Applied Natural Language Processing

Dr Jeff Mitchell, University of Sussex

Autumn 2025

Word sequences

Previously

- Part-of-speech tagging
  - HMMs
  - Viterbi Algorithm
- Information Extraction
  - Named entity recognition (NER)
  - Named entity linking
  - Relation extraction

This time

- Question answering (QA)
- Information retrieval
  - Document retrieval
  - The inverted index
  - Ranked retrieval
  - Query expansion
- Reading-comprehension based QA
- Knowledge-based QA

Computers that can answer questions

- No longer science fiction / fantasy
- In 2011, IBM's "Watson" beat humans to win the quiz show "Jeopardy"

Question answering

- Focus on **factoid question answering**
  - questions that can be answered with simple facts expressed in short texts
  - answers are right or wrong
  - many (but not all) are concerned with relationships between named entities

Where was **Isaac Newton** born?

What is the average age for the onset of autism?

When was the **University of Sussex** founded?

Paradigms in question answering

Information Retrieval (IR-based)

- use *information retrieval* techniques to find **relevant documents** and passages
  - also referred to as *document retrieval*
- use **reading comprehension** algorithms to draw an answer from relevant spans of text

Knowledge-based

- build a semantic or **logical representation** of the query
- use logical meaning representations to query a **database of facts**
  - *information extraction* techniques might be used to build or update database

Information retrieval

- Obtaining *information resources* relevant to an *information need* from a *large collection* of *information resources*
  - *information resources*: unstructured data, e.g., text documents
  - *large collection*: e.g., the web
  - *information need*: expressed as a **query**
    - a set of **query terms** (e.g., a set of words or phrases)
    - an expression in some special purpose **query language**
    - an expression in **natural language**

Issues

- **accuracy**
  - finding what should be found
- **efficiency**
  - finding what should be found reasonably quickly
  - particular issue for web-scale IR
- **relevance ranking**
  - presenting users with information in a manageable way
  - how often do you look beyond the first page of Google results?

Relevant documents

- Goal is to identify **relevant** documents
  - could be documents containing the query terms
  - could be documents satisfying the query expression
- **Boolean retrieval**
  - no attempt to rank retrieved documents by relevance
- **Ranked retrieval**
  - retrieved documents ranked by relevance

Boolean information retrieval

- Document represented as a set of keywords
- Query expressed as Boolean combination of keywords
  - keywords connected by logical operators AND, OR, NOT
  - brackets used to indicate scope

(((*bank* OR *saving*) AND *account*) AND (NOT *river*))

- Output: set of relevant documents
  - no partial matches
  - no ranking

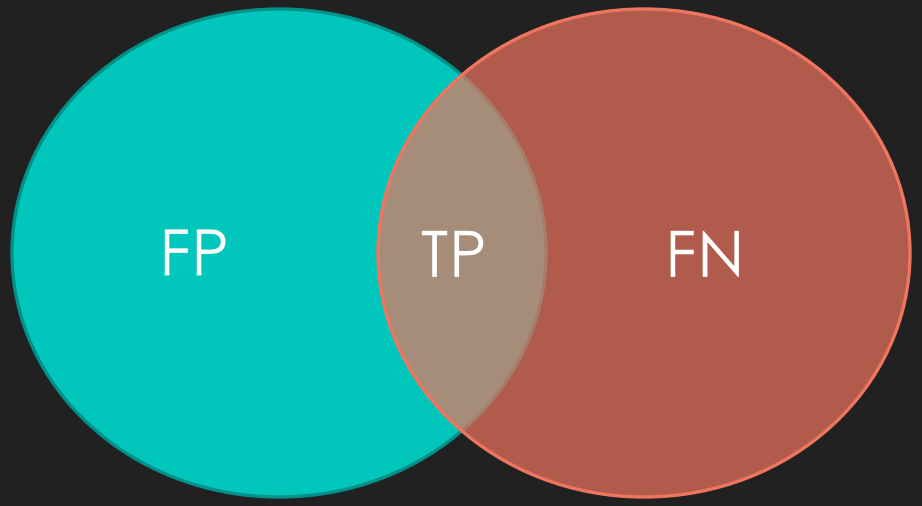
Canonicalisation of terms

- Apply standard canonicalisations of terms (see week 2)
  - case normalisation
  - number normalisation
  - stopword removal
  - stemming / lemmatisation
- Must apply to both the documents and the queries
- Reduces term vocabulary
  - improves efficiency
  - often improves performance too

Measuring performance

How do the *retrieved* documents **compare** to the actually *relevant* documents?

Measuring performance



It can be difficult to measure recall. Why?

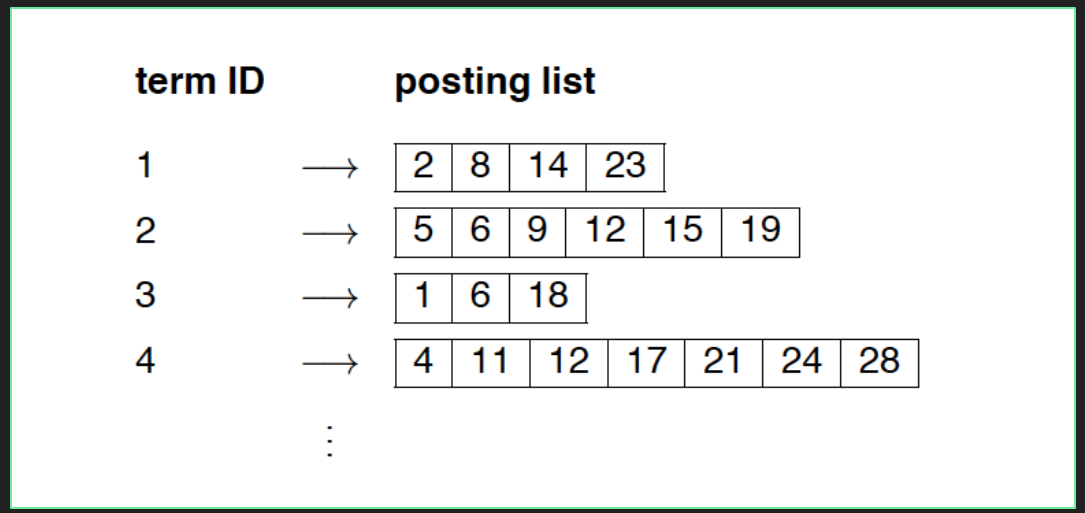
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = \frac{2PR}{P + R}$$

Indexing documents

- Need to know which documents contain query terms
- Very inefficient to search all documents every time
- Build an **inverted index**:
  - Index documents by the potential query terms they contain



Retrieval with an inverted index

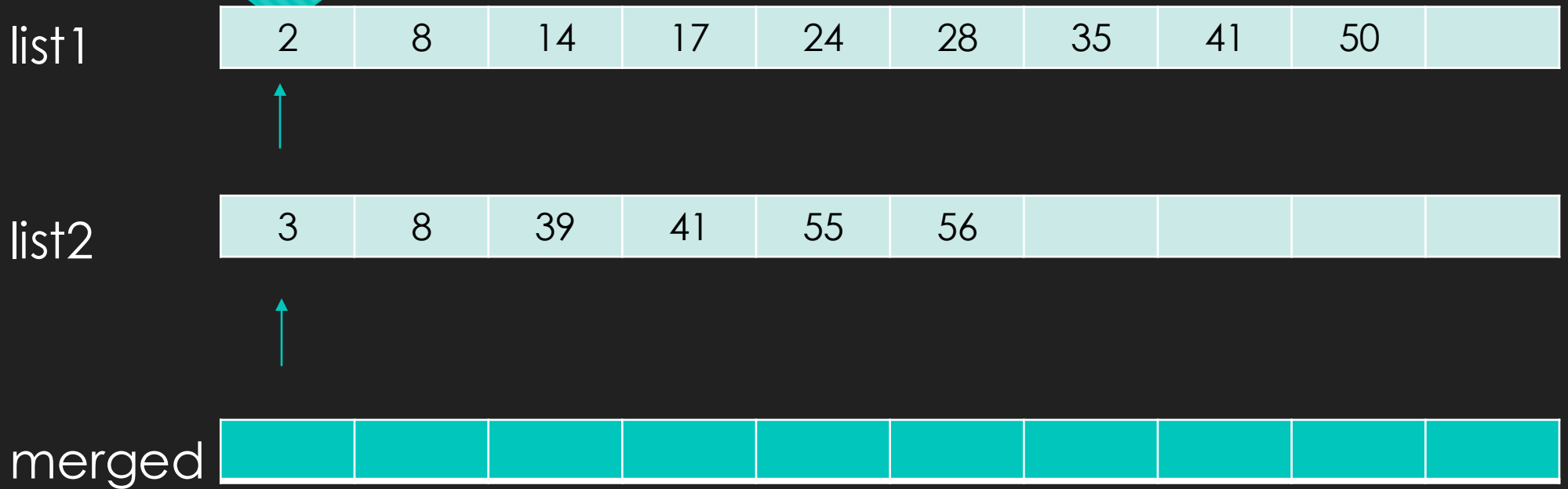
- Query is a collection of terms connected by logical operators
- Look up posting lists for each term in the query in inverted index
- Start with one posting list and compute documents in intersection / union successively with other lists
- Finally return documents after all posting lists have been considered

Intersection algorithm

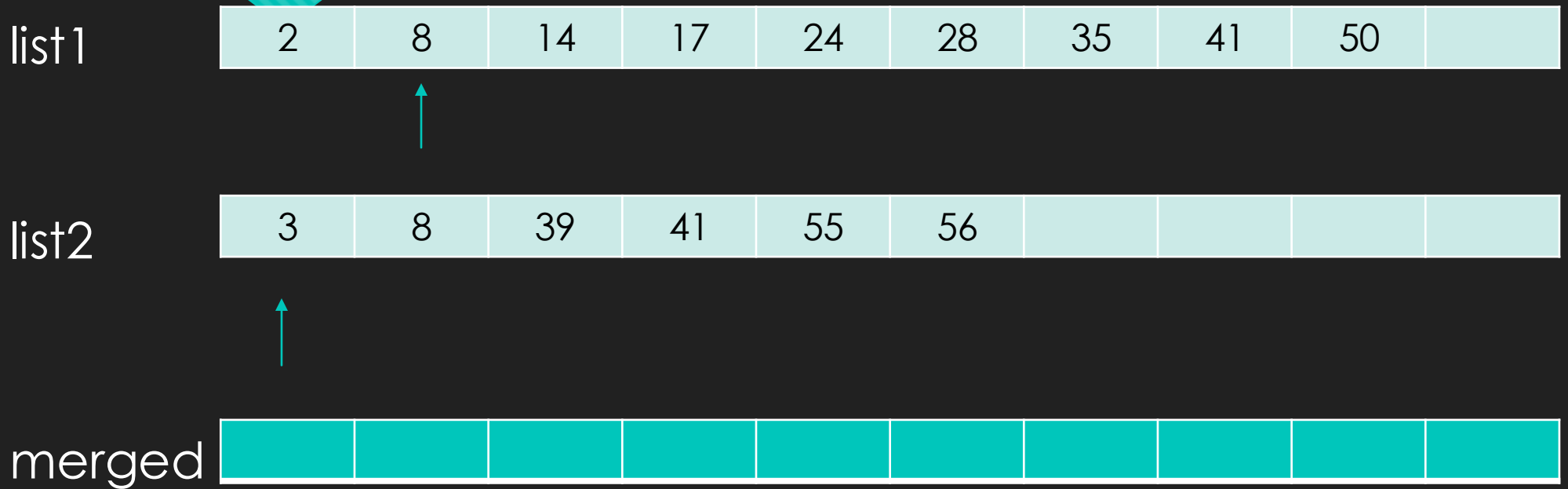
```
def merge(list1,list2):
    '''
    function to intersect 2 sorted lists
    '''
    pointer1=0
    pointer2=0
    merged=[]
    while pointer1<len(list1) and pointer2<len(list2):
        if list1[pointer1]==list2[pointer2]:
            merged.append(list1[pointer1])
            pointer1+=1
            pointer2+=1

        elif list1[pointer1]<list2[pointer2]:
            pointer1+=1
        else:
            pointer2+=1

    return merged
```

- assume both posting lists are sorted by document id
- set pointers to the start of each list
- while still elements to consider in each list
- compare items pointed at
- if the same, then include in intersection and advance both pointers
- otherwise advance pointer on list with lower id

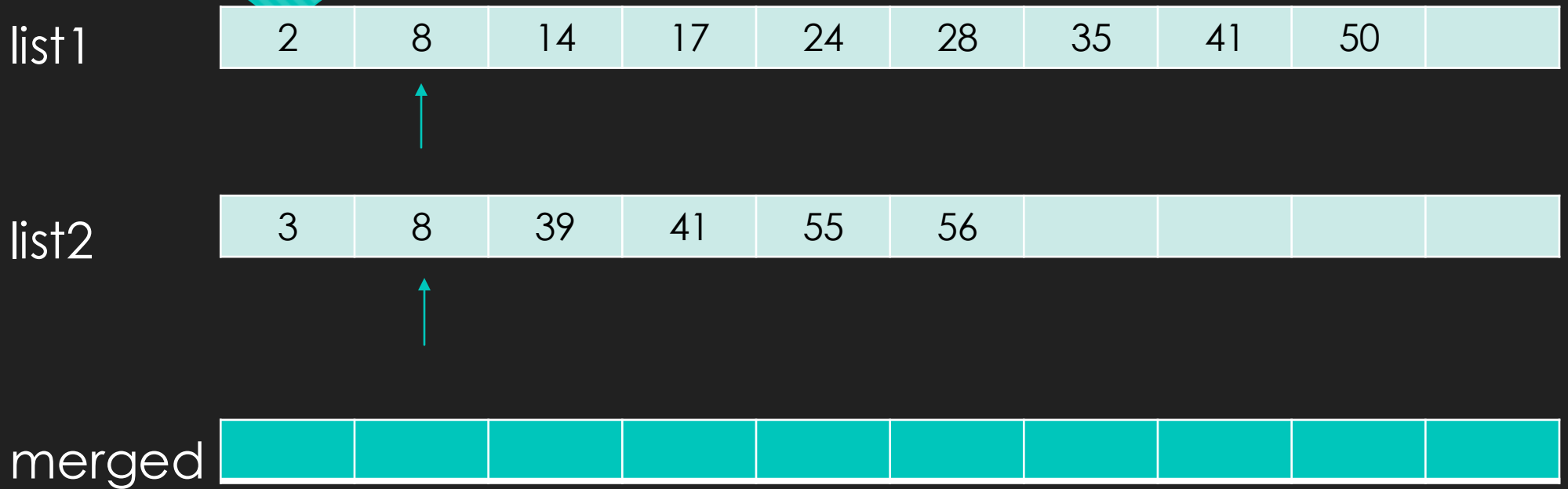
Example



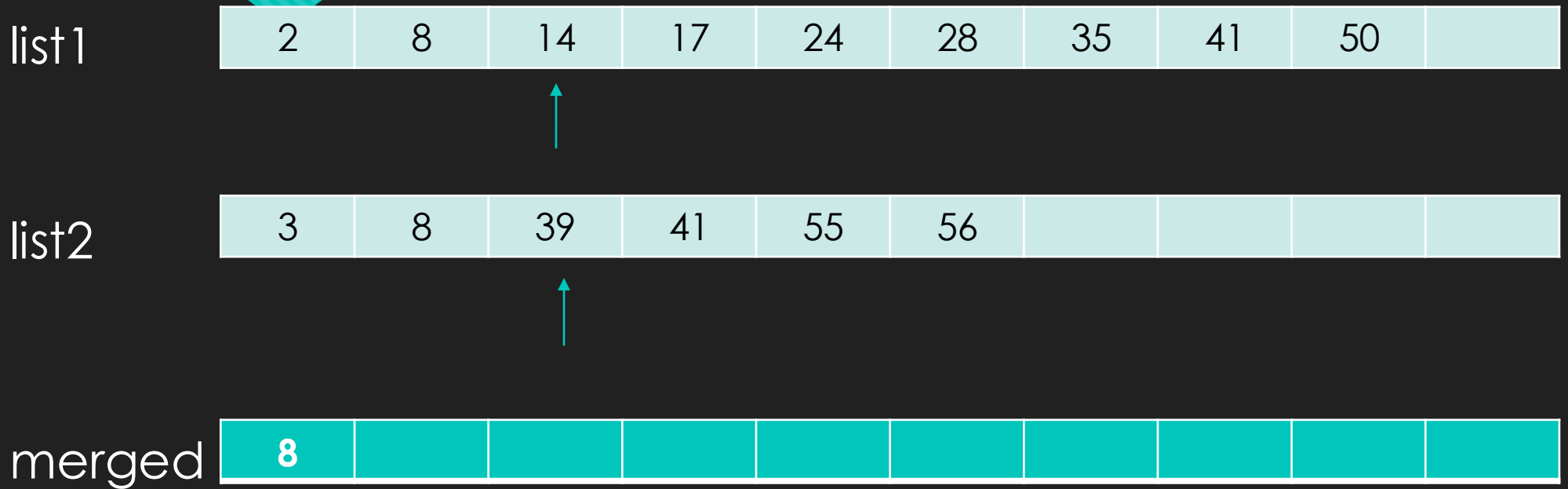
Example



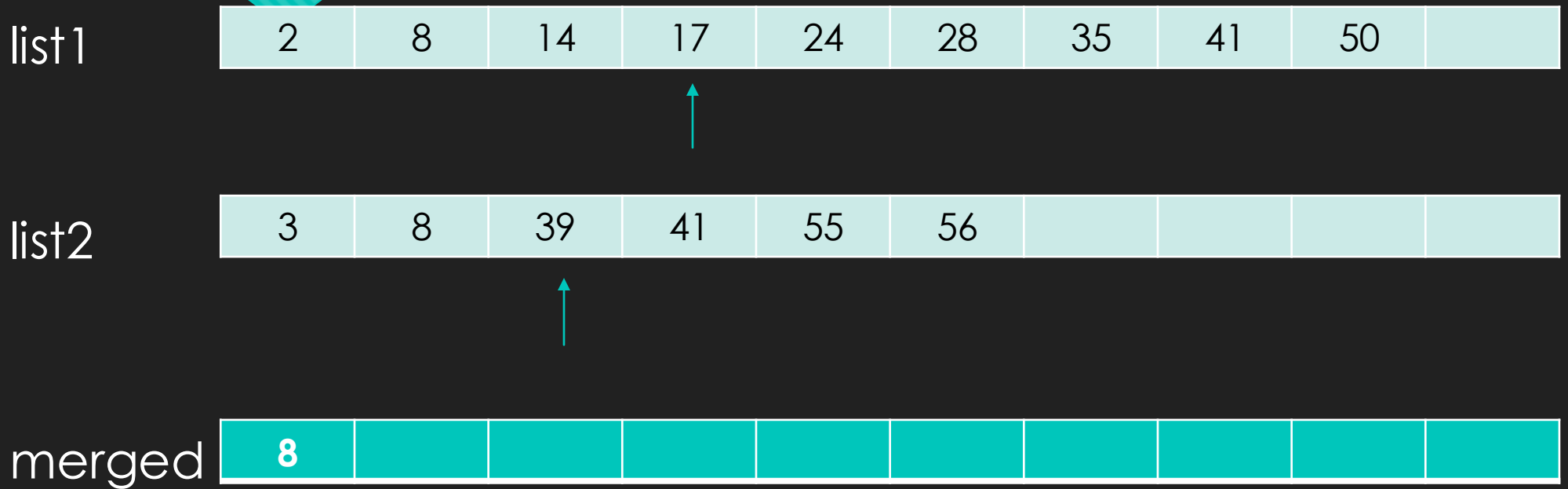
Example



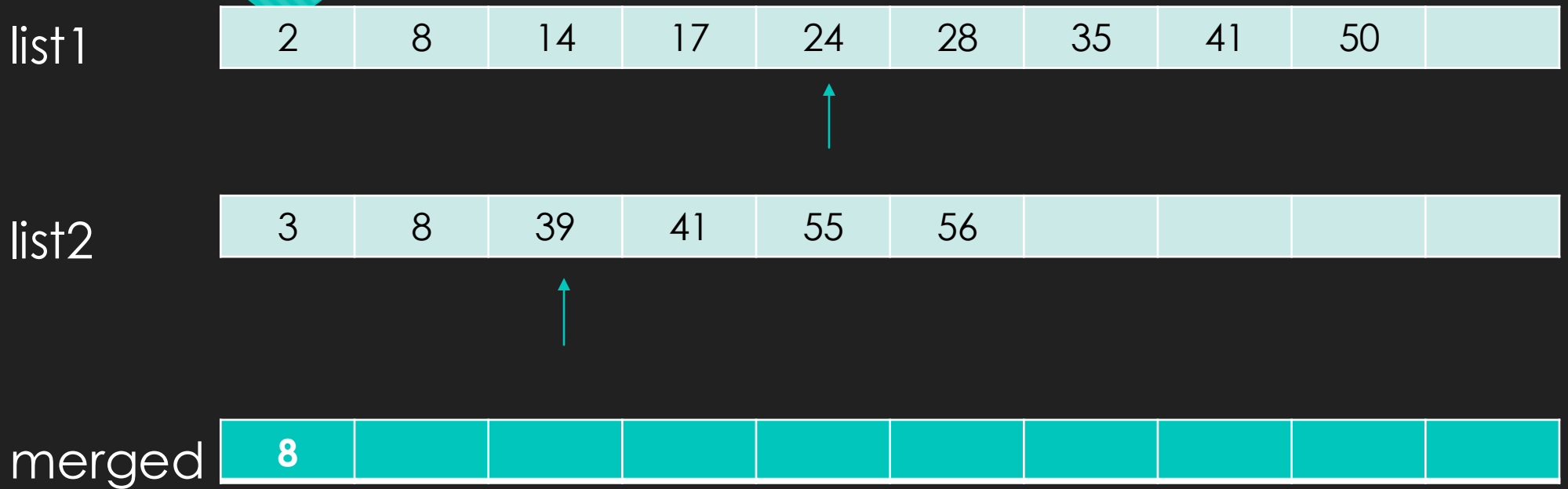
Example



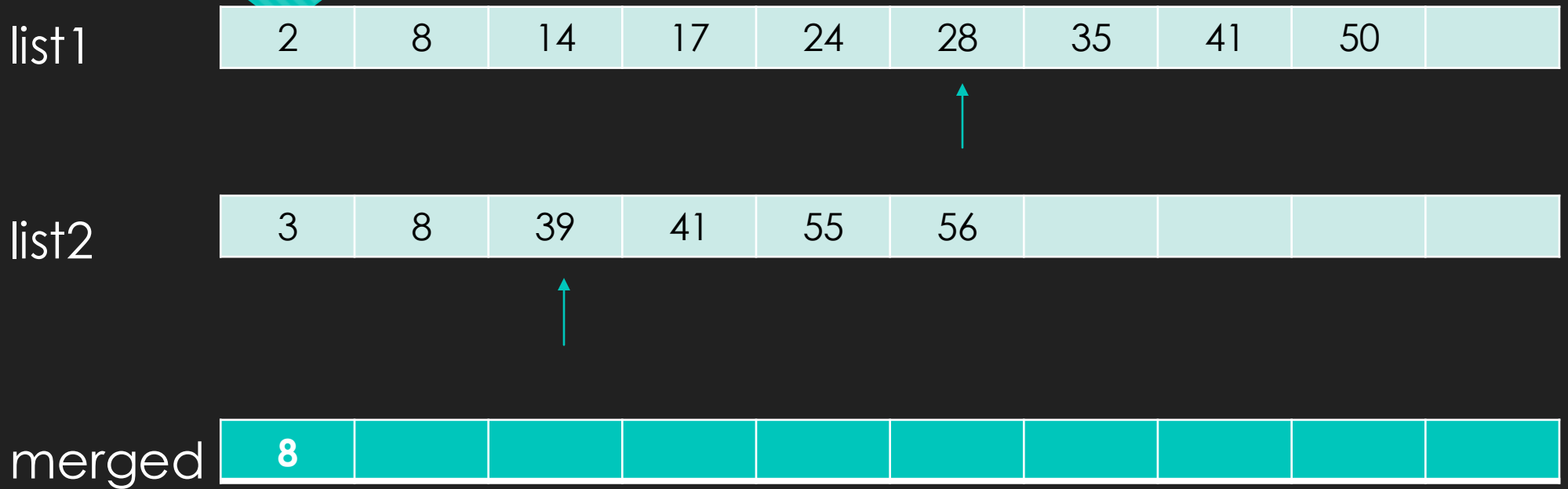
Example



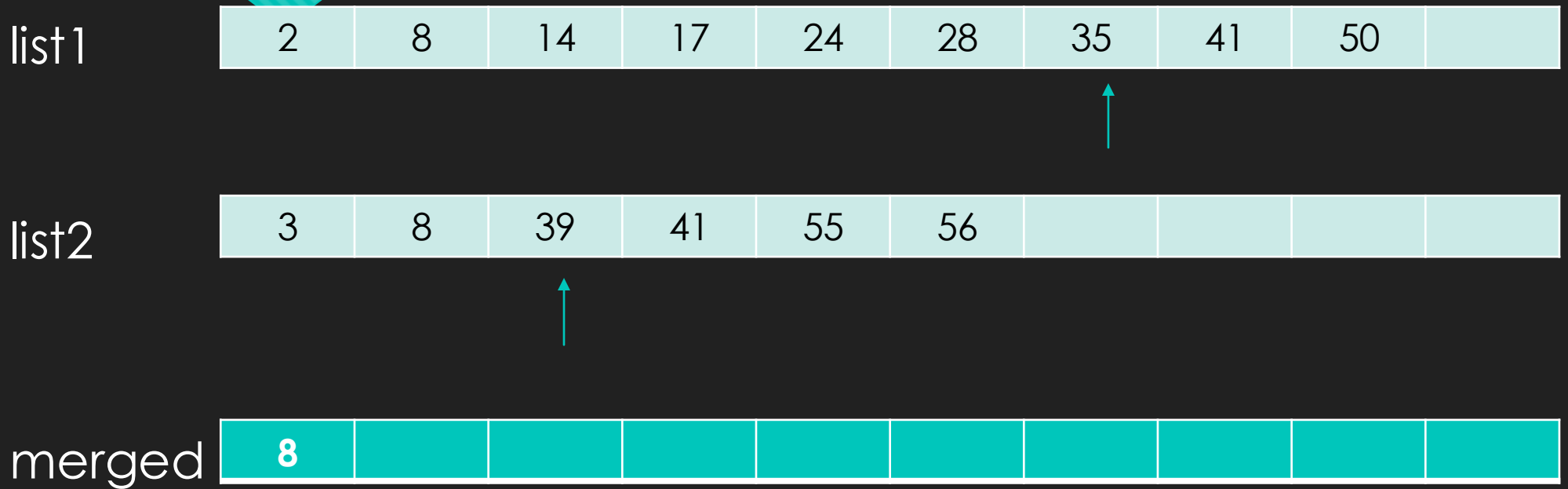
Example



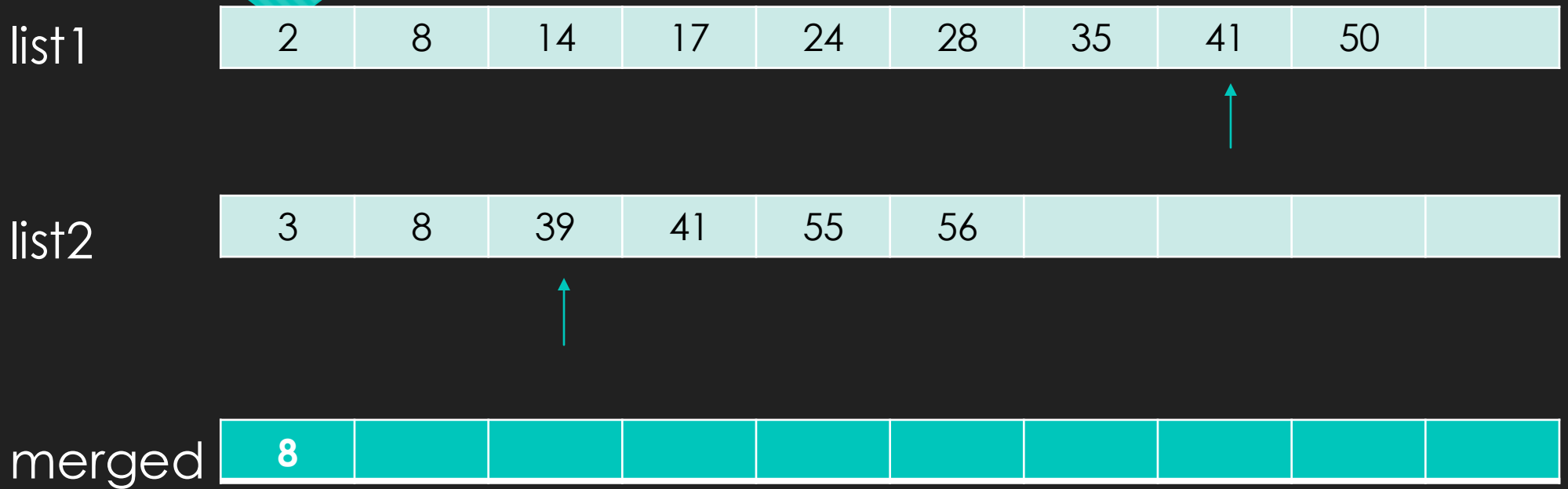
Example



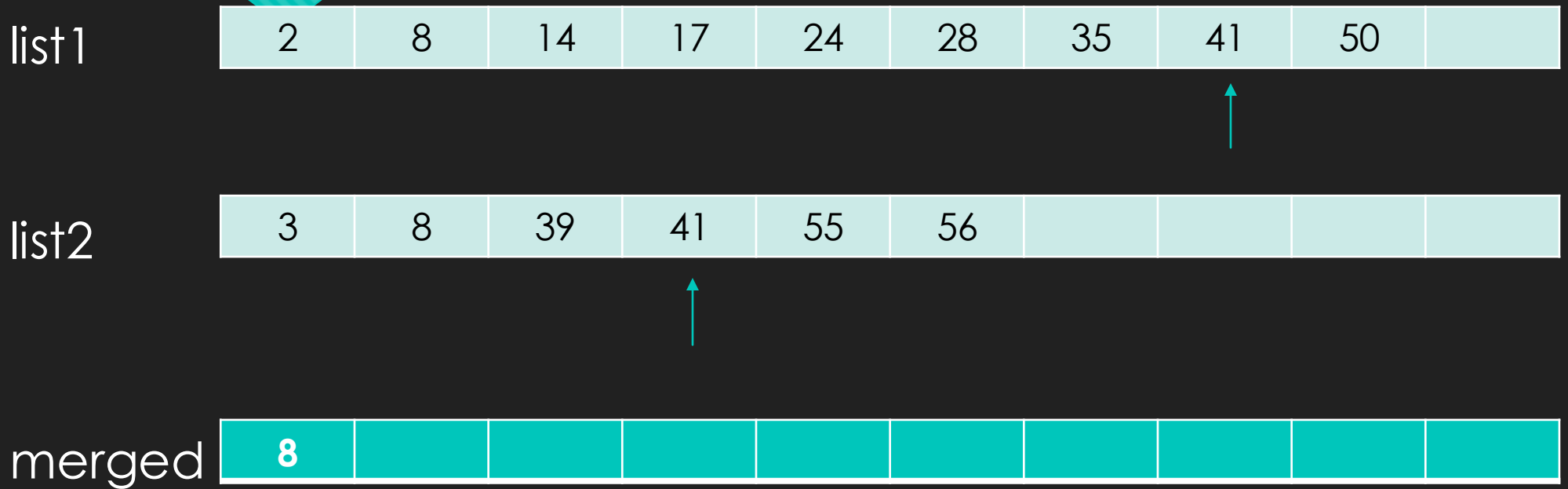
Example



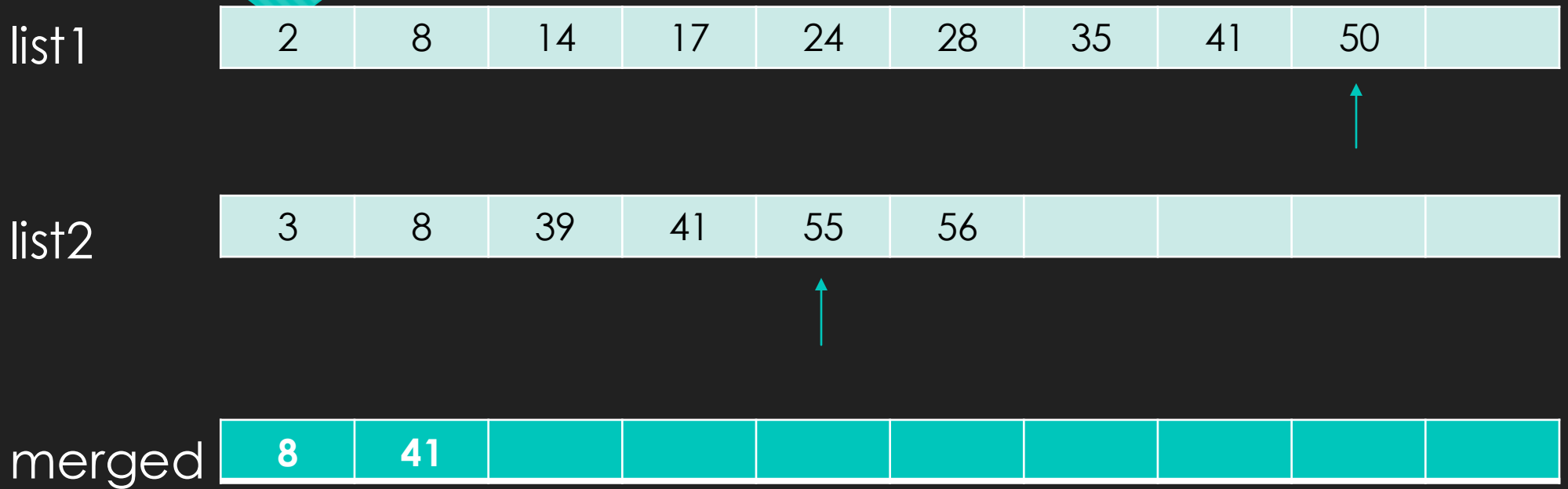
Example



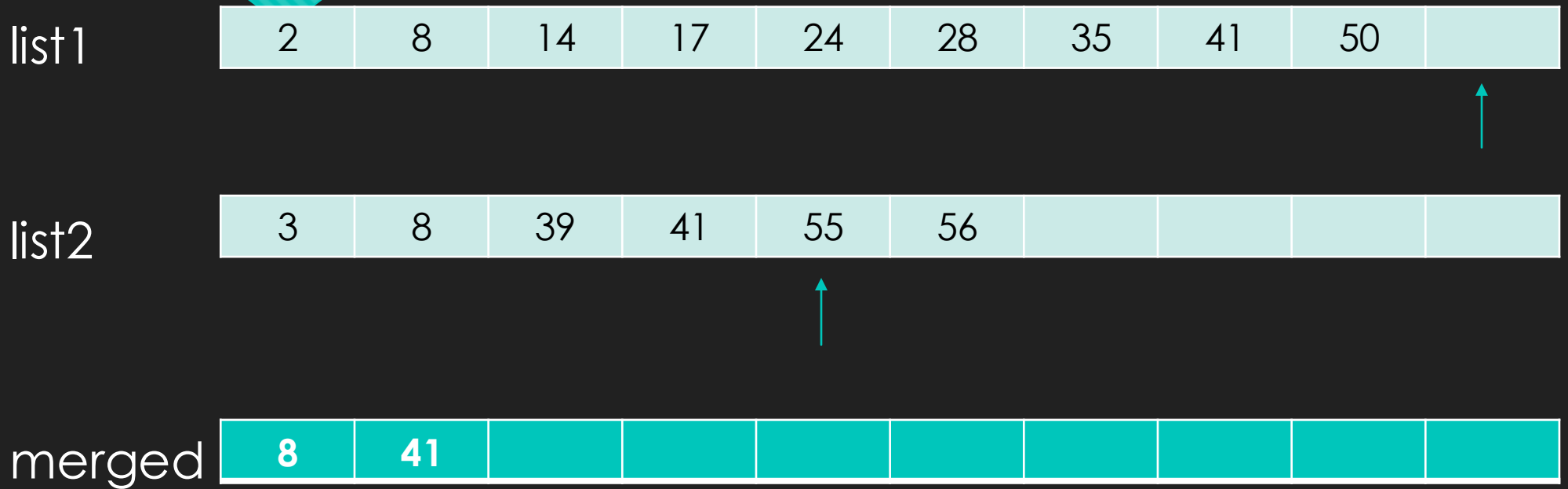
Example



Example



Example



More implementation details

- If more than two posting lists
  - repeat with each list in turn
  - most efficient to start with shortest posting list
- Algorithm linear in total length of posting lists
- Algorithms for OR and NOT are analogous to AND
- Inverted index can be implemented efficiently using a python dictionary
  - uses hashing to find posting list associated with term
- Extensions
  - bi-word (phrase) inverted index
  - positional index
    - posting lists include position on document so that adjacency within document can be checked.

Union by merge

How would you update the algorithm for intersecting two sorted lists to find the union of two sorted lists?

```
def merge(list1,list2):
    '''
    function to intersect 2 sorted lists
    '''
    pointer1=0
    pointer2=0
    merged=[]
    while pointer1<len(list1) and pointer2<len(list2):
        if list1[pointer1]==list2[pointer2]:
            merged.append(list1[pointer1])
            pointer1+=1
            pointer2+=1

        elif list1[pointer1]<list2[pointer2]:
            pointer1+=1
        else:
            pointer2+=1

    return merged
```

A NOT B (by merge)

How would you update the algorithm for intersecting two sorted lists to find A NOT B for two sorted lists?

```
def merge(list1,list2):
    '''
    function to intersect 2 sorted lists
    '''
    pointer1=0
    pointer2=0
    merged=[]
    while pointer1<len(list1) and pointer2<len(list2):
        if list1[pointer1]==list2[pointer2]:
            merged.append(list1[pointer1])
            pointer1+=1
            pointer2+=1

        elif list1[pointer1]<list2[pointer2]:
            pointer1+=1
        else:
            pointer2+=1

    return merged
```

Ranked retrieval

Boolean retrieval often useless

- too many matching documents
- need to be ranked by degree of relevance

less common query terms occur more frequently in the document

query terms occur more frequently in document

Relevancy scoring

- tf-idf scores can be calculated for each document *d* and query *q*
- Once set of all possibly relevant documents obtained, score them:
  - as a sum of tf-idf scores
  - as a product of tf-idf scores
  - as the cosine of the vector representation of *d* with *q*

$$\text{Relevance}(d) = \sum_{w \epsilon q} \text{tfidf}(d, w)$$

$$\text{Relevance}(d) = \prod_{w \epsilon q} \text{tfidf}(d, w)$$

$$\text{Relevance}(d) = \frac{d.q}{\sqrt{d.d \times q.q}}$$

Query expansion

- Exact match between words in document and query not necessary
  - normalisation
- Spelling errors, spelling alternative, synonyms and near synonyms?
- Technical language vs. informal language
- Use a thesaurus to add "**nearest neighbours**" as alternatives in query
  - Hand-crafted thesaurus (e.g., WordNet) vs distributional thesaurus?
- In practice:
  - Increases recall but often damages precision

Part 2

IR-based and knowledge-based question answering

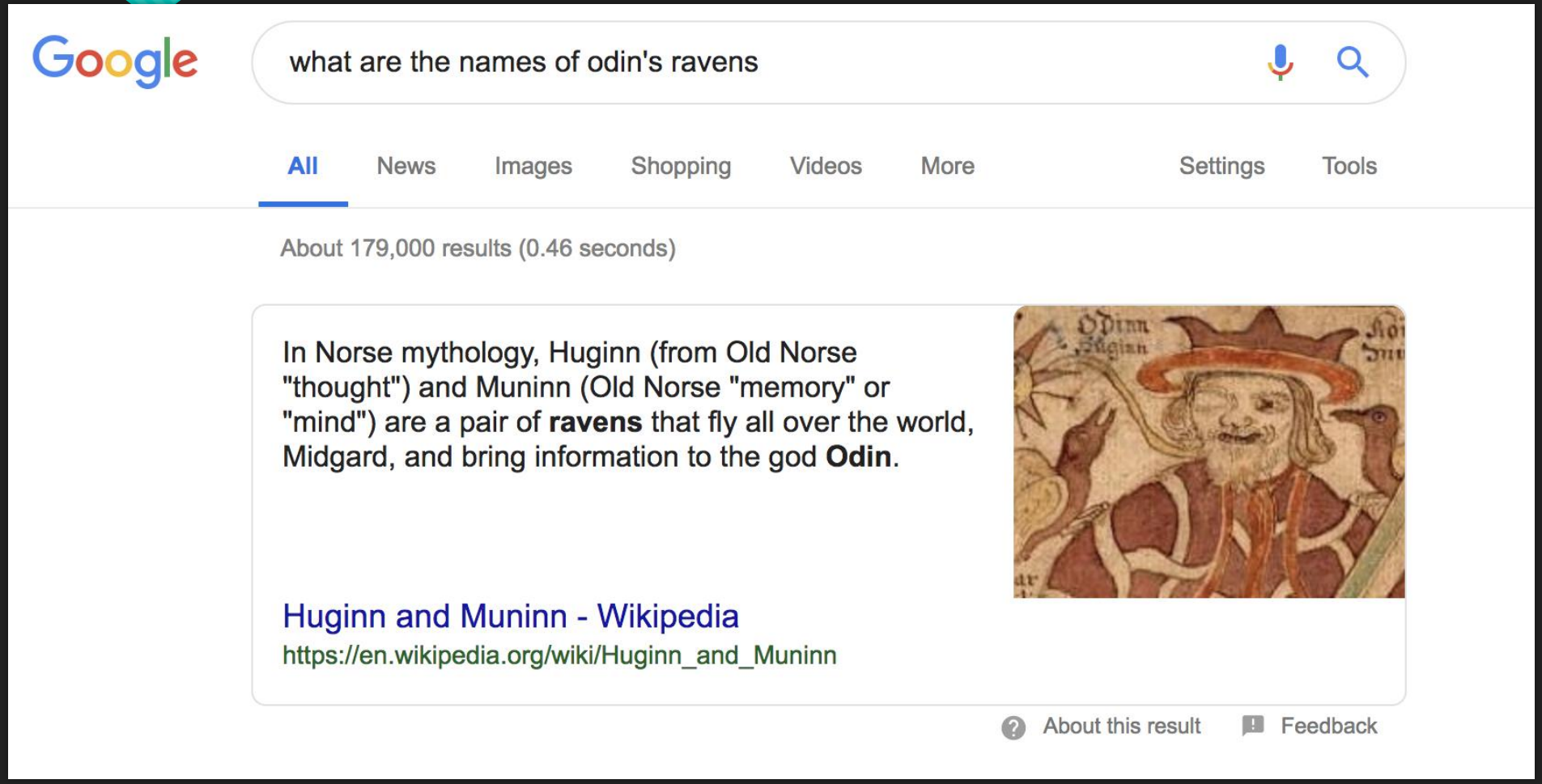
Credit to Dan Jurafksy for examples / slides

IR-based factoid question answering

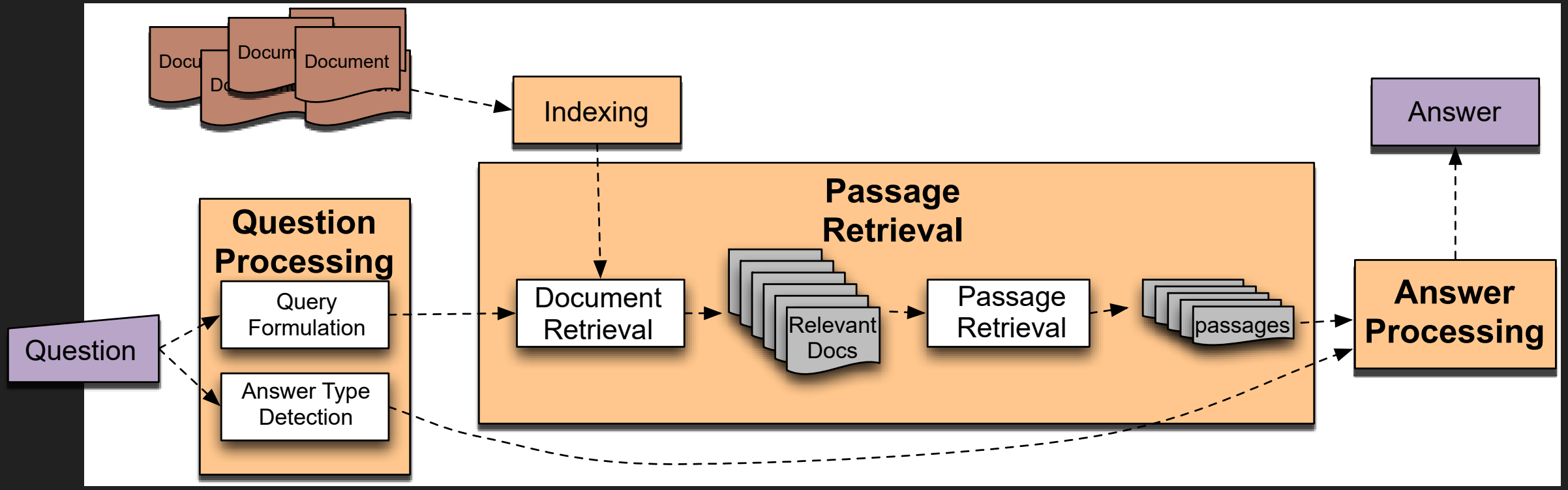
Example factoid questions and answers

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	the yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What's the official language of Algeria?	Arabic
How many pounds are there in a stone?	14

Many factoid questions can already be answered by web search



IR-based Factoid QA



IR-based Factoid QA

- **Question processing**
  - Detect question type, answer type, focus, relations
  - Formulate queries to send to a search engine
- **Passage retrieval**
  - Retrieve ranked documents
  - Break into suitable passages and rerank
- **Answer processing**
  - Extract candidate answers
  - Rank candidates
    - using evidence from the text and external sources

Question Processing: Things to extract from the question

- Answer Type Detection
  - Decide the **named entity type** (person, place) of the answer
- Query Formulation
  - Choose **query keywords** for the IR system
- Question Type classification
  - Is this a definition question, a math question, a list question?
- Focus Detection
  - Find the question words that are replaced by the answer
- Relation Extraction
  - Find relations between entities in the question

Question processing

What are the two states you could be reentering if you're crossing Florida's northern border?

- Answer Type: US state
- Query: two states, border, Florida, north
- Focus: the two states
- Relations: borders(Florida, ?x, north)

Answer Type Detection: Named Entities

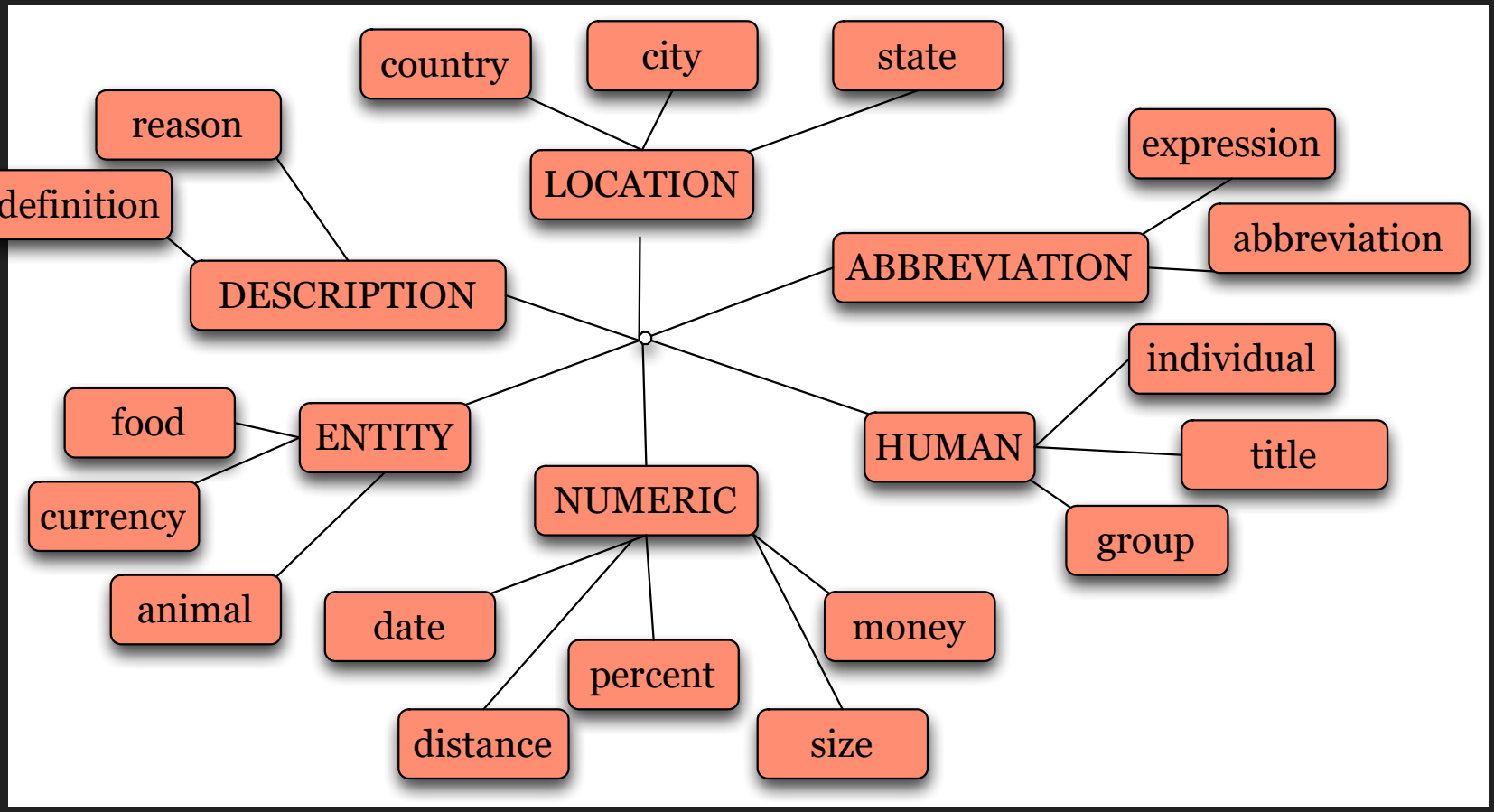
- *Who founded Virgin Airlines?*
  - PERSON
- *What Canadian city has the largest population?*
  - *CITY*

Answer Type Taxonomy

Xin Li, Dan Roth. 2002. Learning Question Classifiers. COLING'02

- 6 coarse classes
  - ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC
- 50 finer classes
  - LOCATION: city, country, mountain…
  - HUMAN: group, individual, title, description
  - ENTITY: animal, body, color, currency…

Part of Li & Roth's Answer Type Taxonomy



Answer Types

ENTITY	
animal	What are the names of Odin's ravens?
body	What part of your body contains the corpus callosum?
color	What colors make up a rainbow ?
creative	In what book can I find the story of Aladdin?
currency	What currency is used in China?
disease/medicine	What does Salk vaccine prevent?
event	What war involved the battle of Chapultepec?
food	What kind of nuts are used in marzipan?
instrument	What instrument does Max Roach play?
lang	What's the official language of Algeria?
letter	What letter appears on the cold-water tap in Spain?
other	What is the name of King Arthur's sword?
plant	What are some fragrant white climbing roses?
product	What is the fastest computer?
religion	What religion has the most members?
sport	What was the name of the ball game played by the Mayans?
substance	What fuel do airplanes use?
symbol	What is the chemical symbol for nitrogen?
technique	What is the best way to remove wallpaper?
term	How do you say " Grandma " in Irish?
vehicle	What was the name of Captain Bligh's ship?
word	What's the singular of dice?

More Answer Types

HUMAN	
description	Who was Confucius?
group	What are the major companies that are part of Dow Jones?
ind	Who was the first Russian astronaut to do a spacewalk?
title	What was Queen Victoria's title regarding India?
LOCATION	
city	What's the oldest capital city in the Americas?
country	What country borders the most others?
mountain	What is the highest peak in Africa?
other	What river runs through Liverpool?
state	What states do not have state income tax?
NUMERIC	
code	What is the telephone number for the University of Colorado?
count	About how many soldiers died in World War II?
date	What is the date of Boxing Day?
distance	How long was Mao's 1930s Long March?
money	How much did a McDonald's hamburger cost in 1963?
order	Where does Shanghai rank among world cities in population?
other	What is the population of Mexico?
period	What was the average life expectancy during the Stone Age?
percent	What fraction of a beaver's life is spent swimming?
speed	What is the speed of the Mississippi River?
temp	How fast must a spacecraft travel to escape Earth's gravity?
size	What is the size of Argentina?
weight	How many pounds are there in a stone?

Answer types in Jeopardy

Ferrucci et al. 2010. Building Watson: An Overview of the DeepQA Project. AI Magazine. Fall 2010. 59-79.

- 2500 answer types in 20,000 Jeopardy question sample
- The most frequent 200 answer types cover < 50% of data
- The 40 most frequent Jeopardy answer types

he, country, city, man, film, state, she, author, group, here, company, president, capital, star, novel, character, woman, river, island, king, song, part, series, sport, singer, actor, play, team, show, actress, animal, presidential, composer, musical, nation, book, title, leader, game

Answer Type Detection

- Hand-written rules
- Machine Learning
- Hybrids

Answer Type Detection

- Regular expression-based rules can get some cases:
  - Who {is | was | are | were} PERSON
- Other rules use the **question headword**:

  (the headword of the first noun phrase after the wh-word)

  - Which **city** in China has the largest number of foreign financial companies?
  - What is the state **flower** of California?

Answer Type Detection

Most often, we treat the problem as machine learning classification

- **Define** a taxonomy of question types
- **Annotate** training data for each question type
- **Train** classifiers for each question class
  - using a rich set of features.
  - features include those hand-written rules!

Features for Answer Type Detection

- Question words and phrases
- Part-of-speech tags
- Parse features (headwords)
- Named Entities
- Semantically related words

Keyword Selection Algorithm

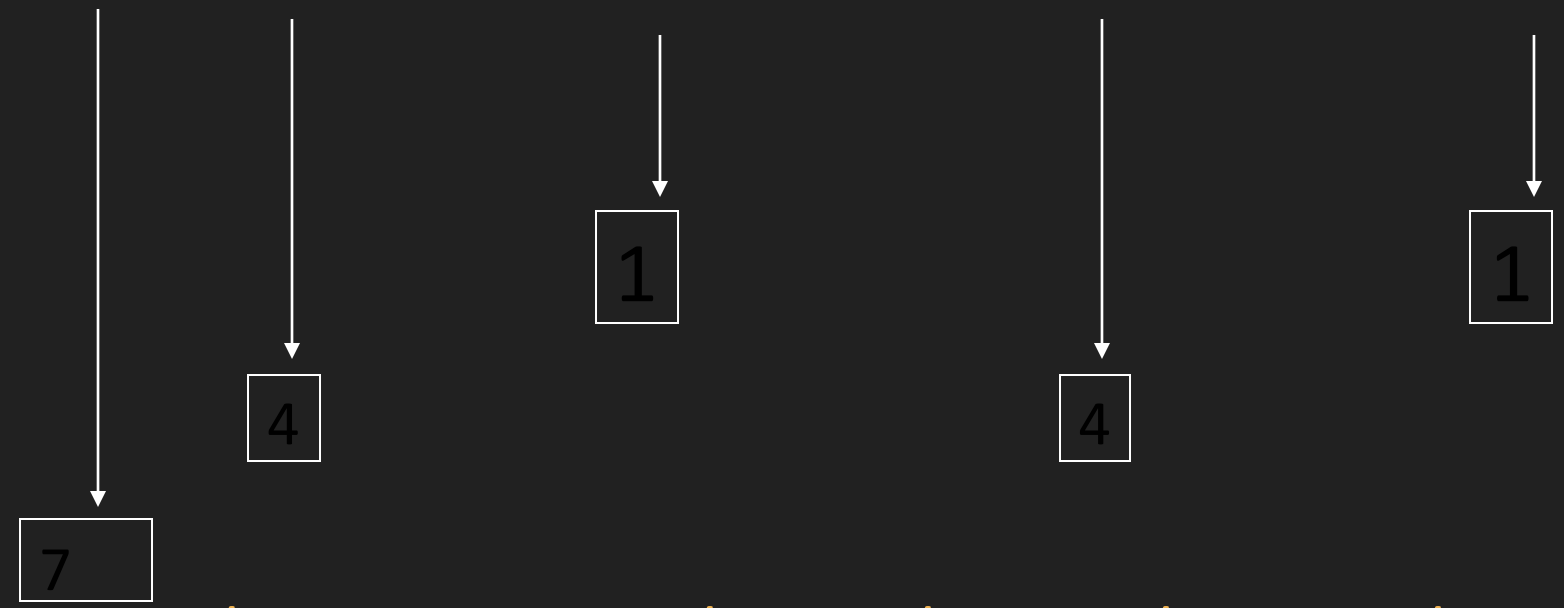
Dan Moldovan, Sanda Harabagiu, Marius Paca, Rada Mihalcea, Richard Goodrum, Roxana Girju and Vasile Rus. 1999. Proceedings of TREC-8.

1. Select all non-stop words in quotations
2. Select all NNP words in recognized named entities
3. Select all complex nominals with their adjectival modifiers
4. Select all other complex nominals
5. Select all nouns with their adjectival modifiers
6. Select all other nouns
7. Select all verbs
8. Select all adverbs
9. Select the QFW word (skipped in all previous steps)
10. Select all other words

Choosing keywords from the query

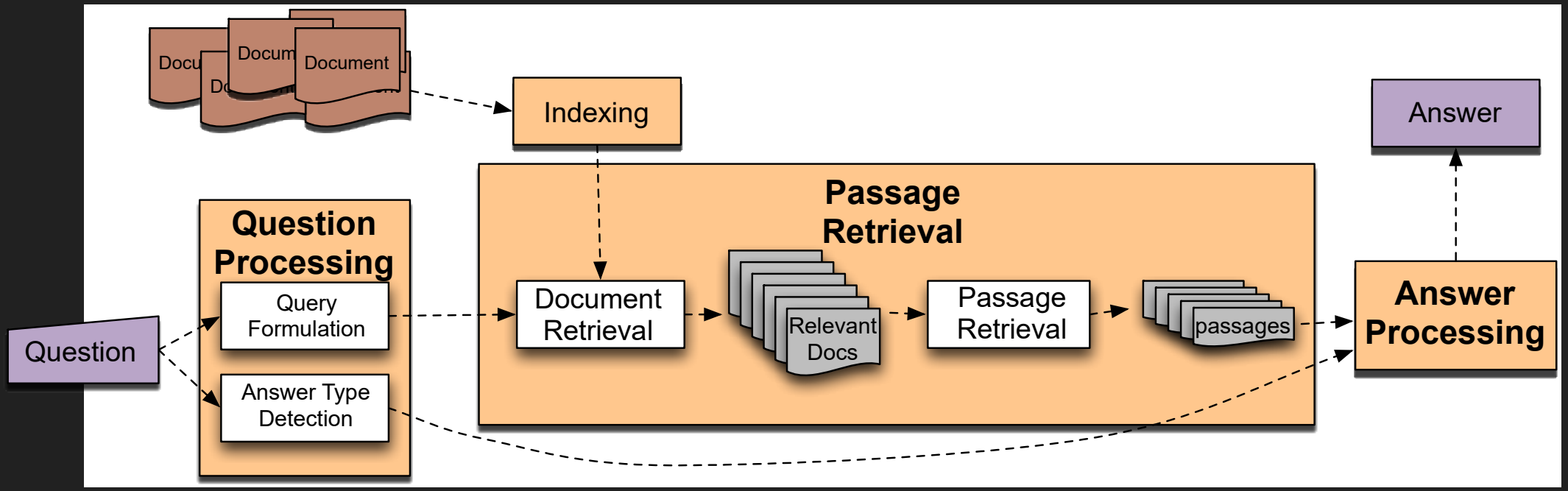
Slide from Mihai Surdeanu

Who coined the term “cyberspace” in his novel “Neuromancer”?



cyberspace/1 Neuromancer/1 term/4 novel/4 coined/7

Factoid Q/A


Document
Indexing
Answer
Question Processing
Query Formulation
Answer Type Detection
Question
Passage Retrieval
Document Retrieval
Relevant Docs
Passage Retrieval
passages
Answer Processing

Passage Retrieval

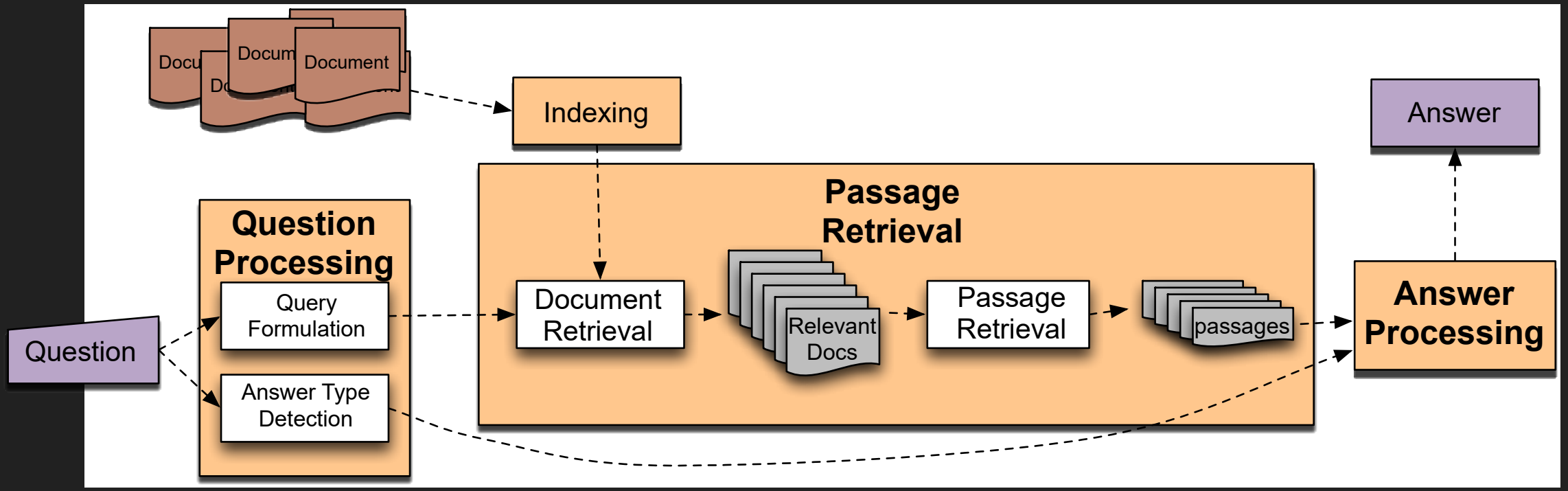
- **Step 1**: IR engine retrieves documents using query terms
- **Step 2**: Segment the documents into shorter units
  - something like paragraphs
- **Step 3**: Passage ranking
  - Use answer type to help rerank passages

Features for Passage Ranking

Either in rule-based classifiers or with supervised machine learning

- Number of Named Entities of the right type in passage
- Number of query words in passage
- Number of question N-grams also in passage
- Proximity of query keywords to each other in passage
- Longest sequence of question words
- Rank of the document containing passage

Factoid Q/A


Document
Indexing
Answer
Question Processing
Query Formulation
Answer Type Detection
Question
Passage Retrieval
Document Retrieval
Relevant Docs
Passage Retrieval
passages
Answer Processing

Answer Extraction

- Run an answer-type named-entity tagger on the passages
  - Each answer type requires a named-entity tagger that detects it
  - Can be full NER, simple regular expressions, or hybrid
- Return the string with the right type:
  - `Who is the prime minister of India` (PERSON)

    `The incumbent prime minister of India is Narendra Modi who has headed the BJP-led NDA government since 26 May 2014.`
  - `How tall is Mt. Everest?` (LENGTH)

    `The official height of Mount Everest is` **`29035 feet`**

Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: **Person**

- Passage:

The Marie biscuit is named after Marie Alexandrovna, the daughter of Czar Alexander II of Russia and wife of Alfred, the second son of Queen Victoria and Prince Albert

Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: **Person**

- Passage:

The Marie biscuit is named after Marie Alexandrovna, the daughter of Czar Alexander II of Russia and wife of Alfred, the second son of Queen Victoria and Prince Albert

Use machine learning: Features for ranking candidate answers

Answer type match: Candidate contains a phrase with the correct answer type.

Pattern match: Regular expression pattern matches the candidate.

Question keywords: # of question keywords in the candidate.

Keyword distance: Distance in words between the candidate and query keywords

Novelty factor: A word in the candidate is not in the query.

Apposition features: The candidate is next to question terms

Punctuation location: The candidate is immediately followed by a comma, period, quotation marks, semicolon, or exclamation mark.

Sequences of question terms: The length of the longest sequence of question terms that occurs in the candidate answer.

Common Evaluation Metrics

1. *Accuracy* (does answer match gold-labeled answer?)
2. *Mean Reciprocal Rank*
   - For each query return a ranked list of M candidate answers.
   - Query score is 1/Rank of the first correct answer
     - *If first answer is correct: 1*
     - *else if second answer is correct: ½*
     - *else if third answer is correct: ⅓, etc.*
     - *Score is 0 if none of the M answers are correct*
   - Take the mean over all N queries

$$MRR = \frac{\sum_{i=1}^{N} \frac{1}{rank_i}}{N}$$

Knowledge-based approaches (Siri)

- Build a semantic representation of the query
  - Times, dates, locations, entities, numeric quantities
- Map from this semantics to query structured data or resources
  - Geospatial databases
  - Ontologies (Wikipedia infoboxes, dbPedia, WordNet, Yago)
  - Restaurant review sources and reservation services
  - Scientific databases

Relation Extraction

- Answers: Databases of Relations
  - born-in("Emma Goldman", "June 27 1869")
  - author-of("Cao Xue Qin", "Dream of the Red Chamber")
  - Extract from Wikipedia infoboxes, DBpedia, FreeBase, etc.
- Questions: Extracting Relations in Questions

Whose granddaughter starred in E.T.?

```
(acted-in ?x "E.T.")
    (granddaughter-of ?x ?y)
```

Temporal Reasoning

- Relation databases
  - (and obituaries, biographical dictionaries, etc.)
- IBM Watson

``In 1594 he took a job as a tax collector in Andalusia”

Candidates:

- Thoreau is a bad answer (born in 1817)
- Cervantes is possible (was alive in 1594)

Geospatial knowledge (containment, directionality, borders)

- Beijing is a good answer for ``Asian city''
- California is ``southwest of Montana''
- geonames.org:

Context and conversation in virtual assistants like Siri

- Coreference helps resolve ambiguities

  U: "Book a table at Il Fornaio at 7:00 with **my mom**"

  U: "Also send **her** an email reminder"

- Clarification questions:

  U: "Chicago pizza"

  S: "Did you mean pizza restaurants in Chicago or Chicago-style pizza?"

Hybrid approaches (IBM Watson)

- Build a shallow semantic representation of the query
- Generate answer candidates using IR methods
  - Augmented with ontologies and semi-structured data
- Score each candidate using richer knowledge sources
  - Geospatial databases
  - Temporal reasoning
  - Taxonomical classification

What Now?

- Next week we will be focusing on exam preparation
- If you want to learn more about this field, take Advanced Natural Language Engineering

Making progress

- There is one notebook to complete for this week:
  - Part 1: Lab_10_1.ipynb