CSEN1076: NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

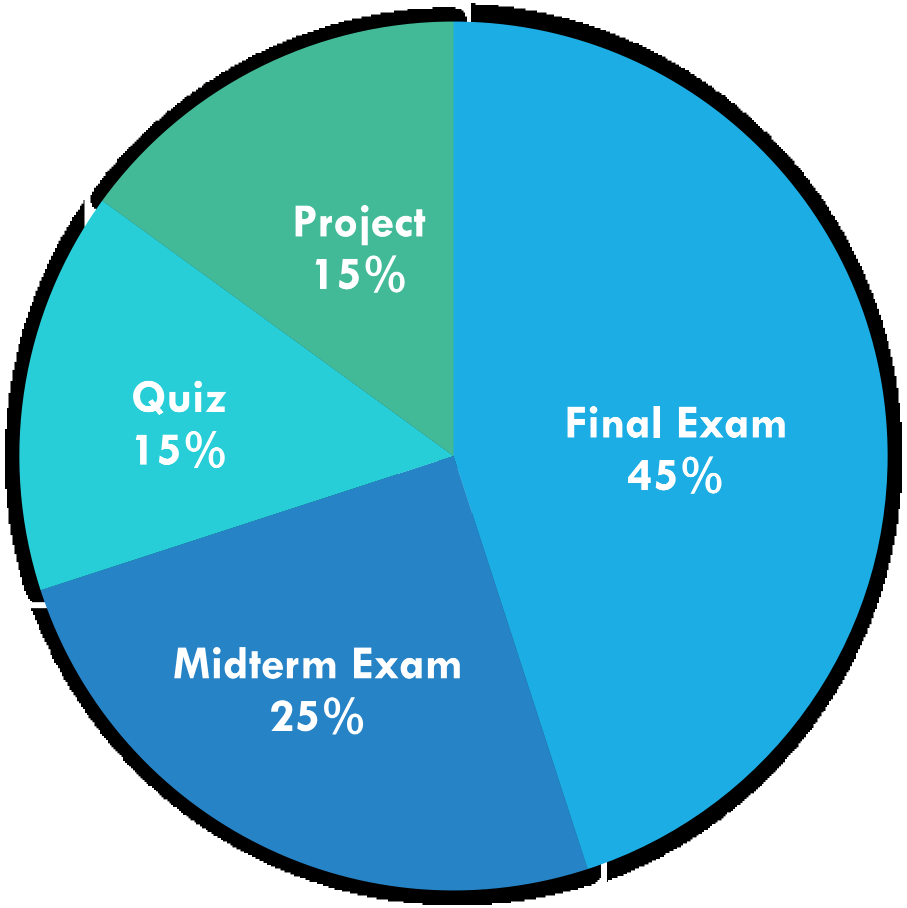
LECTURE 1 – INTRODUCTION & INFORMATION RETRIEVAL I

MERVAT ABUELKHEIR

FIRST, USEFUL INFORMATION

Course Name	Natural Language Processing and Information Retrieval
My Email	mervat.abuelkheir@guc.edu.eg
TA	Mohamed Abdelfattah
Textbooks	Introduction to Information Retrieval, Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. https://nlp.stanford.edu/IR-book/
	Natural Language Processing with Python, Steven Bird, Ewan Klein, and Edward Loper. http://www.nltk.org/book/
Optional	Speech and Language Processing, Daniel Jurafsky and James Martin, Prentice Hall, 2000.

COURSE GRADING



REQUIRED TOOLS

All of the following tools have to be installed on your laptops before your next tutorial:

- Python 3.X
- Jupyter (Python IDE)
- NLTK (Natural Language Processing Toolkit)
- Seaborn (Statistical Data Visualization)
- Sublime (Text Editor)

Don't forget to bring your laptop with you on every lab session

It would be great to learn about Python before your next tutorial

IR? NLP?

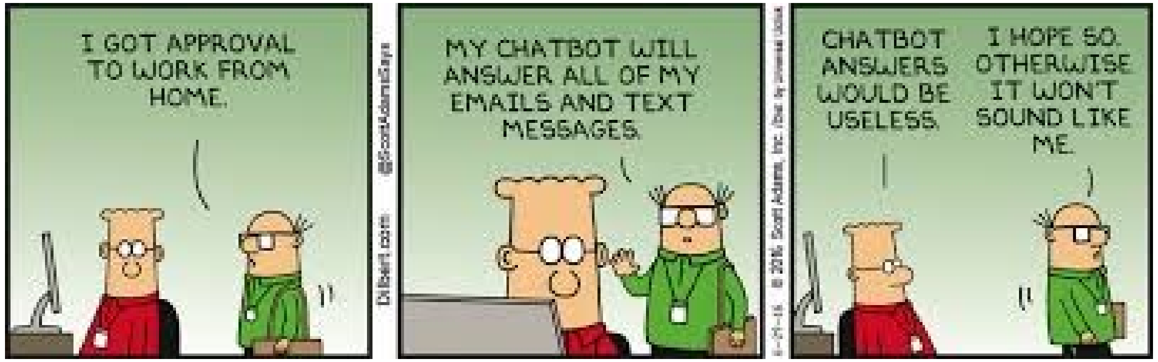
IR – Extract

Finding resources (of unstructured nature) that are relevant to an information need from a large collection of resources

"If you trust your search engine more than you trust me, maybe you should switch doctors."

NLP – Interact

Creating systems that efficiently process texts and make their information accessible to computer applications



THE KEY IS IN "NATURAL"

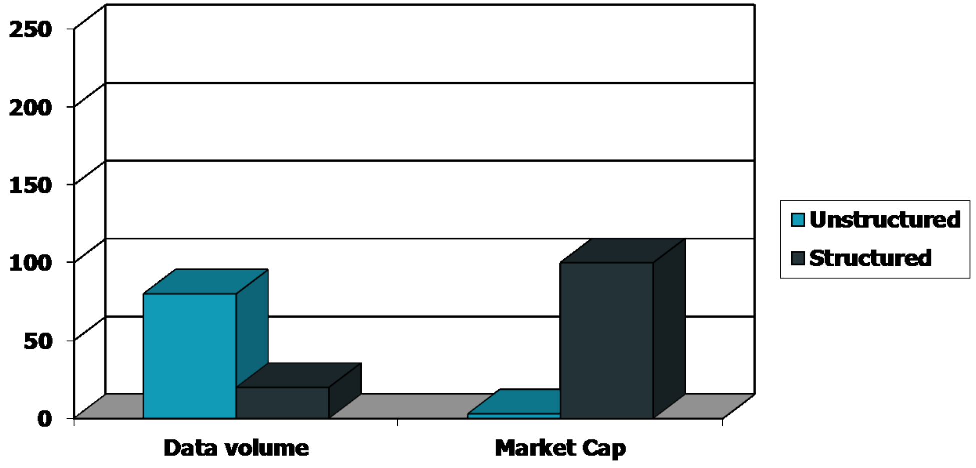
IR and NLP are not concerned with structured data

There is SQL for that!

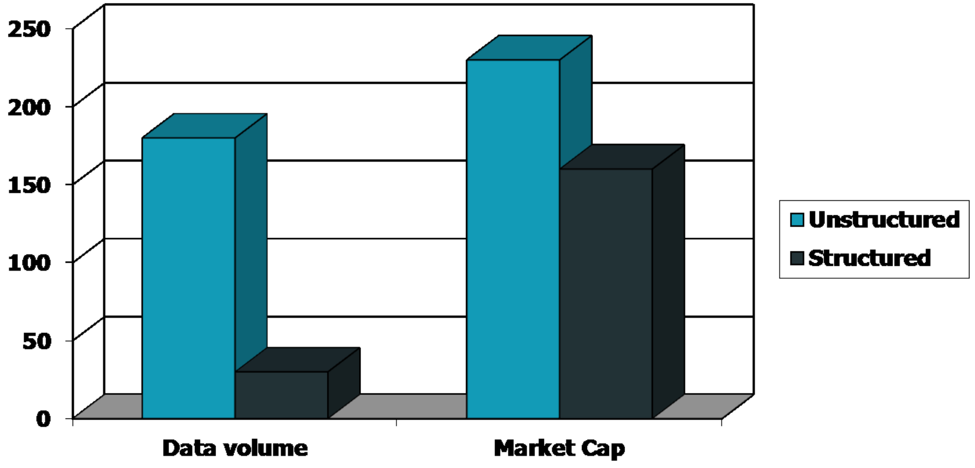
We are not looking for "exact" results, but for – hopefully – "relevant" results

THINGS HAVE CHANGED

Unstructured (text) vs. structured (database) data in the mid-nineties

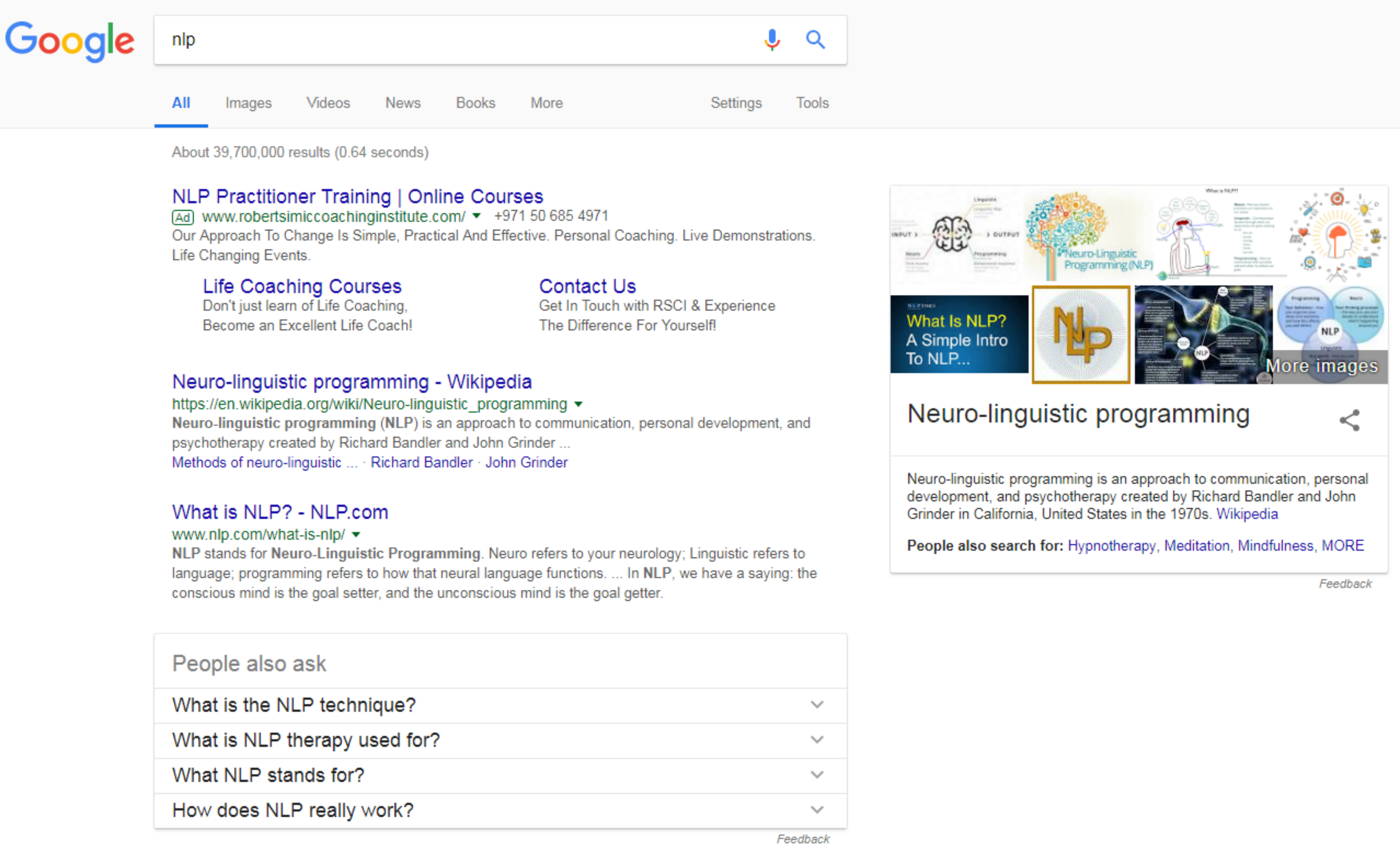


Unstructured (text) vs. structured (database) data today



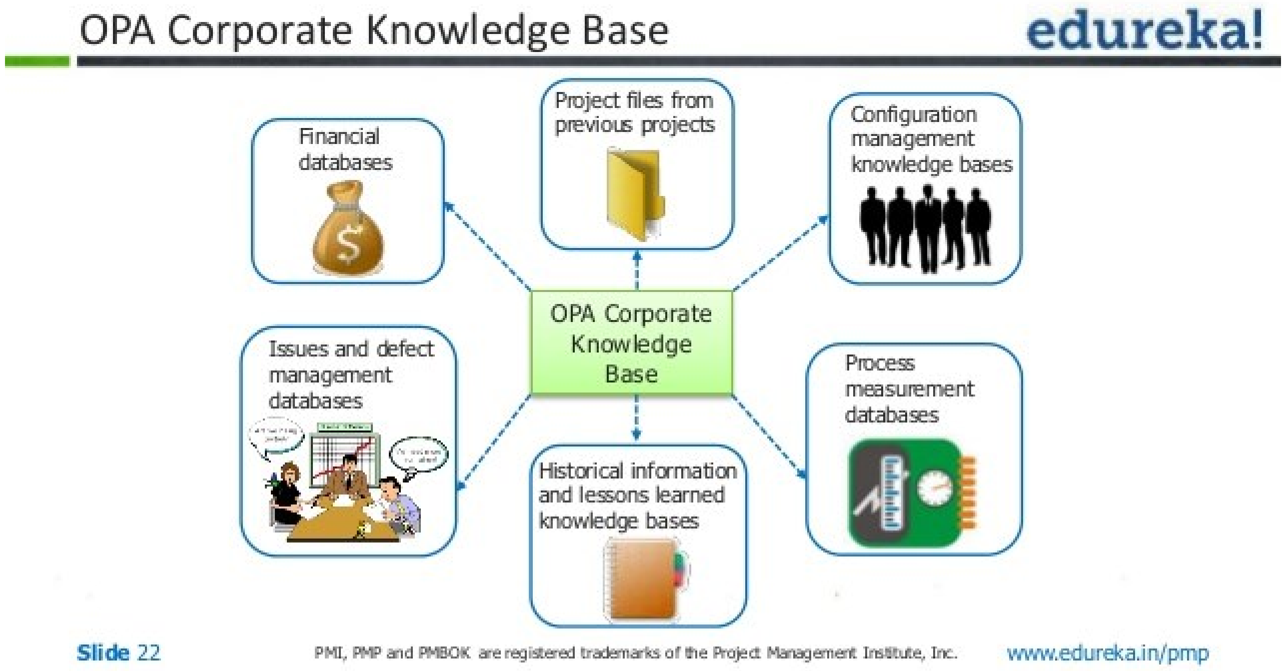
IR EXAMPLE APPLICATIONS

Web Search (duh!)



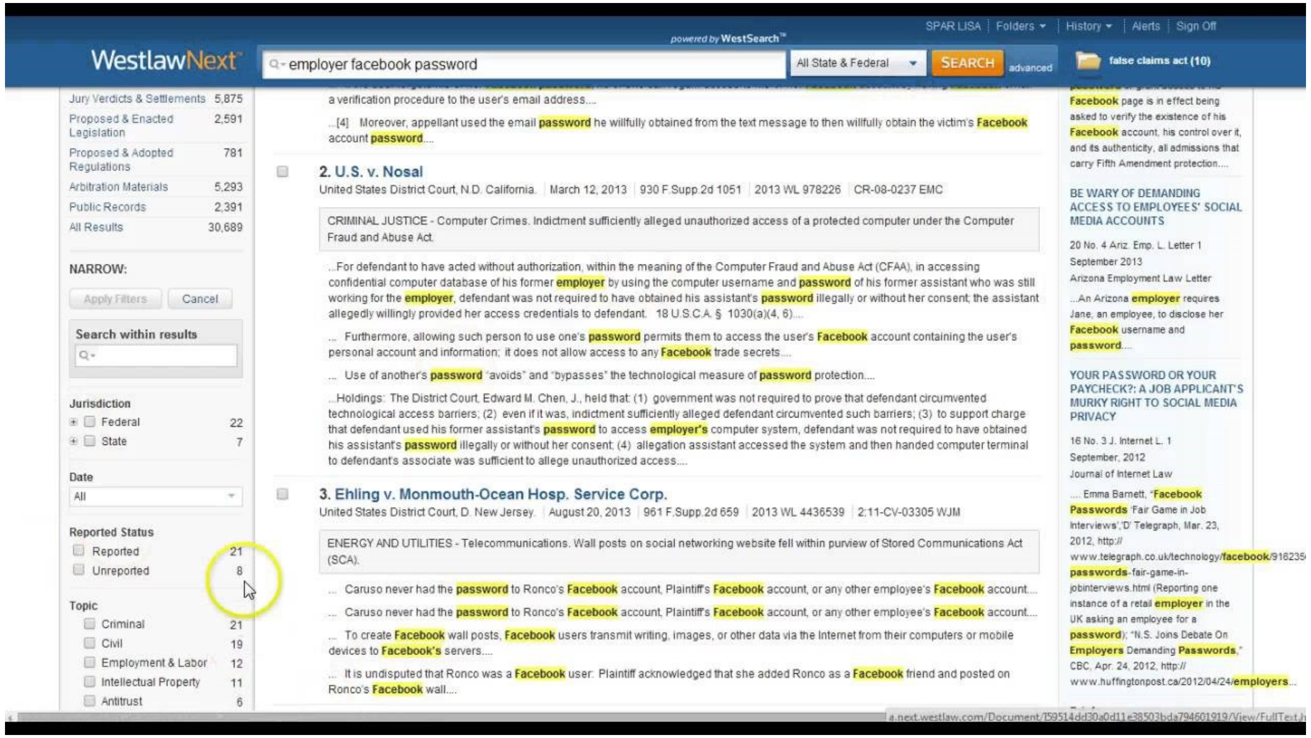
IR EXAMPLE APPLICATIONS

Corporate Knowledgebase



IR EXAMPLE APPLICATIONS

Legal Information Retrieval



NLP EXAMPLE APPLICATIONS

Information Extraction

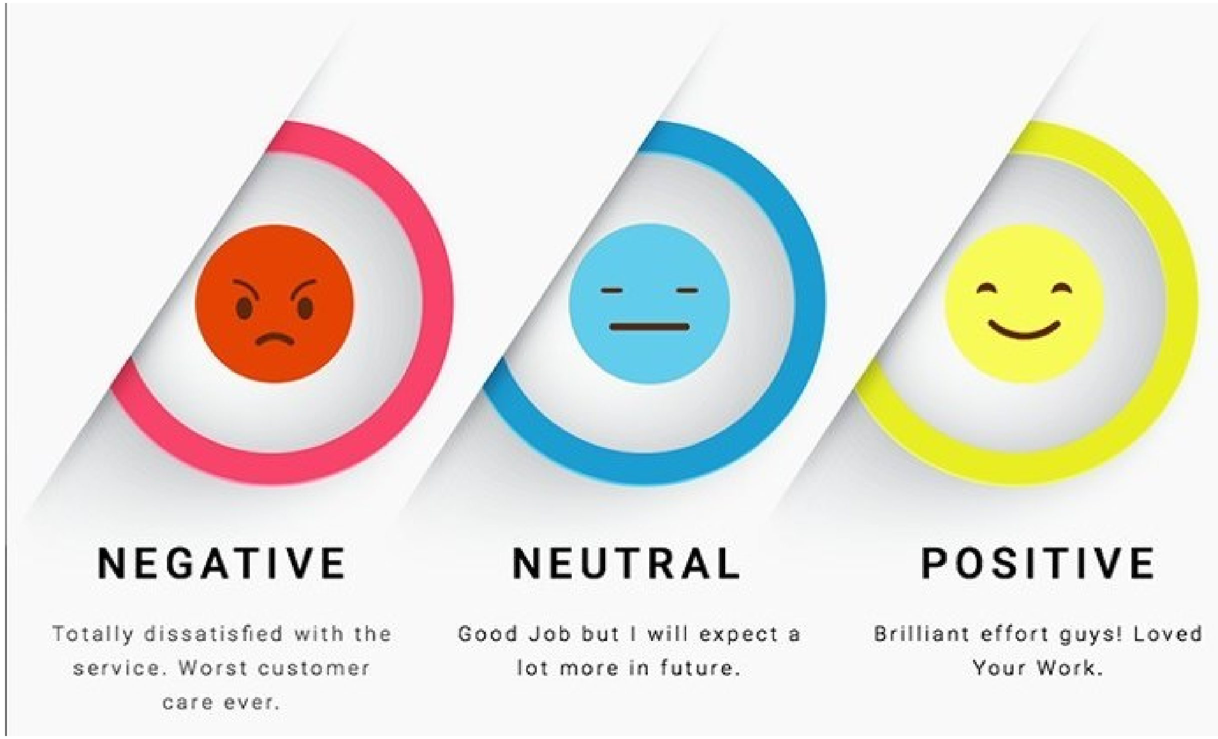
Dear Ali,
let's meet tomorrow from 1:00-2:00pm in C5-202 to discuss project status.
Yours,
Mohamed

↓ Create Calendar entry

Event:	project status meeting
Where:	C5-202
Date:	15-Sept.-2017
Start:	1:00pm
End:	2:00pm

NLP EXAMPLE APPLICATIONS

Sentiment Analysis



NLP EXAMPLE APPLICATIONS

Text Mining

Discovering Periodic Patterns in Historical News

Fabon Dzogang, Thomas Lansdall-Welfare, FindMyPast Newspaper Team, Nello Cristianini

Published: November 8, 2016 • http://dx.doi.org/10.1371/journal.pone.0165736

Article | Authors | Metrics | Comments | Related Content

Download PDF

Print | Share

CrossMark

Abstract

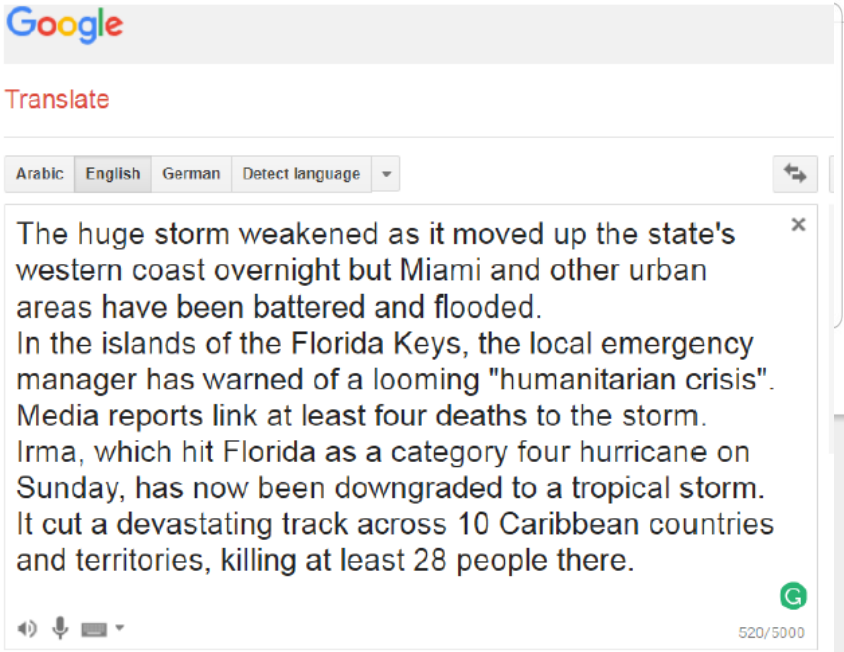
We address the problem of observing periodic changes in the behaviour of a large population, by analysing the daily contents of newspapers published in the United States and United Kingdom from 1836 to 1922. This is done by analysing the daily time series of the relative frequency of the 25K most frequent words for each country, resulting in the study of 50K time series for 31,755 days. Behaviours that are found to be strongly periodic include seasonal activities, such as hunting and harvesting. A strong connection with natural cycles is found, with a pronounced presence of fruits, vegetables, flowers and game. Periodicities dictated by religious or civil calendars are also detected and show a different wave-form than those provoked by weather. States that can be revealed include the presence of infectious disease, with clear annual peaks for fever, pneumonia and diarrhoea. Overall, 2% of the words are found to be strongly periodic, and the period most frequently found is 365 days. Comparisons between UK and US, and between modern and historical news, reveal how the fundamental cycles of life are shaped by the seasons, but also how this effect has been reduced in modern times.

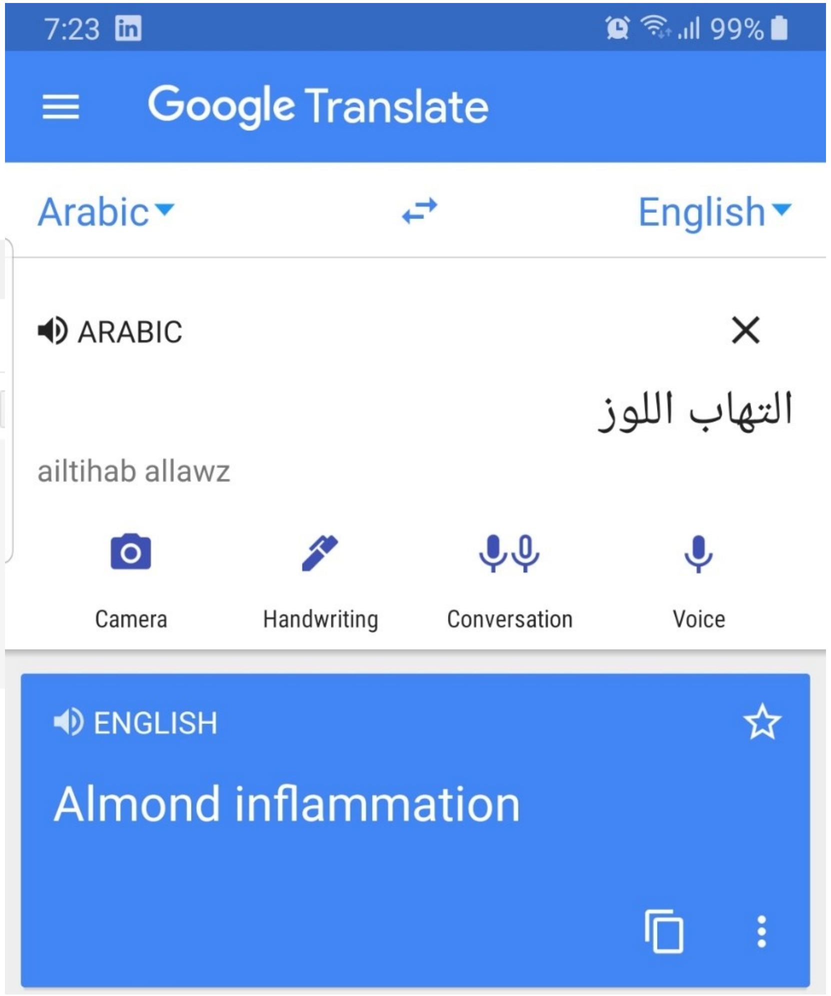
Subject Areas

- Seasons
- Fourier analysis
- Summer
- Historical geography
- Elections
- Lectures
- Weather
- United Kingdom

NLP EXAMPLE APPLICATIONS

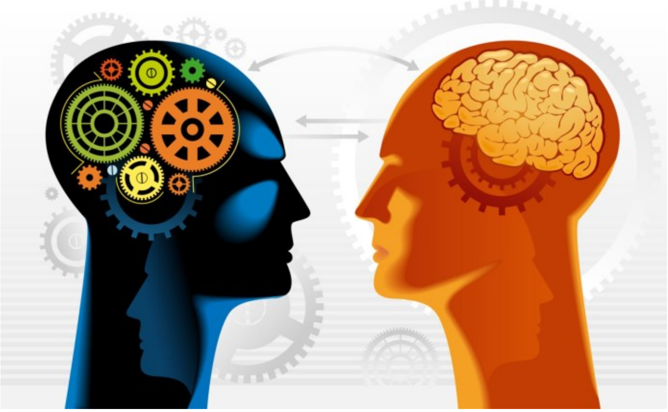
Machine Translation

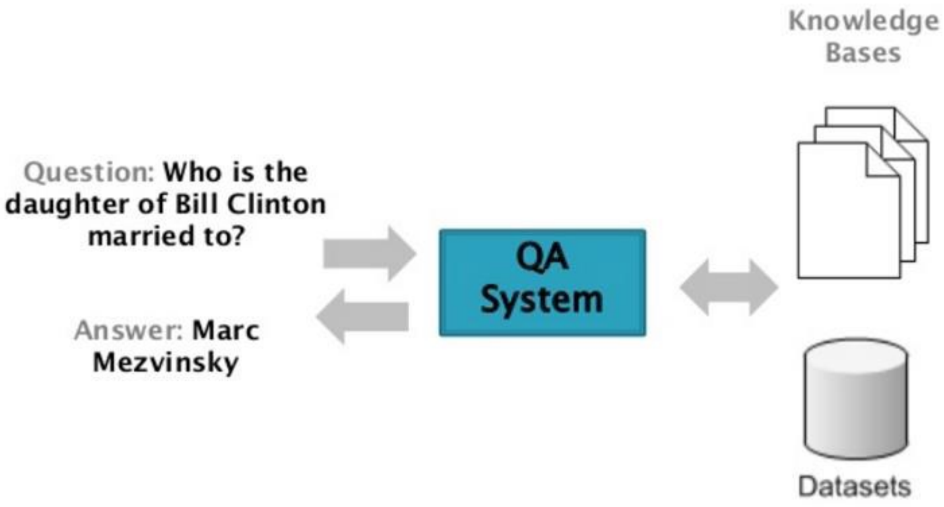




NLP EXAMPLE APPLICATIONS

Question Answering

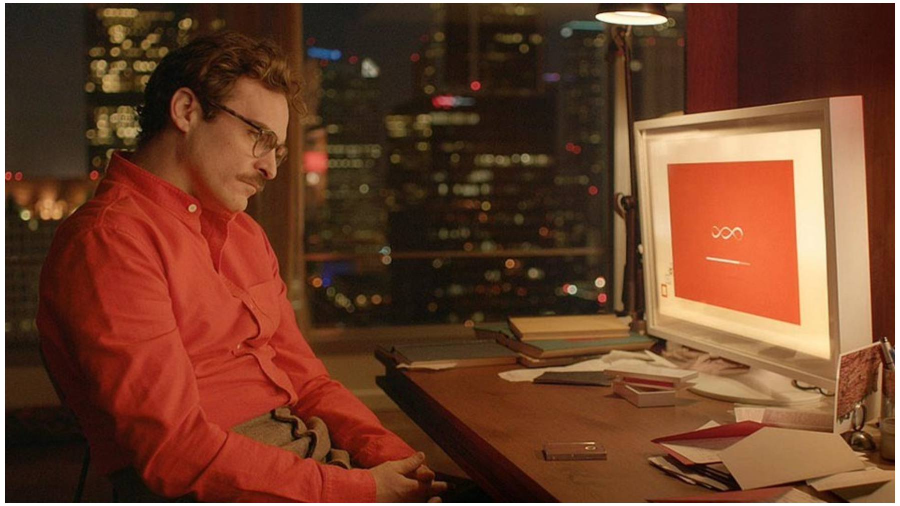




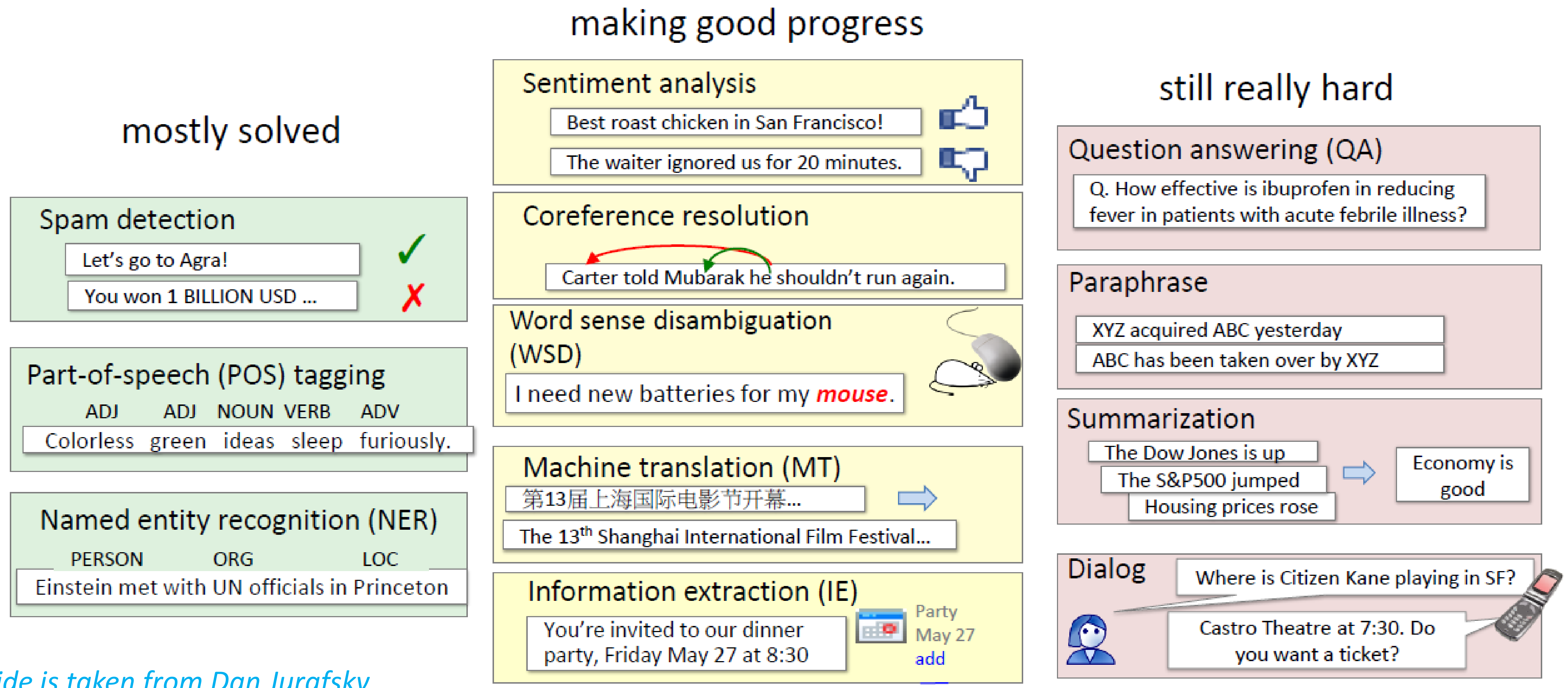
NLP EXAMPLE APPLICATIONS

ELIZA: What's on your mind today? Please let me know your opinion.
YOU: About family.
ELIZA: I see.
YOU: I think Sazae's family is good.
ELIZA: What caused you to feel that way?
YOU: Because the family members believe in themselves.
ELIZA: What other reasons might there be?
YOU: Well, I don't know.
ELIZA: I'd like to know more about your opinion.
YOU: I don't like my family so much.
ELIZA: That is quite interesting.
YOU: Yeah.
ELIZA: I'd like to know more about your opinion.
YOU: My family members aren't friendly, so I can't talk with them.
ELIZA: I see.
YOU: Do you have family?
ELIZA: Why do you think that?
YOU: I don't know who you are.
ELIZA: Why do you ask?
YOU: I'm sorry.
ELIZA: That's okay. Please continue.

Conversational AI



NLP TECHNOLOGY



**This slide is taken from Dan Jurafsky*

WHY IS NLP HARD?

Ambiguity …at all levels of analysis

- Ambiguity is resolved with **knowledge**: world knowledge, contextual knowledge, statistical knowledge

Phonetics and phonology:

- Interpreting a speech signal: "I scream is delicious" vs. "ice cream is delicious”

Morphology: unionized = union + ized? un + ionized?

Syntax: Squad helps dog bite victim. (Does Squad help dog or victim?)

Semantics: Jack invited Mary to the ball. (ball could be “dance” or a “decoration ball”)

Discourse: Merck & Co. formed a joint venture with Ache Group, of Brazil. It will be called Prodome Ltd. (“it” refers to what?)

BUILDING A COMPUTER THAT ‘UNDERSTANDS’ TEXT: FUNDAMENTAL ELEMENTS OF THE NLP PIPELINE

**These slides are taken from Hongning Wang @ UVa*

TOKENIZATION/SEGMENTATION (BASIC)

Split text into words and sentences

- Task: what is the most likely segmentation /tokenization?

Challenges – What to do about:

- **Sentence/word boundaries?**
- **Negation?**
- **Punctuations? Numbers?**
- **Shortened text?**
- **Punctuations that are integral part of names?**
- **Markup symbols in HTML? In social platforms?**
- **Tokenizing other languages?**

There was an earthquake near D.C. I’ve even felt it in Philadelphia, New York, etc.

There + was + an + earthquake + near + D.C.

I + ve + even + felt + it + in + Philadelphia, + $\boxed{\text{New + York,}}$ + etc.

لا تدخل الآن، فالقاعة تعج بهم.

NORMALIZATION (BASIC)

Transform text into a single canonical form

- Task: what is the most likely normalization?

There were 3 résumés posted to our recruitment system in Dec. 5th. That is sooooo few!!

There + were + three + resumes + posted + to + our + recruitment + system + in + December + fifth + . + ▽ + That + is + so + few + !

لَا تَدْخُلِ الْآنَ، فَالْقَاعَةُ تَعُجُّ بِهِمْ.

Challenges – What to do about:

- **Non-standard text?**
- **Task-dependency?**
- **Multiple languages?**

STOPWORDS REMOVAL (BASIC)

Remove words that do not contribute to meaning

- Task: what is the most likely set of words that should be ignored?

Challenges – What to do about:

- **Negation and context?**
- **Desired dimensionality?**
- **Semantics?**

There was an earthquake near D.C. I’ve even felt it in Philadelphia, New York, etc.

~~There~~ + ~~was~~ + ~~an~~ + earthquake + ~~near~~ + D.C.

~~I~~ + ~~ve~~ + ~~even~~ + felt + ~~it~~ + ~~in~~ + Philadelphia, + ~~New~~ + York, + ~~etc~~.

~~لا~~ تدخل ~~الآن~~، فالقاعة تعج بهم.

STEMMING AND LEMMATIZATION (BASIC)

Reduce word to its root/lemma

- Task: what is the most likely root for a word?
- Suffix-stripping (or more general affix strepping) algorithms
- Lemmatization algorithms
- n-gram analysis

In our last meeting, we agreed it is better that meeting the client would be next week to discuss requirements.

meet , agree , good , meet , will , require

تعتمد بلدان العالم على استخدام أنظمة الحاسب

Challenges – What to do about:

- **Affixes and semantics?**
- **Language?**

PART-OF-SPEECH TAGGING (MEDIUM)

Marking up a word in a text (corpus) as corresponding to a particular part of speech

- Task: what is the most likely tag sequence

Challenges – What to do about:

- **Ambiguity of meaning?**
- **Many-to-one tags?**
- **Language?**

A + dog + is + chasing + a + boy + on + the + playground

A	+	dog	+ is	+ chasing	+ a	+ boy	+ on	+ the	+ playground
Det		Noun	Aux	Verb	Det	Noun	Prep	Det	Noun

لا تدخل الآن، فالقاعة تعج بهم.

NAMED ENTITY RECOGNITION (ADVANCED)

Determine text mapping to proper names

- Task: what is the most likely mapping?

Challenges – What to do about:

- **Types of NEs?**
- **Annotation labor?**
- **Concept hierarchies?**
- **Scope and context?**
- **Language?**

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Organization, Location, Person

SUMMARY: INITIAL STAGES OF TEXT PROCESSING

Tokenization

- Cut character sequence into word tokens
  - Consider white spaces, punctuation marks, hyphens, apostrophe, etc.
  - Deal with ***"John's"***, ***a state-of-the-art solution***

Case Folding

- Reduce all letters to lower case. In other tasks like machine translation, case is important

Normalization

- Map text and query term to same form
  - You want ***U.S.A.*** and ***USA*** to match, but not ***C.A.T.*** and ***cat***!

Stemming

- We may wish different forms of a root to match
  - ***authorize***, ***authorization***

Stop words

- We may omit very common words (or not)
  - ***the, a, to, of***

IR BASICS – TERM-DOCUMENT INCIDENCE MATRICES

UNSTRUCTURED DATA IN 1620

Example query: Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?

One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar***, then strip out lines containing ***Calpurnia***?

Why is that not the answer?

- Slow (for large corpora)
- NOT ***Calpurnia*** is non-trivial
- Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
- Ranked retrieval (best documents to return)

No. of Shakespeare plays: ***37***
Avg. no. of words per play: ***22,000***

TERM-DOCUMENT INCIDENCE MATRIX

All distinct words (terms) in **all plays**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
⋮						

Query:* Brutus** *AND* ***Caesar *BUT NOT* ***Calpurnia***

1 if document contains word, 0 otherwise

INCIDENCE VECTORS

So we have a 0/1 **vector** for each term

To answer query: take the vectors for ***Brutus***, ***Caesar*** and ***Calpurnia*** (complemented) ➔ bitwise AND

- 110100 AND
- 110111 AND
- 101111 =
- **100100**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

ANSWERS TO QUERY

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,

When Antony found Julius ***Caesar*** dead,

He cried almost to roaring; and he wept

When at Philippi he found ***Brutus*** slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius ***Caesar*** I was killed i' the

Capitol; ***Brutus*** killed me.

BIGGER COLLECTIONS

Consider N = 1 million documents, each with about 1000 words

Avg 6 bytes/word including spaces/punctuation

- 6GB of data in the documents

Say there are M = 500K distinct **terms** among these

CAN'T BUILD THE MATRIX

500K x 1M matrix has half-a-trillion 0's and 1's

But it has no more than one billion 1's ⇐ Why?

- matrix is extremely sparse

What's a better representation?

- We only record the 1 positions

IR BASICS – THE INVERTED INDEX

INVERTED INDEX

For each term t, we must store a list of all documents that contain t

Identify each document by a *docID*, a document serial number

Can we use fixed-size arrays for this?

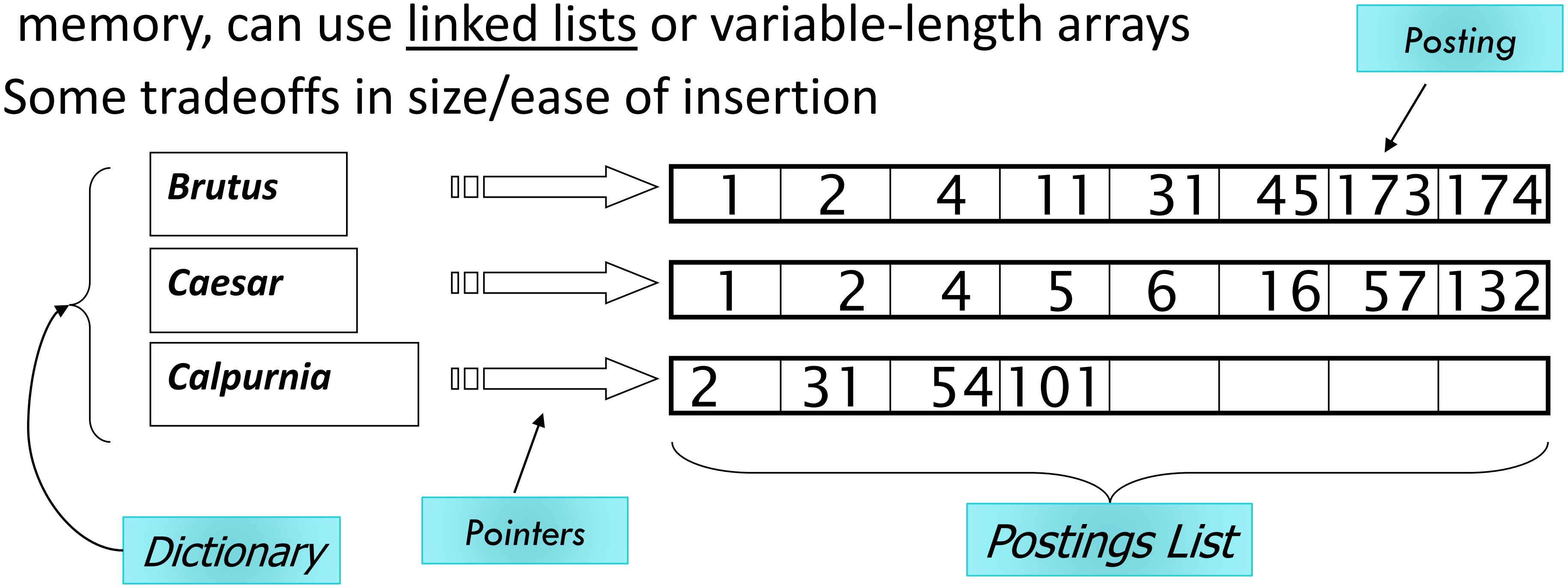
Brutus	1	2	4	11	31	45	173	174
Caesar	1	2	4	5	6	16	57	132
Calpurnia	2	31	54	101				

What happens if the word ***Caesar*** is added to document 14?

INVERTED INDEX

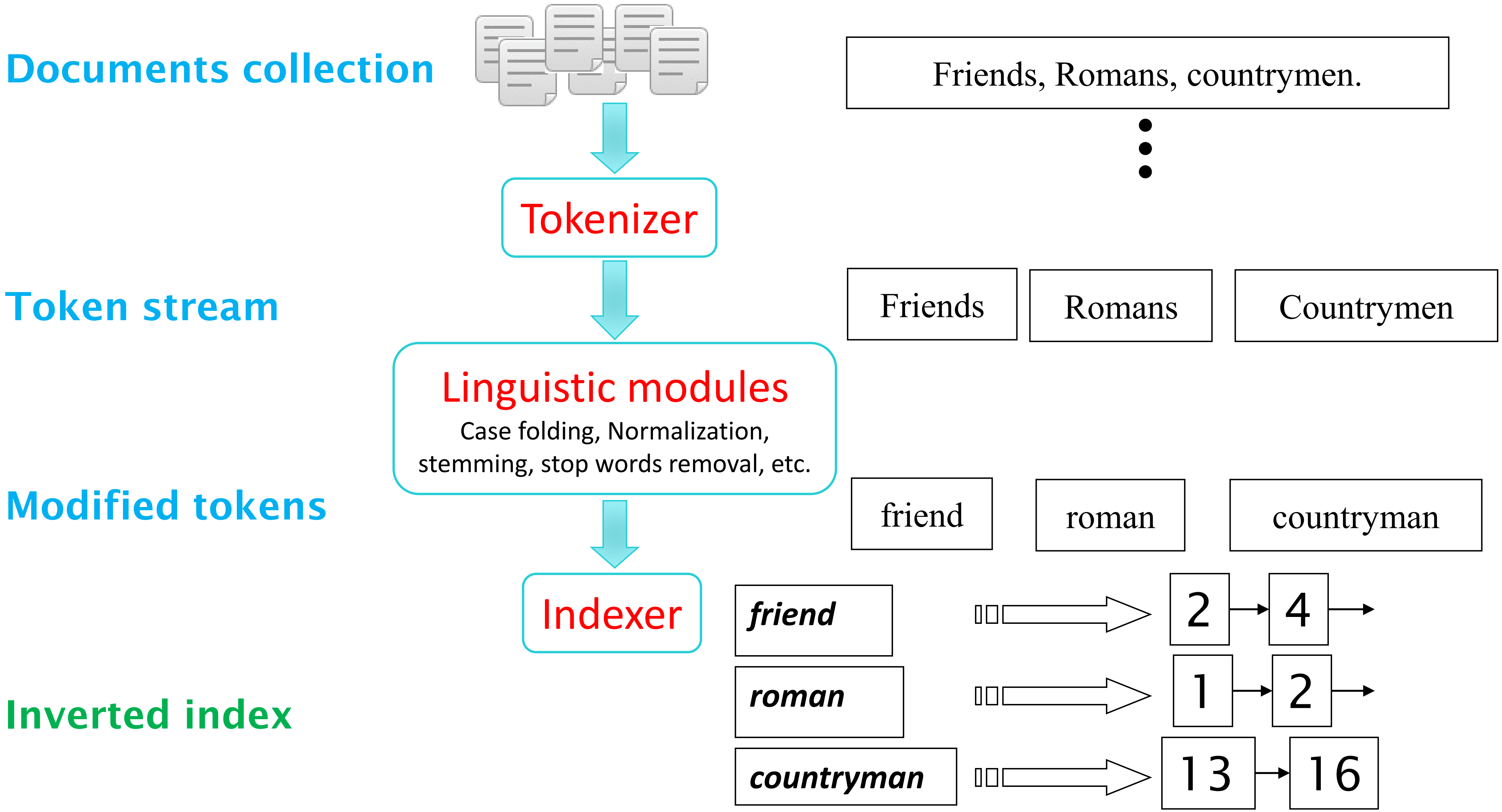
We need variable-size **postings lists**

- On disk, a continuous run of postings is normal and best
- In memory, can use linked lists or variable-length arrays
  - Some tradeoffs in size/ease of insertion



Sorted by *docID* (more later on why)

INVERTED INDEX CONSTRUCTION



INDEXER STEPS: TOKEN SEQUENCE

Sequence of (Modified token, Document ID) pairs

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

INDEXER STEPS: **SORT**

Sort by terms

- And then *docID*

Core indexing step

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

→

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

INDEXER STEPS: DICTIONARY & POSTINGS

- Multiple term entries in a single document are merged
- Split into Dictionary and Postings
- Doc. frequency information is added

Why frequency?
Will discuss later

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

→

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Inverted index

HOW DO WE RESPOND TO A QUERY?

How do we process a query?

- What kinds of queries can we process?

Queries containing one term are pretty easy

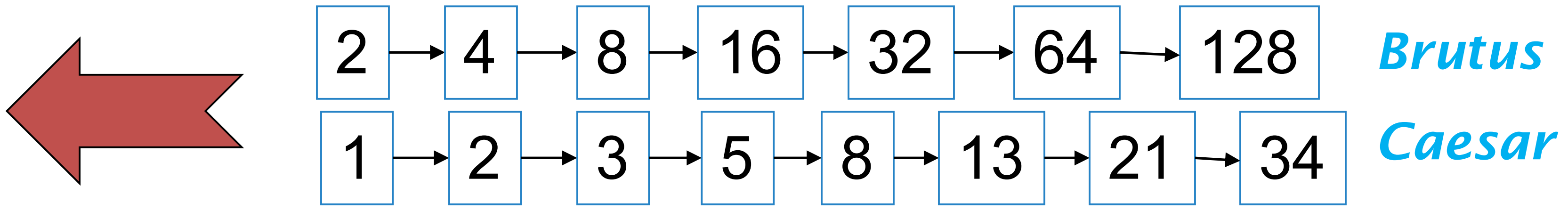
- Just get all postings linked to the term in the dictionary

QUERY PROCESSING: AND

Consider processing the query:

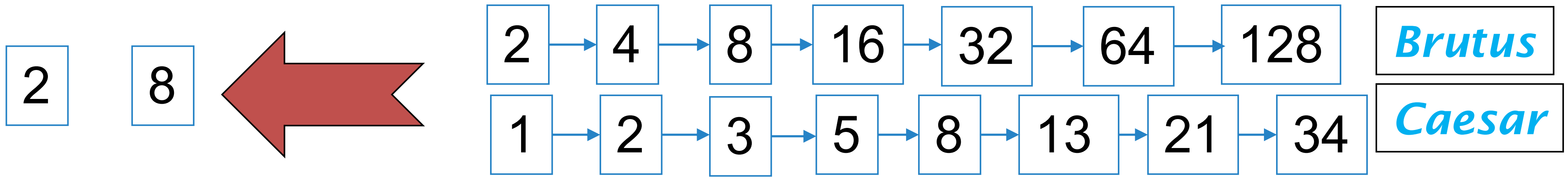
Brutus *AND* ***Caesar***

- Locate ***Brutus*** in the Dictionary;
  - Retrieve its postings
- Locate ***Caesar*** in the Dictionary;
  - Retrieve its postings
- "Merge" the two postings (intersect the document sets):



THE MERGE

Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are x and y, the merge takes $O(x+y)$ operations

Crucial: postings sorted by *docID*

INTERSECTING TWO POSTINGS LISTS (A "MERGE" ALGORITHM)

```
INTERSECT(p1, p2)
  answer ← ⟨ ⟩
  while p1 ≠ NIL and p2 ≠ NIL
  do if docID(p1) = docID(p2)
        then ADD(answer, docID(p1))
             p1 ← next(p1)
             p2 ← next(p2)
        else if docID(p1) < docID(p2)
                then p1 ← next(p1)
                else p2 ← next(p2)
  return answer
```

QUERY OPTIMIZATION: MORE ANDS!

Consider a query that is an AND of n terms, $n > 2$

- For each of the terms, get its postings list, then AND them together

Example query: ***Brutus*** AND ***Calpurnia*** AND ***Caesar***

- What is the best order for processing this query?

QUERY OPTIMIZATION: MORE ANDS!

Example query: ***Brutus*** AND ***Calpurnia*** AND ***Caesar***

Simple and effective optimization: Process in order of increasing frequency

- Start with the shortest postings list, then keep cutting further
- In this example, first ***Caesar***, then ***Calpurnia***, then ***Brutus***

Brutus → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 14

Calpurnia → 2 → 31 → 54 → 101

Caesar → 5 → 31

OPTIMIZED INTERSECTION ALGORITHM FOR CONJUNCTIVE QUERIES

```
INTERSECT(⟨t_1, ..., t_n⟩)
terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
result ← postings(first(terms))
terms ← rest(terms)
while terms ≠ NIL and result ≠ NIL
do result ← INTERSECT(result, postings(first(terms)))
    terms ← rest(terms)
return result
```

MORE GENERAL OPTIMIZATION

Example query: (***madding*** or ***crowd***) and (***ignoble*** or ***strife***)

- Get frequencies for all terms
- Estimate the size of each or by the sum of its frequencies (conservative)
- Process in increasing order of or sizes

EXERCISE

France → 1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 13 → 14 → 15

Paris → 2 → 6 → 10 → 12 → 14

Lear → 12 → 15

Compute hit list for ((*paris* AND NOT *france*) OR *lear*)

NEXT TIME

Phrase Queries

Vector Space Model

Term Weighting

REFERENCES

This lecture is heavily relying on the following courses:

- CS 276 / LING 286: Information Retrieval and Web Search, Stanford University
- Natural Language Processing Lecture Slides from the Stanford Coursera course by Dan Jurafsky and Christopher Manning