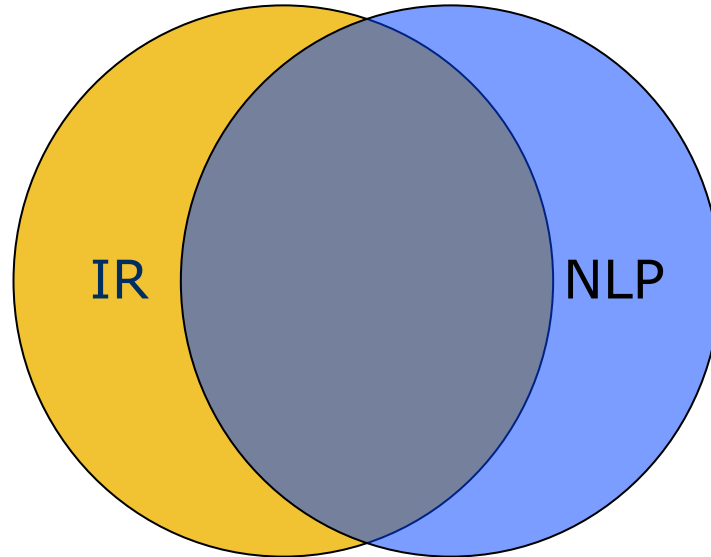


No context!

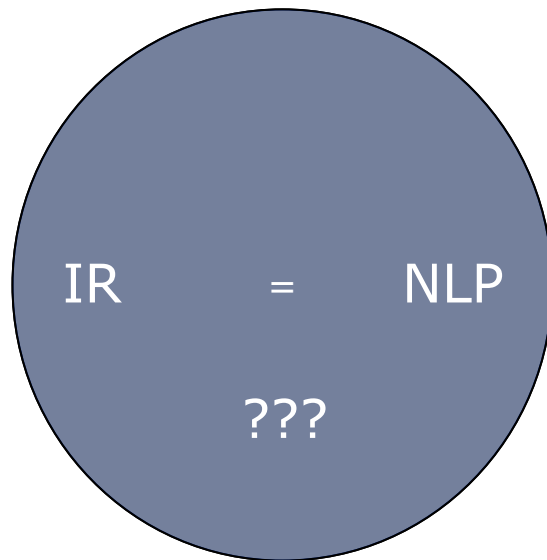
Thomas got a loan from the bank
His bank sits on the river bank



Next lecture: LLMs



Next lecture: LLMs



Maybe not quite...

Summary of Part 2

- *Semantic word vectors* can be learned from *corpora*
 - This is based on the *distributional hypothesis*
- We can use *term-document* or *co-occurrence* matrices
- Word2vec vectors are trained using *skip-grams*
- Vectors can be used to do semantic arithmetic and to find similar words using *cosine similarity*
 - However, these types of vectorization *lack context*
- All *corpus-based methods* risk incorporating *harmful bias*