Introducing Information Retrieval
and Web Search
Information Retrieval

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of web search, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
Unstructured (text) vs. structured (database) data
Basic assumptions of Information Retrieval

Collection: A set of documents
- Assume it is a static collection for the moment

Goal: Retrieve documents with information that is relevant to the user’s information need and helps the user complete a task
The classic search model

User task:
- Get rid of mice in a politically correct way

Info need:
- Info about removing mice without killing them

Query:
- how trap mice alive

Search engine:

Results:

Collection:

Query refinement:

Misconception?:

Misformulation?:
How good are the retrieved docs?

- **Precision**: Fraction of retrieved docs that are relevant to the user’s information need

- **Recall**: Fraction of relevant docs in collection that are retrieved

  - More precise definitions and measurements to follow later
The Inverted Index

The key data structure underlying modern IR
Inverted index

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

- Identify each doc by a docID, a document serial number

Can we used fixed-size arrays for this?

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

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brutus</td>
<td>1  2  4  11  31  45  173  174</td>
</tr>
<tr>
<td>Caesar</td>
<td>1  2  4  5  6  16  57  132</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>2  31  54  101</td>
</tr>
</tbody>
</table>
Inverted index construction

Documents to be indexed

Token stream

Linguistic modules

Modified tokens

Indexer

Inverted index

Sec. 1.2

Friends, Romans, countrymen.
Initial stages of text processing

Tokenization
- Cut character sequence into word tokens
  - Deal with “John’s”, a state-of-the-art solution

Normalization
- Map text and query term to same form
  - You want U.S.A. and USA to match

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
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

<table>
<thead>
<tr>
<th>Term</th>
<th>docID</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>i’</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>with</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>hath</td>
<td>2</td>
</tr>
<tr>
<td>told</td>
<td>2</td>
</tr>
<tr>
<td>you</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>was</td>
<td>2</td>
</tr>
<tr>
<td>ambitious</td>
<td>2</td>
</tr>
</tbody>
</table>
Indexer steps: Sort

Sort by terms
- And then docID

Core indexing step

<table>
<thead>
<tr>
<th>Term</th>
<th>docID</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>with</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>hath</td>
<td>2</td>
</tr>
<tr>
<td>told</td>
<td>2</td>
</tr>
<tr>
<td>you</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>was</td>
<td>2</td>
</tr>
<tr>
<td>ambitious</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>docID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambitious</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>hath</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>it</td>
<td>1</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>told</td>
<td>2</td>
</tr>
<tr>
<td>you</td>
<td>2</td>
</tr>
<tr>
<td>was</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>2</td>
</tr>
<tr>
<td>with</td>
<td>2</td>
</tr>
</tbody>
</table>
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.
Where do we pay in storage?

- Lists of docIDs
- IR system implementation
  - How do we index efficiently?
  - How much storage do we need?
Query processing with an inverted index
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):

```
Brutus
    2  4  8  16  32  64  128
Caesar
    1  2  3  5  8  13  21  34
```
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.
(Longer) phrase queries

Longer phrases can be processed by breaking them down

**stanford university palo alto** can be broken into the Boolean query on biwords:

**stanford university** AND **university palo** AND **palo alto**

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!
Extended biwords

Parse the indexed text and perform part-of-speech-tagging (POST).

Bucket the terms into (say) Nouns (N) and articles/prepositions (X).

Call any string of terms of the form NX*N an extended biword.
  ◦ Each such extended biword is now made a term in the dictionary.

Example: catcher in the rye
        N   X    X    N

Query processing: parse it into N’s and X’s
  ◦ Segment query into extended biwords
  ◦ Look up in index: catcher rye
Issues for biword indexes

False positives, as noted before

Index blowup due to bigger dictionary
  ◦ Infeasible for more than biwords, big even for them

Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy
Solution 2: Positional indexes

In the postings, store, for each term the position(s) in which tokens of it appear:

<term, number of docs containing term;
doc1: position1, position2 ... ;
doc2: position1, position2 ... ;
etc.>
Positional index example

<be: 993427;
  1: 7, 18, 33, 72, 86, 231;
  2: 3, 149;
  4: 17, 191, 291, 430, 434;
  5: 363, 367, ...>

Which of docs 1, 2, 4, 5 could contain “to be or not to be”?

For phrase queries, we use a merge algorithm recursively at the document level.

But we now need to deal with more than just equality.
Processing a phrase query

Extract inverted index entries for each distinct term: to, be, or, not.

Merge their doc:position lists to enumerate all positions with “to be or not to be”.

- to:
  - 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...

- be:
  - 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...

Same general method for proximity searches
Proximity queries

Again, here, \( k \) means “within \( k \) words of”.

Clearly, positional indexes can be used for such queries; biword indexes cannot.

Exercise: Adapt the linear merge of postings to handle proximity queries. Can you make it work for any value of \( k \)?

This is a little tricky to do correctly and efficiently.
Positional index size

A positional index expands postings storage* substantially
  - Even though indices can be compressed

Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries ... whether used explicitly or implicitly in a ranking retrieval system.
Combination schemes

These two approaches can be profitably combined

- For particular phrases ("Michael Jackson", "Britney Spears") it is inefficient to keep on merging positional postings lists
  - Even more so for phrases like "The Who"

Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme

- A typical web query mixture was executed in \( \frac{1}{4} \) of the time of using just a positional index
- It required 26% more space than having a positional index alone
Introducing ranked retrieval
Problem with Boolean search: feast or famine

Boolean queries often result in either too few (∼0) or too many (1000s) results.

- Query 1: “standard user dlink 650” → 200,000 hits
- Query 2: “standard user dlink 650 no card found” → 0 hits

It takes a lot of skill to come up with a query that produces a manageable number of hits.

- AND gives too few; OR gives too many
Ranked retrieval models

Rather than a set of documents satisfying a query expression, in ranked retrieval models, the system returns an ordering over the (top) documents in the collection with respect to a query.

Free text queries: Rather than a query language of operators and expressions, the user’s query is just one or more words in a human language.

In principle, there are two separate choices here, but in practice, ranked retrieval models have normally been associated with free text queries and vice versa.
Scoring as the basis of ranked retrieval

We wish to return in order the documents most likely to be useful to the searcher

How can we rank-order the documents in the collection with respect to a query?

Assign a score – say in [0, 1] – to each document

This score measures how well document and query “match”.
Take 1: Jaccard coefficient

A commonly used measure of overlap of two sets $A$ and $B$ is the Jaccard coefficient

$$\text{jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{jaccard}(A,A) = 1$$

$$\text{jaccard}(A,B) = 0 \text{ if } A \cap B = 0$$

$A$ and $B$ don’t have to be the same size.

Always assigns a number between 0 and 1.
tf-idf weighting
Term-document count matrices

Consider the number of occurrences of a term in a document:
- Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Bag of words model**

Vector representation doesn’t consider the ordering of words in a document

*John is quicker than Mary* and *Mary is quicker than John* have the same vectors

This is called the **bag of words** model.

In a sense, this is a step back: The positional index was able to distinguish these two documents

- We will look at “recovering” positional information later on
- For now: bag of words model
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

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

$0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4$, etc.

Score for a document-query pair: sum over terms $t$ in both $q$ and $d$:

$$\text{score} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$$

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
  ◦ Recall stop words

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<sub>t</sub> is the document frequency of <i>t</i>: the number of documents that contain <i>t</i>

- df<sub>t</sub> is an inverse measure of the informativeness of <i>t</i>
- df<sub>t</sub> ≤ <i>N</i>

We define the idf (inverse document frequency) of <i>t</i> by

\[ \text{idf}_t = \log_{10} \left( \frac{N}{\text{df}_t} \right) \]

- We use log (\(N/\text{df}_t\)) instead of \(N/\text{df}_t\) to “dampen” the effect of idf.
Effect of idf on ranking

Question: 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 weighting

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

\[ w_{t,d} = (1 + \log_{10} \text{tf}_{t,d}) \times \log_{10}(\frac{N}{\text{df}_t}) \]

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
Final ranking of documents for a query

\[
\text{Score}(q, d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}
\]
Binary $\rightarrow$ count $\rightarrow$ weight matrix

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>5.25</td>
<td>3.18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>Brutus</td>
<td>1.21</td>
<td>6.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>8.59</td>
<td>2.54</td>
<td>0</td>
<td>1.51</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1.54</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>2.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1.51</td>
<td>0</td>
<td>1.9</td>
<td>0.12</td>
<td>5.25</td>
<td>0.88</td>
</tr>
<tr>
<td>worser</td>
<td>1.37</td>
<td>0</td>
<td>0.11</td>
<td>4.15</td>
<td>0.25</td>
<td>1.95</td>
</tr>
</tbody>
</table>

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$
The Vector Space Model (VSM)
Documents as vectors

Now we have a \(|V|\)-dimensional vector space

Terms are axes of the space

Documents are points or vectors in this space

Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine

These are very sparse vectors – most entries are zero
Queries as vectors

**Key idea 1:** Do the same for queries: represent them as vectors in the space

**Key idea 2:** Rank documents according to their proximity to the query in this space

proximity = similarity of vectors

proximity \approx \text{inverse of distance}

Recall: We do this because we want to get away from the you’re-either-in-or-out Boolean model

Instead: rank more relevant documents higher than less relevant documents
Use angle instead of distance

Thought experiment: take a document \(d\) and append it to itself. Call this document \(d'\).

“Semantically” \(d\) and \(d'\) have the same content.

The Euclidean distance between the two documents can be quite large.

The angle between the two documents is 0, corresponding to maximal similarity.

Key idea: Rank documents according to angle with query.
From angles to cosines

The following two notions are equivalent.

- Rank documents in decreasing order of the angle between query and document
- Rank documents in increasing order of \( \text{cosine(query,document)} \)

Cosine is a monotonically decreasing function for the interval \([0^\circ, 180^\circ]\)
From angles to cosines

But how – *and why* – should we be computing cosines?
Length normalization

A vector can be (length-) normalized by dividing each of its components by its length – for this we use the $L_2$ norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

Dividing a vector by its $L_2$ norm makes it a unit (length) vector (on surface of unit hypersphere)

Effect on the two documents $d$ and $d'$ ($d$ appended to itself) from earlier slide: they have identical vectors after length-normalization.

- Long and short documents now have comparable weights
\[ \cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{||\vec{q}|| ||\vec{d}||} = \frac{\vec{q}}{||\vec{q}||} \cdot \frac{\vec{d}}{||\vec{d}||} = \frac{\sum_{i=1}^{V} q_i d_i}{\sqrt{\sum_{i=1}^{V} q_i^2} \sqrt{\sum_{i=1}^{V} d_i^2}} \]

\( q_i \) is the tf-idf weight of term \( i \) in the query

\( d_i \) is the tf-idf weight of term \( i \) in the document

\( \cos(\vec{q}, \vec{d}) \) is the cosine similarity of \( \vec{q} \) and \( \vec{d} \) … or, equivalently, the cosine of the angle between \( \vec{q} \) and \( \vec{d} \).
Cosine for length-normalized vectors

For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

\[ \cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_{i=1}^{|V|} q_i d_i \]

for \( q, d \) length-normalized.
3 documents example contd.

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>jealous</td>
<td>2.00</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>gossip</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>wuthering</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>jealous</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>gossip</td>
<td>0.335</td>
<td>0</td>
<td>0.405</td>
</tr>
<tr>
<td>wuthering</td>
<td>0</td>
<td>0</td>
<td>0.588</td>
</tr>
</tbody>
</table>

\[
\cos(SaS, PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94
\]

\[
\cos(SaS, WH) \approx 0.79
\]

\[
\cos(PaP, WH) \approx 0.69
\]
tf-idf weighting has many variants

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>n (natural)</td>
<td>tf (_{t,d})</td>
<td>n (no) 1</td>
</tr>
<tr>
<td>l (logarithm)</td>
<td>1 + log(tf (_{t,d}))</td>
<td>t (idf) ( \log \frac{N}{df_t} )</td>
</tr>
<tr>
<td>a (augmented)</td>
<td>(0.5 + \frac{0.5 \times tf (<em>{t,d})}{\max_t(tf (</em>{t,d}))})</td>
<td>p (prob idf) ( \max{0, \log \frac{N - df_t}{df_t}} )</td>
</tr>
<tr>
<td>b (boolean)</td>
<td>(\begin{cases} 1 &amp; \text{if } tf (_{t,d}) &gt; 0 \ 0 &amp; \text{otherwise} \end{cases})</td>
<td>(\begin{cases} \text{b (byte size)} &amp; 1/\text{CharLength}^\alpha \ \alpha &amp; \text{&lt; 1} \end{cases})</td>
</tr>
<tr>
<td>L (log ave)</td>
<td>(\frac{1 + \log(tf (<em>{t,d}))}{1 + \log(\text{ave}</em>{t \leq d}(tf (_{t,d})))})</td>
<td></td>
</tr>
</tbody>
</table>

Columns headed ‘n’ are acronyms for weight schemes.

Amit Singhal, et.al. Pivoted Document Length Normalization; The normalization factor for documents for which \(P(\text{retrieval}) > P(\text{relevance})\) is increased, whereas the normalization factor for documents for which \(P(\text{retrieval}) < P(\text{relevance})\) is decreased using some slope to get high precision; pivot is the same probability point; computing \(u\) (normalization factor): \((1.0 - \text{slope}) \times \text{pivot} + \text{slope} \times \# \text{ of unique term}\).
Weighting may differ in queries vs documents

Many search engines allow for different weightings for queries vs. documents

SMART Notation: denotes the combination in use in an engine, with the notation \textit{ddd.qqq}, using the acronyms from the previous table

A very standard weighting scheme is: \textit{Inc.ltc}

Document: logarithmic tf (\textit{l} as first character), no idf and cosine normalization

Query: logarithmic tf (\textit{l} in leftmost column), idf (\textit{t} in second column), cosine normalization ...
**tf-idf example: Inc.ltc**

**Document:** *car insurance auto insurance*

**Query:** *best car insurance*

<table>
<thead>
<tr>
<th>Term</th>
<th>Query</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wt</td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Score** = 0 + 0 + 0.27 + 0.53 = 0.8

**Doc length** = \( \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92 \)
Evaluating search engines
Measures for a search engine

How fast does it index
- Number of documents/hour
- (Average document size)

How fast does it search
- Latency as a function of index size

Expressiveness of query language
- Ability to express complex information needs
- Speed on complex queries

Uncluttered UI

Is it free?
Measures for a search engine

All of the preceding criteria are measurable: we can quantify speed/size
  ◦ we can make expressiveness precise

The key measure: user happiness
  ◦ What is this?
  ◦ Speed of response/size of index are factors
  ◦ But blindingly fast, useless answers won’t make a user happy

Need a way of quantifying user happiness with the results returned
  ◦ Relevance of results to user’s information need
Evaluating an IR system

An **information need** is translated into a **query**

Relevance is assessed relative to the **information need** not the **query**

E.g., **Information need**: I’m looking for information on whether drinking **red wine** is more effective at reducing your risk of heart attacks than **white wine**.

**Query**: *wine red white heart attack effective*

You evaluate whether the doc addresses the information need, not whether it has these words.
Evaluating ranked results

Evaluation of a result set:

- If we have
  - a benchmark document collection
  - a benchmark set of queries
  - assessor judgments of whether documents are relevant to queries

  Then we can use Precision/Recall/F measure as before

Evaluation of ranked results:

- The system can return any number of results
- By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a precision-recall curve
Recall/Precision

R       P

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>10% R</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>20% R</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>30% R</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assume 10 rel docs in collection
current evaluation measures...

Mean average precision (MAP)

- AP: Average of the precision value obtained for the top $k$ documents, each time a relevant doc is retrieved
- Avoids interpolation, use of fixed recall levels
- Does weight most accuracy of top returned results
- MAP for set of queries is arithmetic average of APs
  - Macro-averaging: each query counts equally
Question Answering

One of the oldest NLP tasks (punched card systems in 1961)

Simmons, Klein, McConlogue. 1964. Indexing and Dependency Logic for Answering English Questions. American Documentation 15:30, 196-204

Question:
What do worms eat?

Potential Answers:

Worms eat grass
Horses with worms eat grass

Birds eat worms
Grass is eaten by worms
Question Answering: IBM’s Watson

Won Jeopardy on February 16, 2011!

WILLIAM WILKINSON’S “AN ACCOUNT OF THE PRINCIPALITIES OF WALLACHIA AND MOLDOVIA” INSPIRED THIS AUTHOR’S MOST FAMOUS NOVEL

Bram Stoker
how many calories are in two slices of banana cream pie?

Assuming any type of pie, banana cream | Use pie, banana cream, prepared from recipe or pie, banana cream, no-bake type, prepared from mix instead

Input interpretation:

<table>
<thead>
<tr>
<th>pie</th>
<th>amount</th>
<th>2 slices</th>
<th>total calories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>banana cream</td>
<td></td>
</tr>
</tbody>
</table>

Average result:

702 Cal (dietary Calories)
Types of Questions in Modern Systems

Factoid questions
- Who wrote “The Universal Declaration of Human Rights”?
- How many calories are there in two slices of apple pie?
- What is the average age of the onset of autism?
- Where is Apple Computer based?

Complex (narrative) questions:
- In children with an acute febrile illness, what is the efficacy of acetaminophen in reducing fever?
- What do scholars think about Jefferson’s position on dealing with pirates?
# Commercial systems: mainly factoid questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where is the Louvre Museum located?</td>
<td>In Paris, France</td>
</tr>
<tr>
<td>What’s the abbreviation for limited partnership?</td>
<td>L.P.</td>
</tr>
<tr>
<td>What are the names of Odin’s ravens?</td>
<td>Huginn and Muninn</td>
</tr>
<tr>
<td>What currency is used in China?</td>
<td>The yuan</td>
</tr>
<tr>
<td>What kind of nuts are used in marzipan?</td>
<td>Almonds</td>
</tr>
<tr>
<td>What instrument does Max Roach play?</td>
<td>Drums</td>
</tr>
<tr>
<td>What is the telephone number for Stanford University?</td>
<td>650-723-2300</td>
</tr>
</tbody>
</table>
Paradigms for QA

IR-based approaches
- TREC; IBM Watson; Google

Knowledge-based and Hybrid approaches
- IBM Watson; Apple Siri; Wolfram Alpha; True Knowledge Evi
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
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
Hybrid approaches (IBM Wats on)

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
Question Answering

ANSWER TYPES AND QUERY FORMULATION
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
They’re 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
- ABBEVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC

50 finer classes
- LOCATION: city, country, mountain...
- HUMAN: group, individual, title, description
- ENTITY: animal, body, color, currency...
# Answer Types

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

### HUMAN

<table>
<thead>
<tr>
<th>description</th>
<th>Who was Confucius?</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>What are the major companies that are part of Dow Jones?</td>
</tr>
<tr>
<td>ind</td>
<td>Who was the first Russian astronaut to do a spacewalk?</td>
</tr>
<tr>
<td>title</td>
<td>What was Queen Victoria’s title regarding India?</td>
</tr>
</tbody>
</table>

### LOCATION

<table>
<thead>
<tr>
<th>city</th>
<th>What’s the oldest capital city in the Americas?</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
<td>What country borders the most others?</td>
</tr>
<tr>
<td>mountain</td>
<td>What is the highest peak in Africa?</td>
</tr>
<tr>
<td>other</td>
<td>What river runs through Liverpool?</td>
</tr>
<tr>
<td>state</td>
<td>What states do not have state income tax?</td>
</tr>
</tbody>
</table>

### NUMERIC

<table>
<thead>
<tr>
<th>code</th>
<th>What is the telephone number for the University of Colorado?</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>About how many soldiers died in World War II?</td>
</tr>
<tr>
<td>date</td>
<td>What is the date of Boxing Day?</td>
</tr>
<tr>
<td>distance</td>
<td>How long was Mao’s 1930s Long March?</td>
</tr>
<tr>
<td>money</td>
<td>How much did a McDonald’s hamburger cost in 1963?</td>
</tr>
<tr>
<td>order</td>
<td>Where does Shanghai rank among world cities in population?</td>
</tr>
<tr>
<td>other</td>
<td>What is the population of Mexico?</td>
</tr>
<tr>
<td>period</td>
<td>What was the average life expectancy during the Stone Age?</td>
</tr>
<tr>
<td>percent</td>
<td>What fraction of a beaver’s life is spent swimming?</td>
</tr>
<tr>
<td>speed</td>
<td>What is the speed of the Mississippi River?</td>
</tr>
<tr>
<td>temp</td>
<td>How fast must a spacecraft travel to escape Earth’s gravity?</td>
</tr>
<tr>
<td>size</td>
<td>What is the size of Argentina?</td>
</tr>
<tr>
<td>weight</td>
<td>How many pounds are there in a stone?</td>
</tr>
</tbody>
</table>
Answer types in Jeopardy


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

Regular expression-based rules can get some cases:

- Who \{is|was|are|were\} PERSON
- PERSON (YEAR – YEAR)

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
Factoid Q/A
Keyword Selection Algorithm

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 the *QFW word (skipped in all previous steps) *question focus word

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

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

PASSAGE RETRIEVAL AND ANSWER EXTRACTION
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
Answer Extraction

Run an answer-type named-entity tagger on the passages
- Each answer type requires a named-entity tagger that detects it
- If answer type is CITY, tagger has to tag CITY
  - Can be full NER, simple regular expressions, or hybrid

Return the string with the right type:
- Who is the prime minister of India (PERSON)
  Manmohan Singh, Prime Minister of India, had told left leaders that the deal would not be renegotiated.
- How tall is Mt. Everest? (LENGTH)
  The official height of Mount Everest is 29035 feet
**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 an appositive 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.
Candidate Answer scoring in IBM Watson

Each candidate answer gets scores from >50 components
- (from unstructured text, semi-structured text, triple stores)
- logical form (parse) match between question and candidate
- passage source reliability
- geospatial location
  - California is "southwest of Montana"
- temporal relationships
- taxonomic classification
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.
   - Its score is $1/\text{Rank of the first right answer}$.
   - Take the mean over all N queries

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}$$
Question Answering

USING KNOWLEDGE IN QA
Relation Extraction

Answers: Databases of Relations
- born-in(“Emma Goldman”, “June 27 1869”)
- author-of(“Cao Xue Qin”, “Dream of the Red Chamber”)
- Draw 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"

gonames.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?”
Question Answering

SUMMARIZATION IN QUESTION ANSWERING
Text Summarization

**Goal**: produce an abridged version of a text that contains information that is important or relevant to a user.

**Summarization Applications**
- outlines or abstracts of any document, article, etc
- summaries of email threads
- action items from a meeting
- simplifying text by compressing sentences
What to summarize?
Single vs. multiple documents

**Single-document summarization**
- Given a single document, produce
  - abstract
  - outline
  - headline

**Multiple-document summarization**
- Given a group of documents, produce a gist of the content:
  - a series of news stories on the same event
  - a set of web pages about some topic or question
Query-focused Summarization & Generic Summarization

**Generic summarization:**
- Summarize the content of a document

**Query-focused summarization:**
- Summarize a document with respect to an information need expressed in a user query.
- A kind of complex question answering:
  - Answer a question by summarizing a document that has the information to construct the answer
Create **snippets** summarizing a web page for a query

- Google: 156 characters (about 26 words) plus title and link
Create answers to complex questions summarizing multiple documents.

- Instead of giving a snippet for each document
- Create a cohesive answer that combines information from each document
Extractive summarization & Abstractive summarization

**Extractive summarization:**
- create the summary from phrases or sentences in the source document(s)

**Abstractive summarization:**
- express the ideas in the source documents using (at least in part) different words
Simple baseline: take the first sentence

Die Brücke

From Wikipedia, the free encyclopedia

For other uses, see Die Brücke (disambiguation).

Die Brücke (The Bridge) was a group of German expressionist artists formed in Dresden in 1905, after which the Brücke Museum in Berlin was named. Founding members were Fritz Bleyl, Erich Heckel, Ernst Ludwig Kirchner and Karl Schmidt-Rottluff. Later members were Emil Nolde, Max Pechstein and Otto Mueller. The seminal group had a major impact on the evolution of modern art in the 20th century and the creation of expressionism.[1]

Die Brücke is sometimes compared to the Fauves. Both movements shared interests in primitivist art. Both
Summarization: Three Stages

1. **content selection**: choose sentences to extract from the document

2. **information ordering**: choose an order to place them in the summary

3. **sentence realization**: clean up the sentences
Basic Summarization Algorithm

1. **content selection**: choose sentences to extract from the document

2. **information ordering**: just use document order

3. **sentence realization**: keep original sentences
Unsupervised content selection


Intuition dating back to Luhn (1958):

- Choose sentences that have salient or informative words

Two approaches to defining salient words

1. **tf-idf:** weigh each word \( w_i \) in document \( j \) by tf-idf

   \[
   \text{weight}(w_i) = \text{tf}_{ij} \times \text{idf}_i
   \]

2. **topic signature:** choose a smaller set of salient words

   - mutual information

\[
\text{weight}(w_i) = \begin{cases} 
1 & \text{if } -2 \log \lambda(w_i) > 10 \\
0 & \text{otherwise}
\end{cases}
\]
Topic signature-based content selection with queries

Conroy, Schlesinger, and O’Leary 2006

choose words that are informative either

- by log-likelihood ratio (LLR)
- or by appearing in the query

\[
\text{weight}(w_i) = \begin{cases} 
1 & \text{if } -2\log \lambda(w_i) > 10 \\
1 & \text{if } w_i \in \text{question} \\
0 & \text{otherwise}
\end{cases}
\]

(could learn more complex weights)

Weigh a sentence (or window) by weight of its words:

\[
\text{weight}(s) = \frac{1}{|S|} \sum_{w \in S} \text{weight}(w)
\]
Supervised content selection

Given:
- a labeled training set of good summaries for each document

Align:
- the sentences in the document with sentences in the summary

Extract features
- position (first sentence?)
- length of sentence
- word informativeness, cue phrases
- cohesion

Problems:
- hard to get labeled training data
- alignment difficult
- performance not better than unsupervised algorithms

So in practice:
- Unsupervised content selection is more common

Train
- a binary classifier (put sentence in summary? yes or no)
ROUGE (Recall Oriented Understudy for Gisting Evaluation)

Lin and Hovy 2003

Intrinsic metric for automatically evaluating summaries
- Based on BLEU (a metric used for machine translation)
- Not as good as human evaluation ("Did this answer the user’s question?")
- But much more convenient

Given a document D, and an automatic summary X:

1. Have N humans produce a set of reference summaries of D
2. Run system, giving automatic summary X
3. What percentage of the bigrams from the reference summaries appear in X?

\[
\text{ROUGE} - 2 = \frac{\sum_{s \in \{\text{RefSummaries}\}} \sum_{i \in S} \min(\text{count}(i, X), \text{count}(i, S))}{\sum_{s \in \{\text{RefSummaries}\}} \sum_{i \in S} \text{count}(i, S)}
\]
A ROUGE example:
Q: “What is water spinach?”

Human 1: Water spinach is a green leafy vegetable grown in the tropics.

Human 2: Water spinach is a semi-aquatic tropical plant grown as a vegetable.

Human 3: Water spinach is a commonly eaten leaf vegetable of Asia.

System answer: Water spinach is a leaf vegetable commonly eaten in tropical areas of Asia.

ROUGE-2 =

\[
\frac{3 + 3 + 6}{10 + 9 + 9} = \frac{12}{28} = .43
\]
Question Answering

COMPLEX QUESTIONS: SUMMARIZING MULTIPLE DOCUMENTS
Q: What is water spinach?

A: Water spinach (Ipomoea aquatica) is a semi-aquatic leafy green plant with long hollow stems and spear- or heart-shaped leaves, widely grown throughout Asia as a leaf vegetable. The leaves and stems are often eaten stir-fried flavored with salt or in soups. Other common names include morning glory vegetable, kangkong (Malay), rau muong (Viet.), ong choi (Cant.), and kong xin cai (Mand.). It is not related to spinach, but is closely related to sweet potato and convolvulus.
Q: In children with an acute febrile illness, what is the efficacy of single medication therapy with acetaminophen or ibuprofen in reducing fever?

A: Ibuprofen provided greater temperature decrement and longer duration of antipyresis than acetaminophen when the two drugs were administered in approximately equal doses. (PubMedID: 1621668, Evidence Strength: A)
Other complex questions

Modified from the DUC 2005 competition (Hoa Trang Dang 2005)

1. How is compost made and used for gardening (including different types of compost, their uses, origins and benefits)?

2. What causes train wrecks and what can be done to prevent them?

3. Where have poachers endangered wildlife, what wildlife has been endangered and what steps have been taken to prevent poaching?

4. What has been the human toll in death or injury of tropical storms in recent years?
Answering harder questions: Query-focused multi-document summarization

The (bottom-up) snippet method
- Find a set of relevant documents
- Extract informative sentences from the documents
- Order and modify the sentences into an answer

The (top-down) information extraction method
- Build specific answerers for different question types:
  - definition questions
  - biography questions
  - certain medical questions
Query-Focused Multi-Document Summarization

Input Docs

Sentence Segmentation

All sentences from documents

Sentence Simplification

All sentences plus simplified versions

Sentence Extraction: LLR, MMR

Extracted sentences

Content Selection

Summary

Sentence Realization

Information Ordering
Simplifying sentences

Zajic et al. (2007), Conroy et al. (2006), Vanderwende et al. (2007)

Simplest method: parse sentences, use rules to decide which modifiers to prune
(more recently a wide variety of machine-learning methods)

<table>
<thead>
<tr>
<th>appositives</th>
<th>Rajam, 28, an artist who was living at the time in Philadelphia, found the inspiration in the back of city magazines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribution clauses</td>
<td>Rebels agreed to talks with government officials, international observers said Tuesday.</td>
</tr>
<tr>
<td>PPs without named entities</td>
<td>The commercial fishing restrictions in Washington will not be lifted unless the salmon population increases [PP to a sustainable number]</td>
</tr>
<tr>
<td>initial adverbials</td>
<td>“For example”, “On the other hand”, “As a matter of fact”, “At this point”</td>
</tr>
</tbody>
</table>
Maximal Marginal Relevance (MMR)

Jaime Carbonell and Jade Goldstein, The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries, SIGIR-98

An iterative method for content selection from multiple documents

Iteratively (greedily) choose the best sentence to insert in the summary/answer so far:

- **Relevant**: Maximally relevant to the user’s query
  - high cosine similarity to the query
- **Novel**: Minimally redundant with the summary/answer so far
  - low cosine similarity to the summary

Stop when desired length

\[ \hat{s}_{MMR} = \max_{s \in D} \lambda \text{sim}(s, Q) - (1-\lambda) \max_{s \in S} \text{sim}(s, S) \]
LLR+MMR: Choosing informative yet non-redundant sentences

One of many ways to combine the intuitions of LLR and MMR:

1. Score each sentence based on LLR (including query words)
2. Include the sentence with highest score in the summary.
3. Iteratively add into the summary high-scoring sentences that are not redundant with summary so far.
Information Ordering

Chronological ordering:
◦ Order sentences by the date of the document (for summarizing news).
  (Barzilay, Elhadad, and McKeown 2002)

Coherence:
◦ Choose orderings that make neighboring sentences similar (by cosine).
◦ Choose orderings in which neighboring sentences discuss the same entity
  (Barzilay and Lapata 2007)

Topical ordering:
◦ Learn the ordering of topics in the source documents
Domain-specific answering: The Information Extraction method

A good **biography** of a person contains:

- a person’s **birth/death, fame factor, education, nationality** and so on

A good **definition** contains:

- **genus** or **hyponym**
  - *The Hajj is a type of ritual*

A **medical answer about a drug’s use** contains:

- **the problem** (the medical condition),
- **the intervention** (the drug or procedure), and
- **the outcome** (the result of the study).
Information that should be in the answer for 3 kinds of questions

<table>
<thead>
<tr>
<th>Definition</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>genus</td>
<td>The Hajj is a type of ritual</td>
</tr>
<tr>
<td>species</td>
<td>the annual hajj begins in the twelfth month of the Islamic year</td>
</tr>
<tr>
<td>synonym</td>
<td>The Hajj, or Pilgrimage to Mecca, is the central duty of Islam</td>
</tr>
<tr>
<td>subtype</td>
<td>Qiran, Tamattu’, and Ifrad are three different types of Hajj</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Biography</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dates</td>
<td>was assassinated on April 4, 1968</td>
</tr>
<tr>
<td>nationality</td>
<td>was born in Atlanta, Georgia</td>
</tr>
<tr>
<td>education</td>
<td>entered Boston University as a doctoral student</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drug efficacy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>population</td>
<td>37 otherwise healthy children aged 2 to 12 years</td>
</tr>
<tr>
<td>problem</td>
<td>acute, intercurrent, febrile illness</td>
</tr>
<tr>
<td>intervention</td>
<td>acetaminophen (10 mg/kg)</td>
</tr>
<tr>
<td>outcome</td>
<td>ibuprofen provided greater temperature decrement and longer duration of antipyresis than acetaminophen when the two drugs were administered in approximately equal doses</td>
</tr>
</tbody>
</table>
The Hajj, or pilgrimage to Makkah [Mecca], is the central duty of Islam. More than two million Muslims are expected to take the Hajj this year. Muslims must perform the hajj at least once in their lifetime if physically and financially able. The Hajj is a milestone event in a Muslim's life. The annual hajj begins in the twelfth month of the Islamic year (which is lunar, not solar, so that hajj and Ramadan fall sometimes in summer, sometimes in winter). The Hajj is a week-long pilgrimage that begins in the 12th month of the Islamic lunar calendar. Another ceremony, which was not connected with the rites of the Ka'ba before the rise of Islam, is the Hajj, the annual pilgrimage to 'Arafat, about two miles east of Mecca, toward Mina...