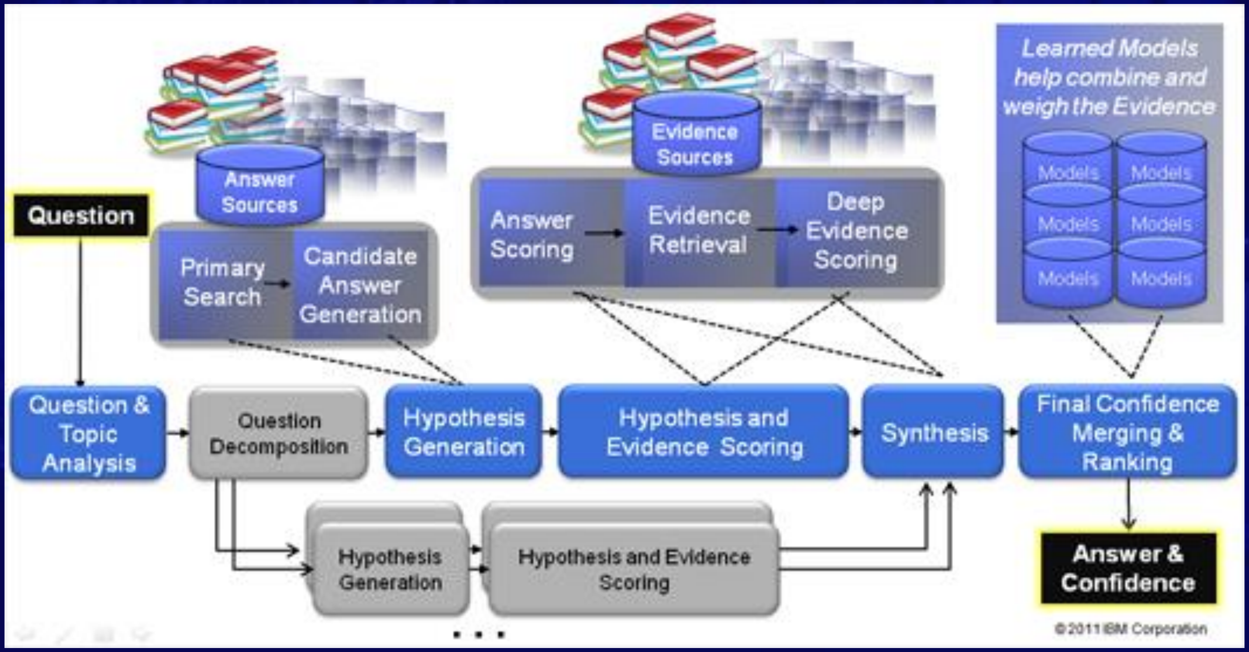
Statistical Natural Language Processing

Source: Richard Khoury (*Laval U.*)
Instructor: B. John Oommen

Objectives

- Overview of standard NLP techniques
- Build a question-answering AI
  - Text Representation → Classification → Information Retrieval → Question-Answering



Natural Language Processing

- Branch of AI that makes computer agents that can “use” human languages
  - Automated translation
  - Automated summary
  - Text classification & clustering
  - Text generation
  - Speech understanding & generation
  - Author recognition
  - Information retrieval & question-answering
  - Sentiment detection
  - Event detection in social media
  - Spam filtering
  - Digital assistants
  - Etc.

Statistical NLP

- Uses **corpus** of language to learn typical usage probabilities
- Develop algorithms that make best (e.g. most probably correct) decision
- Other alternatives:
  - Neural network NLP
  - Rule-based NLP
  - Fuzzy NLP

Statistical NLP

- Modelling language use with probabilities
  - “But I don’t count probabilities when I talk!”
  - Actually, you do…
- **Zipf’s law of least effort**
  - People try to do the minimum amount of effort
  - But they are smart about it
    - If more effort now means much less effort later on, they use effort now
  - So what is the effort to minimize when using language?

Least Effort in Language

- Imagine two people
  - Tina talks all the time
  - Lester listens all the time

Least Effort in Language

- What's the effort in talking?
- Choosing the words to say with the correct meaning
- Least effort = one word with all meanings

Least Effort in Language

- What's the effort in listening?
- Choosing the correct meaning of the words heard
- Least effort = one meaning per word

Least Effort in Language

- Tina and Lester have exactly opposite least efforts
- But in real life, no one talks or listens all the time
- Natural languages evolve to balance out effort of talking and listening

Least Effort in Language

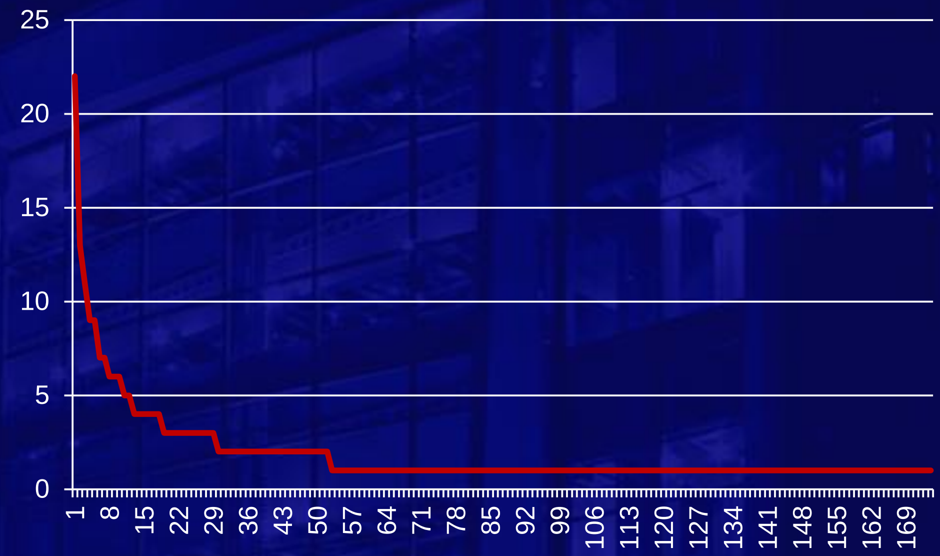
- Natural language have:
  - A few words that are used a lot and have flexible meanings
  - A lot of words that have precise meanings and are rarely used

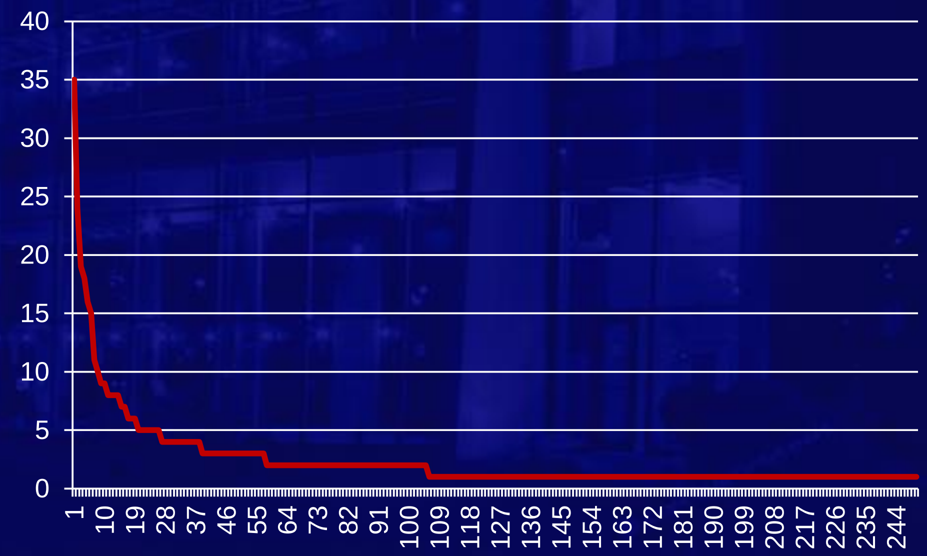
Zipf-Mandelbrot Law

- Count frequency of each word in a NL corpus of any language
- Plot frequency rank $\times$ frequency count
- Will create a power-law distribution
  - $frequency = P(rank + \rho)^{-B}$
  - English: $P = 10^{5.4}$, $\rho = 100$, $B = 1.15$
- Foundation of statistical NLP!

Zipf-Mandelbrot Law (example)

- Two articles from The Charlatan, 8 January 2014
- “Sprott unveils new Master of Accounting program”
  - 331 words
  - 174 distinct words
  - 122 words used once
- “CUSA Midterm Review: have they kept their campaign promises”
  - 611 words
  - 250 distinct words
  - 145 words used once





Zipf-Mandelbrot Law (example)

"Sprott unveils new Master of Accounting program"

Word	Frequency
the	22
to	13
program	11
said	9
of	9
Herauf	7
a	7
students	6
in	6
professional	6
by	5

"CUSA Midterm Review: have they kept their campaign promises"

Word	Frequency
the	35
to	24
of	19
and	18
a	16
CUSA	15
is	11
it	10
Odunayo	9
in	9
students	8

Zipf-Mandelbrot Law

- A few words that are used a lot and have flexible meanings
  - 10 most frequent words are 30% of each article
  - The (10 definitions), to (14 definitions), of (12 definitions), and (15 definitions)
  - Short list of stopwords that can be filtered out
- A lot of words that have precise meanings and are rarely used
  - More than half the words used only once
  - Includes: campaign, faculty, graduate, administration, vegetables, president, education,…
- And some exceptions!
  - Program, students, professional, CUSA, Herauf, Odunayo

Bag of Words

- The “word + frequency” representation of text is the **bag of words model**
  - Assumes words are independent of each other
  - Assumes word order does not matter
  - Sometimes enhanced by including **collocations**: pairs of words used together (ex.: wall street, make up)
- Creates the document vector

Word	Frequency
the	22
to	13
program	11
said	9
of	9
Herauf	7
a	7
students	6
in	6
professional	6
by	5

Bag of Words Classification

- Predicting the topic of a document given its words
  - Common task in NLP
    - Spam filtering, genre classification (libraries, websites), reading level assessment (education), etc.
    - Active research area
- **Supervised learning**: Trained on correct examples
  - Training corpus of classified documents
  - AI learns features useful for classification

Bag of Words Classification

- Training:
  1. Build word list of entire corpus
  2. Build bag of word for each category
- Example
  - 5 Charlatan articles each on sports & arts
    - “The best and brightest films of December”, “Aboriginal Service Centre hosts storytelling night”, “Author talks queerness in art history”, “Monthly concert series welcomes minors”, “Venus Envy hosts second dirty art show”, “Midseason review: Women’s hockey”, “Midseason review: Women’s basketball”, “Midseason review: Men’s basketball”, “Men’s hockey team edges York in exhibition play”, “Killeen practices with the Senators”
  - 2418 instances of 1233 words

	Art	Show	Films	Film	Ottawa	Years	Said	Ravens	Game	Team
Sports	0	1	0	0	4	1	26	32	28	17
Arts	24	11	8	6	9	1	30	0	2	2

Bag of Words Classification

- Testing
  1. Build word vector of new document *d*
  2. Compute cosine similarity with vector of each category *c*

$$\mathbf{w}_d \bullet \mathbf{w}_c = w_{d,0}w_{c,0} + w_{d,1}w_{c,1} + \ldots + w_{d,N-1}w_{c,N-1}$$

$$sim(d,c) = \frac{\mathbf{w}_d \bullet \mathbf{w}_c}{\|\mathbf{w}_d\|\|\mathbf{w}_c\|}$$

$$\|\mathbf{w}_c\| = \sqrt{w_{c,0}^2 + w_{c,1}^2 + \ldots + w_{c,N-1}^2}$$

  3. Classify into most similar category

Bag of Words Classification

	Art	Show	Films	Film	Ottawa	Years	Said	Ravens	Game	Team
Sports	0	1	0	0	4	1	26	32	28	17
Arts	24	11	8	6	9	1	30	0	2	2

- Can we do better?
  - We already filter out stopwords (determiners, articles, and pronouns)
- Group together different forms of same word
  - **Word stemming**: remove prefix, suffix and ending of words, keeping only stem or root
- Add words missing from some documents
  - **Expansion:** Add synonyms, dictionary definitions, collocations, of keywords
  - Especially useful for short texts (e.g. query expansion)
- Weight words
  - Tell apart significant and insignificant words

TF×IDF

- Recall Zipf-Mandelbrot law
  - Significant words are generally rare words with unusually high occurrence in a document
- So we need the general occurrence
  - Proportion of documents *d* in corpus *C* that contain word *w*
  - Inverse Document Frequency $idf(w, C) = \log \frac{|C|}{|w \in C|}$
- And the occurrence in document d_i
  - Term Frequency $tf(w, d_i) = \frac{|w \in d_i|}{|d_i|}$

TF×IDF

	Art	Show	Films	Film	Ottawa	Years	Said	Ravens	Game	Team
Sports	0	1	0	0	4	1	26	32	28	17
Arts	24	11	8	6	9	1	30	0	2	2

	Art	Show	Films	Film	Ottawa	Years	Said	Ravens	Game	Team
Sports	0	0.15	0	0	0.61	0.15	4.00	13.32	4.30	2.61
Arts	9.00	1.52	3.00	2.25	1.24	0.13	4.16	0	0.28	0.28

- Computation notes:
- |Sports| = 1147, |Arts| = 1271, |Corpus| = 3
  - Assume a 3rd category that is 0 everywhere
  - Otherwise log(2/2) = 0
  - Not an issue for real corpora
- TF×IDF values $\times 10^{-3}$

Naïve Bayes Classification

- Most popular alternative to bag of words

$$P(C|d) = P\left(C|w_{d,0},\ldots,w_{d,N-1}\right) = P(C)\prod_{i=0}^{N-1} P\left(w_{d,i}|C\right)$$

- *P*(*C*)
  - Proportion of documents per category in the corpus
- $P(w_{d,i}|C)$
  - Proportion of word w_i per category in the training corpus
- Classify in most probable category

Naïve Bayes Classification

$$P(C|d) = P\left(C|w_{d,0}, \ldots, w_{d,N-1}\right) = P(C) \prod_{i=0}^{N-1} P\left(w_{d,i}|C\right)$$

- $P(w_i|C)$ computed from observation in corpus
- What if word w_i never occurs in a category?
  - Multiplication by zero!
  - Given most words occur rarely (recall Zipf-Mandelbrot), this will happen a lot
- **Probability smoothing**: decreasing probability of observed words and distributing it to unobserved words

Information Retrieval

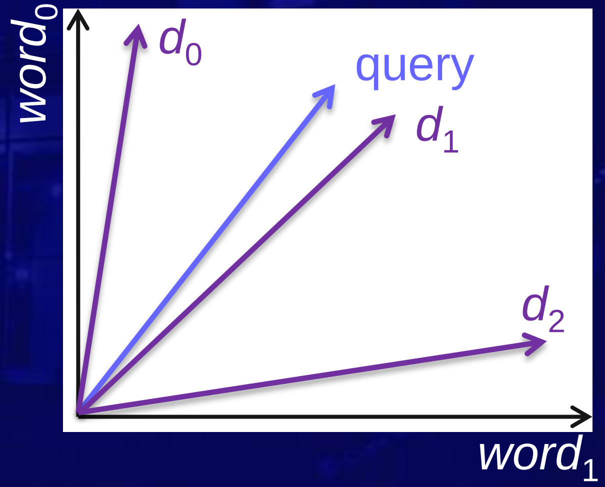
- Retrieve text documents that contain a requested (query) information in a corpus
- How?
  - Text classification (BoW or NB) identifies document topics and measures similarity of documents
  - IR Query = short document
  - Find most similar documents in corpus!

Information Retrieval

- Massive and unstructured document corpus
  - Institutional databases, Internet collections, Wikipedia
  - How to speed up IR search?
- **Inverted index**
  - Database with words as keys and documents as attributes
  - Faster retrieval of documents using keywords
- Positional information
  - Position of words within document
  - Easier to retrieve collocations
  - Find documents with multiple keywords in proximity of each other

Information Retrieval

- How to find and rank relevant documents?
- **Vector Space Model** (VSM)
  - Most widely used IR method
  - Each document's *N*-word-long bag-of-words is a vector in *N*-dimensional space
  - Cosine distance between query BoW and each document BoW
    - Most similar documents are most relevant
    - Can be made more accurate using TF×IDF, etc.
    - Inverted index useful to quickly find documents for cosine



Information Retrieval

- Problem: short queries
  - Might not use same words as documents
- Word co-occurrence
  - Some words are frequently used in the same context
  - If query has one word and document has the other, VSM won't match them

	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
Art	7	1	15	0	0	0	0	0	0	0
Show	9	1	0	1	0	0	1	0	0	0
Films	0	0	0	0	8	0	0	0	0	0
Ravens	0	0	0	0	0	7	7	5	6	9
Game	0	0	0	0	2	0	8	5	11	4
Team	0	0	0	0	2	3	3	3	2	3

Information Retrieval

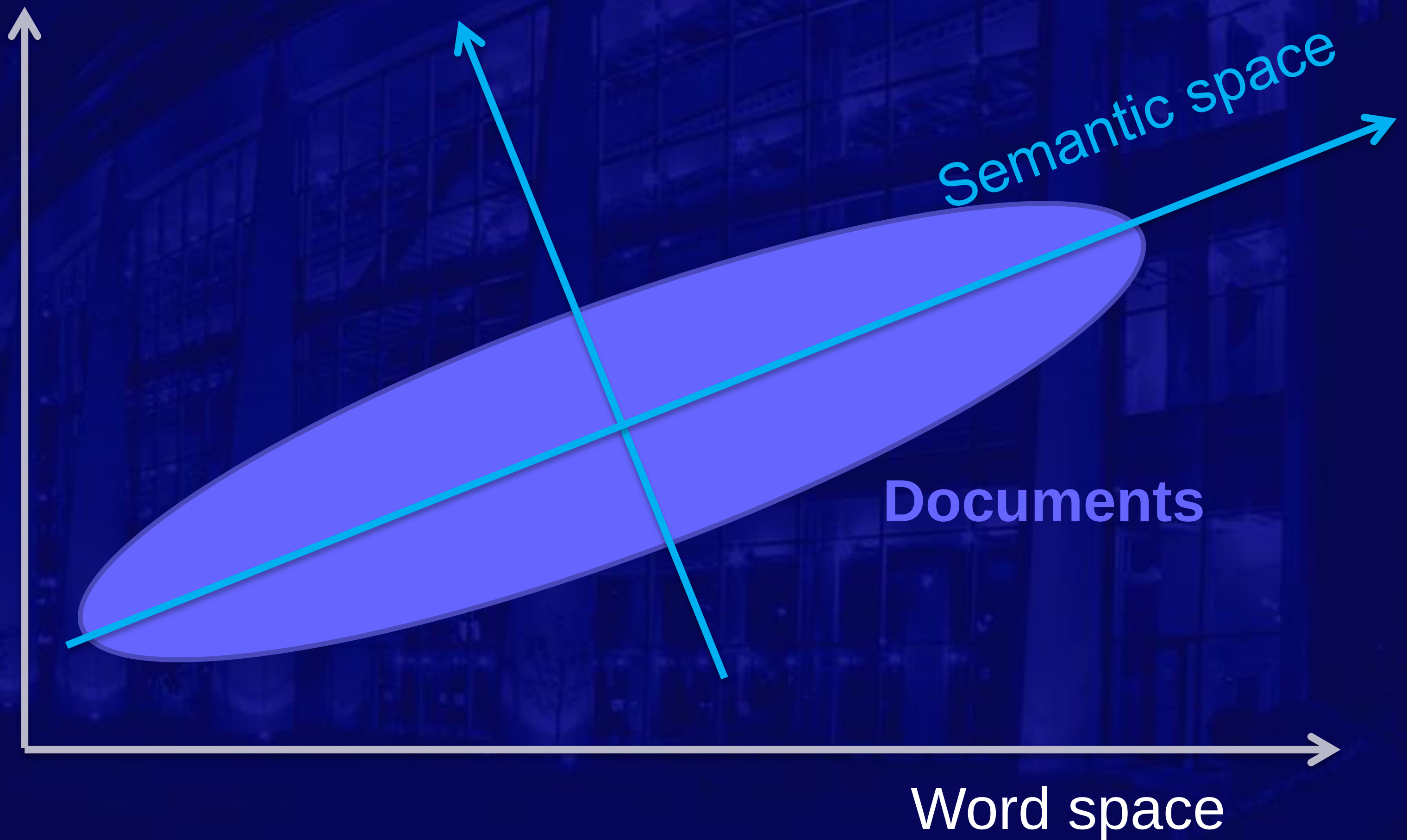
- **Solution:** **Latent Semantic Indexing** (LSI)
  - Vector space with latent semantic dimensions
  - Co-occurring words projected into same dimension
  - Non-co-occurring words projected into orthogonal dimensions
- Dimensionality reduction
  - VSM has N dimensions for N words
  - LSI has k dimensions for k semantic meanings
  - $k \ll N$

Information Retrieval with LSI

- Singular Value Decomposition (SVD)
  - LSI computation method
  - Represents matrix **A** (in word space) as matrix **Â** (in semantic space)
    - Matrix **A** is the document/word matrix with *N* words and *d* documents
    - Matrix **Â** is the semantic space matrix with *N* words and *d* documents
  - Sorts **Â** dimensions by importance
    - Makes it easy to reduce by cropping less useful dimension

Information Retrieval with LSI

- Visualizing SVD transformation



Information Retrieval with LSI

- Reduced Singular Value Decomposition
  - One of many SVD algorithms

1. Document/word matrix **A** with *N* words/dimensions
2. Decompose matrix as $\mathbf{A}_{N\times d} = \mathbf{U}_{N\times d}\mathbf{\Sigma}_{n\times n}(\mathbf{V}_{d\times n})^T$
   - Rotates dimensions to orientation of largest variation
   - Matrix **Σ** contains ordered singular values measuring variation of each dimension
3. Keep *k* dimensions in **Σ** where **Σ** value > threshold
   - Keep only *k* dimensions with noteworthy variations
   - $\hat{\mathrm{A}}_{N\times d} = \mathbf{U}_{N\times k}\mathbf{\Sigma}_{k\times k}(\mathbf{V}_{d\times k})^T$
4. Rescale to *k* dimensions
   - Word/document matrix: $\mathbf{A}_{N\times d} \rightarrow \mathbf{\Sigma}_{k\times k}(\mathbf{V}_{d\times k})^T = \mathbf{A}'_{k\times d}$
   - Document or IR query: $\mathbf{q}_{N\times 1} \rightarrow (\mathbf{U}_{N\times k})^T\mathbf{q}_{N\times 1} = \mathbf{q}'_{k\times 1}$

Information Retrieval with LSI

- Reduced SVD computation:

$$\mathbf{A}_{N\times d} = \mathbf{U}_{N\times d}\mathbf{\Sigma}_{n\times n}(\mathbf{V}_{d\times n})^T$$

  - **U** is the eigenvectors of $\mathbf{AA}^{\mathrm{T}}$ sorted by decreasing order of eigenvalue
  - **V** is the eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ sorted by decreasing order of eigenvalue
  - **Σ** Is the square roots of the eigenvalues sorted by decreasing order in a diagonal matrix

Information Retrieval with LSI

- Reduced SVD example:

$$\mathbf{A} = \begin{array}{c} \\ \textit{Art} \\ \textit{Show} \\ \textit{Films} \\ \textit{Ravens} \\ \textit{Game} \\ \textit{Team} \end{array} \begin{array}{cccccccccc} d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 & d_8 & d_9 \\ 7 & 1 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 7 & 5 & 6 & 9 \\ 0 & 0 & 0 & 0 & 2 & 0 & 8 & 5 & 11 & 4 \\ 0 & 0 & 0 & 0 & 2 & 3 & 3 & 3 & 2 & 3 \end{array}$$

$$\mathbf{A}\mathbf{A}^T = \begin{bmatrix} 275 & 64 & 0 & 0 & 0 & 0 \\ 64 & 84 & 0 & 7 & 8 & 3 \\ 0 & 0 & 64 & 0 & 16 & 16 \\ 0 & 7 & 0 & 240 & 183 & 96 \\ 0 & 8 & 16 & 183 & 230 & 77 \\ 0 & 3 & 16 & 96 & 77 & 44 \end{bmatrix}$$

Eigenvalues = $[455.58 \quad 294.40 \quad 70.95 \quad 64.32 \quad 50.45 \quad 1.30]$

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 130 & 16 & 105 & 9 & 0 & 0 & 9 & 0 & 0 & 0 \\ 16 & 2 & 15 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 105 & 15 & 225 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 72 & 6 & 22 & 16 & 26 & 14 \\ 0 & 0 & 0 & 0 & 6 & 58 & 58 & 44 & 48 & 72 \\ 9 & 1 & 0 & 1 & 22 & 58 & 123 & 84 & 136 & 104 \\ 0 & 0 & 0 & 0 & 16 & 44 & 84 & 59 & 91 & 74 \\ 0 & 0 & 0 & 0 & 26 & 48 & 136 & 91 & 161 & 104 \\ 0 & 0 & 0 & 0 & 14 & 72 & 104 & 74 & 104 & 106 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 0.01 & -0.96 & 0.00 & 0.29 & -0.02 & 0.00 \\ 0.03 & -0.29 & -0.01 & -0.95 & 0.07 & 0.00 \\ 0.04 & 0.00 & -0.87 & 0.04 & 0.43 & -0.23 \\ 0.69 & 0.01 & 0.37 & 0.05 & 0.49 & -0.37 \\ 0.66 & 0.01 & -0.31 & -0.03 & -0.68 & 0.01 \\ 0.29 & 0.01 & -0.07 & 0.03 & 0.32 & 0.90 \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 21.34 & 0 & 0 & 0 & 0 & 0 \\ 0 & 17.16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8.42 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.02 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7.10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.14 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0.02 & 0.54 & 0.01 & 0.82 & 0.07 & 0.02 \\ 0.00 & 0.07 & 0.00 & 0.08 & 0.01 & 0.00 \\ 0.01 & 0.84 & 0.00 & -0.54 & -0.04 & -0.01 \\ 0.00 & 0.02 & 0.00 & 0.12 & 0.01 & 0.00 \\ 0.10 & 0.00 & 0.92 & -0.04 & 0.38 & 0.03 \\ 0.27 & -0.01 & -0.29 & -0.06 & 0.62 & -0.11 \\ 0.52 & 0.00 & 0.01 & 0.09 & -0.14 & -0.16 \\ 0.36 & -0.01 & -0.01 & -0.03 & 0.00 & -0.79 \\ 0.56 & -0.01 & 0.15 & -0.01 & -0.55 & 0.28 \\ 0.46 & -0.01 & -0.23 & -0.05 & 0.38 & 0.51 \end{bmatrix}$$

Information Retrieval with LSI

- Reduced SVD example: Rescaling into 2D
  - Simply crop matrices to *k* = 2!

$$\mathbf{U} = \begin{bmatrix} 0.01 & -0.96 & 0.00 & 0.29 & -0.02 & 0.00 \\ 0.03 & -0.29 & -0.01 & -0.95 & 0.07 & 0.00 \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 21.34 & 0 \\ 0 & 17.16 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0.02 & 0.54 \\ 0.00 & 0.07 \\ 0.01 & 0.84 \\ 0.00 & 0.02 \\ 0.10 & 0.00 \\ 0.27 & 0.01 \\ 0.52 & 0.00 \\ 0.36 & -0.01 \\ 0.56 & -0.01 \\ 0.46 & -0.01 \end{bmatrix}$$

Information Retrieval with LSI

- Rescaling word/document matrix to 2D

$$\mathbf{A'}_{k\times d} = \mathbf{\Sigma}_{k\times k}(\mathbf{V}_{d\times k})^T$$

$$= \begin{bmatrix} 21.34 & 0 \\ 0 & 17.16 \end{bmatrix} \begin{bmatrix} 0.02 & 0.00 & 0.01 & 0.00 & 0.10 & 0.27 & 0.52 & 0.36 & 0.56 & 0.46 \\ 0.54 & 0.07 & 0.84 & 0.02 & 0.00 & -0.01 & 0.00 & -0.01 & -0.01 & -0.01 \end{bmatrix}$$

$$= \begin{bmatrix} & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 & d_8 & d_9 \\ \textit{Dimension 1} & 0.43 & 0.00 & 0.21 & 0.00 & 2.13 & 5.76 & 11.10 & 7.68 & 11.95 & 9.82 \\ \textit{Dimension 2} & 9.27 & 1.20 & 14.41 & 0.34 & 0.00 & -0.17 & 0.00 & -0.17 & -0.17 & -0.17 \end{bmatrix}$$

Information Retrieval with LSI

- Rescaling IR queries to 2D

$$\mathbf{q}'_{k\times 1} = (\mathbf{U}_{N\times k})^T \mathbf{q}_{N\times 1}$$

- Query 1: 10x “game”

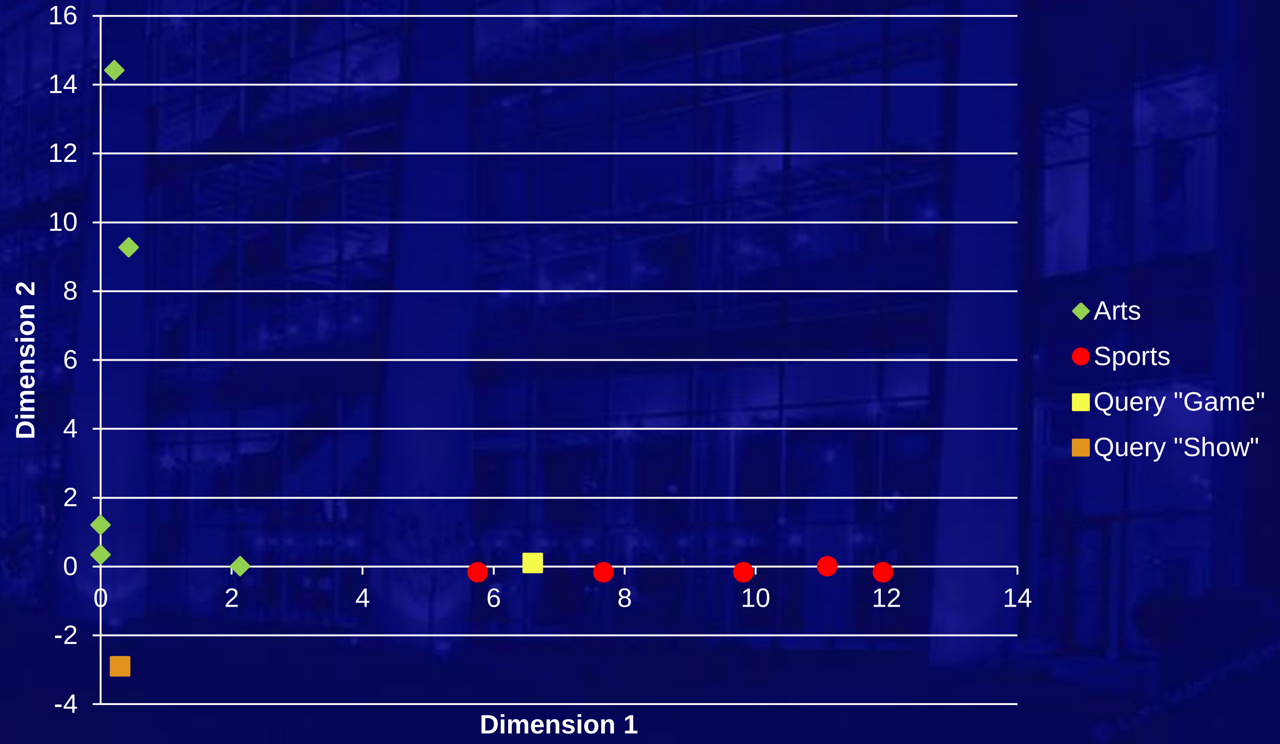
$$q'_1 = \begin{bmatrix} 0.01 & 0.03 & 0.04 & 0.69 & 0.66 & 0.29 \\ -0.96 & -0.29 & 0.00 & 0.01 & 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 10 \\ 0 \end{bmatrix} = \begin{bmatrix} \textit{Dimension 1} & 6.60 \\ \textit{Dimension 2} & 0.10 \end{bmatrix}$$

- Query 2: 10x “show”

$$q'_2 = \begin{bmatrix} 0.01 & 0.03 & 0.04 & 0.69 & 0.66 & 0.29 \\ -0.96 & -0.29 & 0.00 & 0.01 & 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \textit{Dimension 1} & 0.30 \\ \textit{Dimension 2} & -2.90 \end{bmatrix}$$

Information Retrieval with LSI

- SVD example: Visualization into 2D
  - Easy to find relevant documents for IR using Euclidean distance, cosine similarity, clustering, etc.



Question Answering

- IR: query is words, retrieved info is relevant documents
- QA: query is question, retrieved info is answer found in relevant documents
- Requires additional steps from IR
  - Question analysis
    - Precisely determine what is asked for
  - Passage retrieval
    - Find texts likely to contain answers
  - Answer selection
    - Pick the best answer from the passages
  - Answer presentation
    - Give the answer to the user

Question Analysis

- AI needs to understand what user is asking about
  - What type of information?
  - About what subject?
- Problem: Query is only a few words (1 to 4 on average after stopword removal)

Question Analysis

- **Question type** or **Answer type**
- What type of information is asked for?
  - E.g.: person, location, date, quantity, definition, yes/no, …
  - No standard type list
  - Not as simple as it sounds
    - “Who was Napoleon?” vs. “Who defeated Napoleon?”
    - “What French emperor was defeated at Waterloo?” vs. “What year was a French emperor defeated at Waterloo?” vs. “What was the Battle of Waterloo?”

Question Analysis

- Question type classification approaches
- Bayesian classification
  - $P(C|q) = P(C|w_0, \ldots, w_{N-1}) = P(C) \prod_{i=0}^{N-1} P(w_i|C)$
  - Where $P(C)$ is probability of a question type and $P(w_i|C)$ is probability of a query word given a question type
- Lexicon sorting keywords by query types
  - Classify question based on its keywords
  - Recall Zipf's law: a few words are used most often; lexicon can work with only 100 common words
- Pattern matching
  - "what {is|are} <noun phrase>" → Definition
  - Can match entire query or a subset of words (the **informer span**)

Question Analysis

- Dealing with few words: picking out important information
- **Named Entity Recognition** (NER)
  - A proper named used in the query is usually very important
  - Compare words to database of NE (can be constructed from Wikipedia)

Question Analysis

- Dealing with few words: inferring more words
- Query expansion
  - Problem: “Who killed Abraham Lincoln?” and “President Lincoln was assassinated by John Wilkes Booth.” have only one word in common
  - Add more words in query to help IR
- Add synonyms
  - Kill → murder, assassinate, take down, defeat
  - Problem: some words have multiple meanings
  - Lincoln → president, capital of Nebraska, mutton sheep

Passage Retrieval

- We can already retrieve documents with IR
- But for QA we need to retrieve specific passages with possible answers
  - "Who killed Abraham Lincoln?"
  - Answer is not Wikipedia page on Abraham Lincoln
  - Answer is not paragraph about Lincoln assassination
  - Answer is the passage "President Lincoln was assassinated by John Wilkes Booth."
- Different algorithms possible based on info available to us

Passage Retrieval (example)

- If we have:
  - Important query keywords
    - From NER, word weights (e.g. TFxIDF), etc.
  - Inverted index & positional information
    - From IR: Index of corpus words linked to documents with exact position of word in document
- Algorithm:
  - Window of words in document around each keyword

Passage Retrieval (example)

- If we have:
  - Question patterns & correct pattern for query (from question type classification)
  - Relevant documents (from IR)
- Algorithm:
  - Find matching answer patterns in document
  - Simple modification of question pattern
    - E.g.: “what {is|are} <noun phrase>” → “<noun phrase> {is|are} <answer>”
  - Can be enhanced with grammar rules, synonyms of less important words
    - “Who killed Abraham Lincoln” → “Abraham Lincoln was {killed|assassinated|murdered} by <answer>”

Answer Selection

- Passage retrieval will generate a lot of passages
  - Some will have the same answer written differently
  - Some will have different/conflicting answers
  - Some are irrelevant
- Problem: how to find the answer?
  - Need to rate confidence of individual passage
  - Need to compare/contrast different passages

Answer Selection

- Passage / answer rating
  - Determine system confidence in individual answer
- Does answer fit the question?
  - Important question words (especially named entities) appear in passage
  - Keywords in passage match question semantics (distance in ontology, WordNet)
  - Answer in passage match syntactic role in question
- Does answer fit predicted answer type?
  - But the answer type classifier can make mistakes…
- Geospatial & temporal relevance of answer
  - Useful for cell phone QA, virtual assistants, etc.

Answer Selection

- Deal with same answer written differently
  - System needs lists of synonyms, abbreviations, idioms, alternate spellings, etc.
  - Merge together answers & combine ratings
    - Can serve as democratic rating below
- Deal with different/conflicting answers
  - Answer rating includes answer frequency
    - Democratic: correct answer is most frequently cited in different sources
    - But don’t score same source multiple times!
  - Answer rating includes source reliability
    - Elitism: answer from trusted source is correct one

Answer Presentation

- What is returned by QA system?
  - Single best answer vs. list of possible answers
  - Include or not confidence rating of answers
  - Include or not sources of answer
  - Reword answer properly or copy-paste text

Answer Presentation

- How to reword answers properly
- Syntactic answer templates
  - Detect question syntax
    - Words' part-of-speech and word order
  - Corresponding answer template filled it from question and answer words
  - Advantage: human-like answers
- Semantic answer templates
  - Tag semantic content in answer
    - E.g. core answer, complementary information, justifications, constraints, etc.
  - Template selects which content to display and where
  - Advantage: can be tailored to devices or needs without losing critical info

Q-Sentence
What was the president of Yugoslavia indicted for?
Q-Template
WP VBD DT NN IN NP VVD IN
Original A-sentence
Judge Arbour's last major act as chief prosecutor was to indict Slobodan Milosevic for crimes against humanity.
A-Template
DT NN IN NP VBD VVN IN NNS IN NN
Reworded A-Sentence
The president of Yugoslavia was indicted for crimes against humanity.

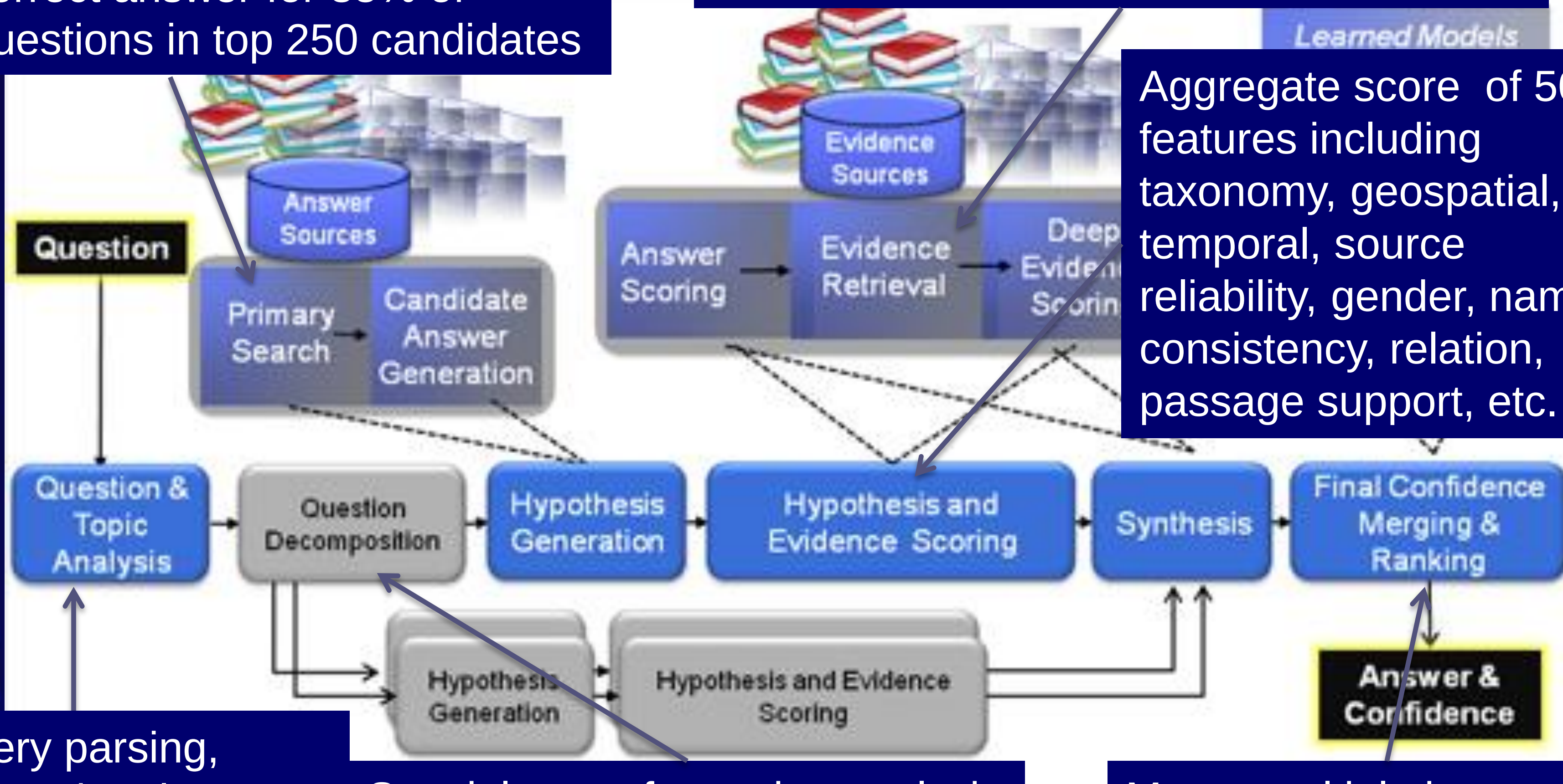
```
Template 2
if ((mobile device) && (presentation on screen))
then SET STYLE = ¬VERBOSE
     provide qmp:CoreInfo
     provide qmp:Justification
```

Example of QA Systems: IBM Watson

Passage retrieval contains correct answer for 85% of questions in top 250 candidates

Evidence: passage search featuring candidate answer in context of question

Aggregate score of 50 features including taxonomy, geospatial, temporal, source reliability, gender, name consistency, relation, passage support, etc.



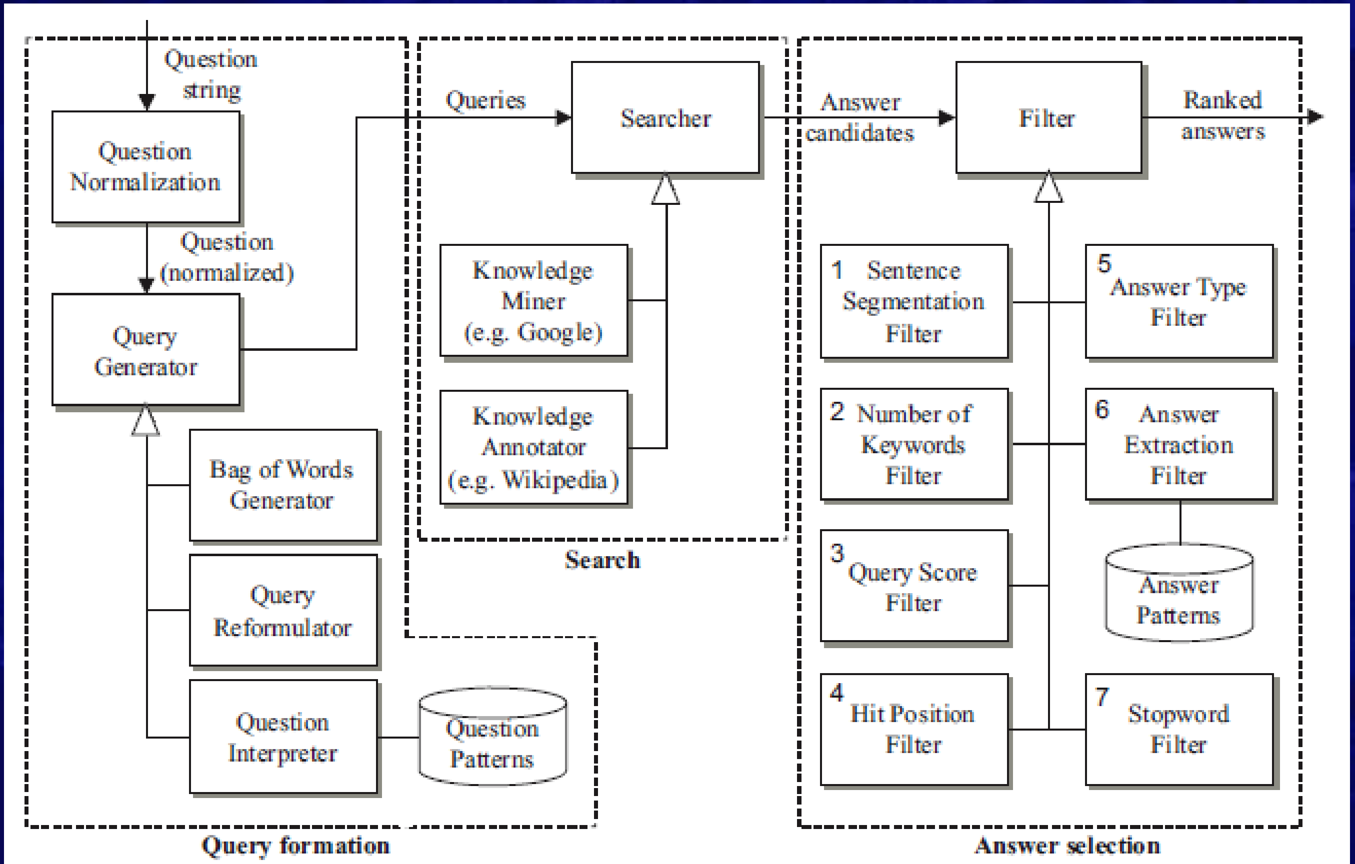
Query parsing, semantic role labelling, NER, etc.

Special type of question analysis to handle Jeopardy clues

Merge multiple instances of same answer

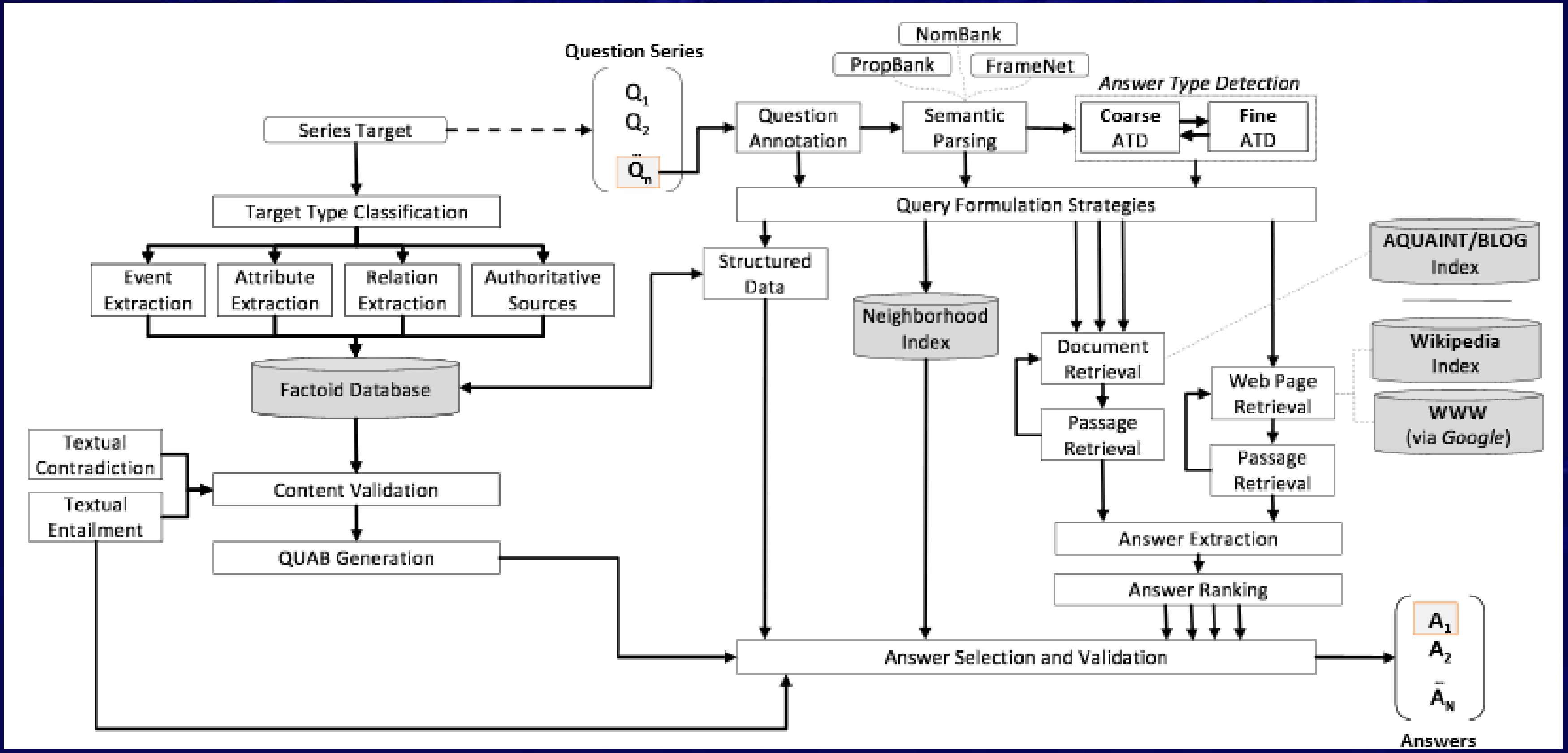
Example of QA Systems: Ephyra

- Ephyra uses “filters” for answer selection
- Selection and order can be changed



Example of QA Systems: LCC Chaucer-2

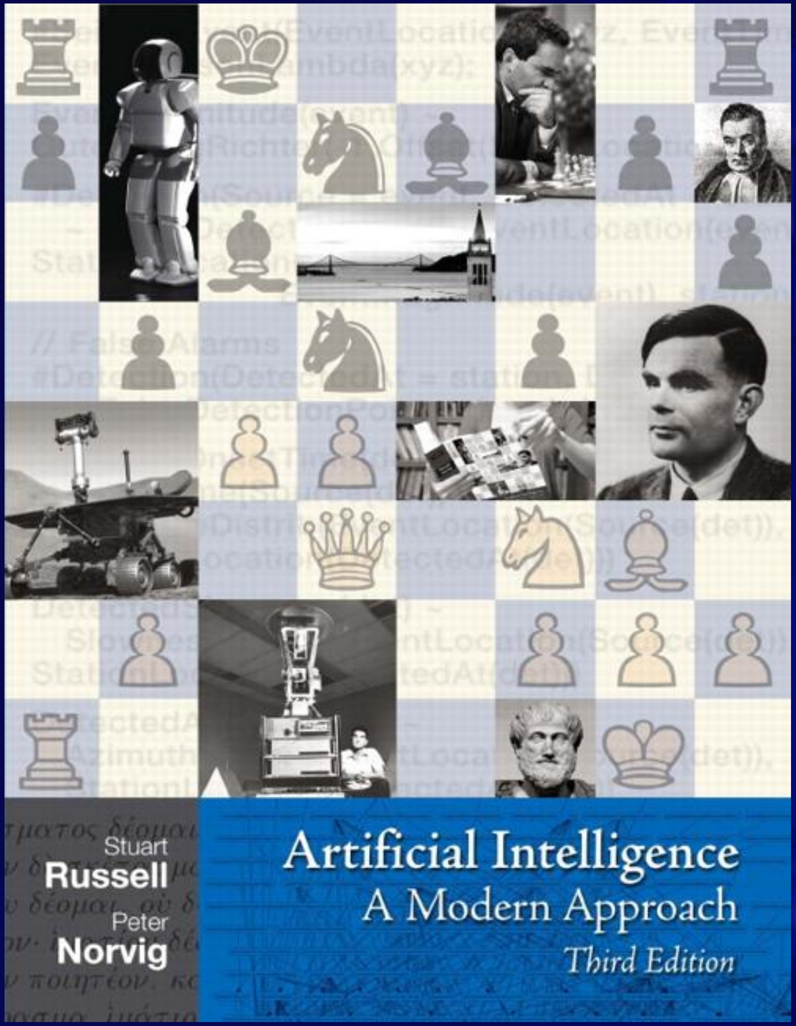
- 2nd place in TREC 2007 QA competition
- Series of queries on each topic

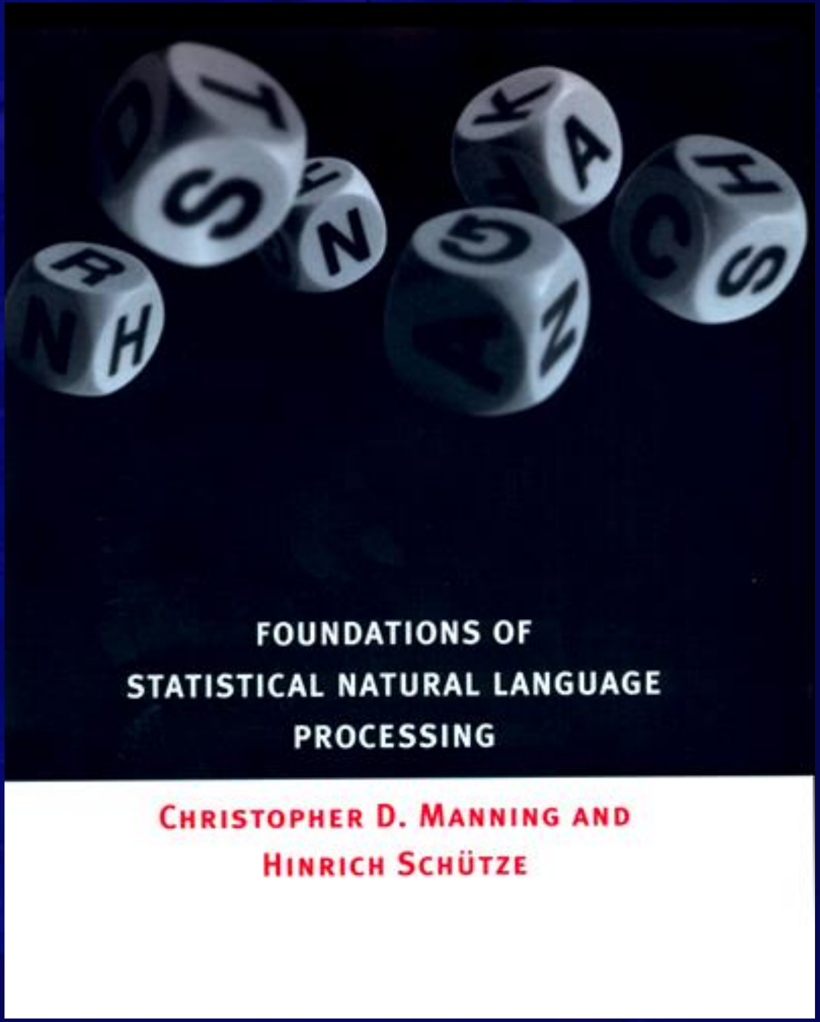


Summary: Natural Language Processing

- Zipf-Mandelbrot Law
- Bag of Words representation
  - Stopwords, word stemming, collocations, TFxIDF
- Classification
  - Bag of Words & cosine distance, Naïve Bayes
- Information retrieval
  - Inverted index, Vector Space Model, Latent Semantic Indexing, text segmentation
- Question Answering
  - Question analysis, passage retrieval, answer rating, answer presentation, examples

Further Readings





Further Readings

- U. Côté Allard, R. Khoury, L. Lamontagne, J. Bergeron, F. Laviolette, and A. Bergeron-Guyard, "Optimizing Question-Answering Systems Using Genetic Algorithms", *FLAIRS-28* (2015).
- F.B. Dian Paskalis and M.L. Khodra, "Word sense disambiguation in information retrieval using query expansion", *ICEEI, pp. 1-6* (2011).
- D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, C. Welty, "Building Watson: An Overview of the DeepQA Project", *AI Magazine*, 31:3, pp. 59-79 (2010).
- S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrji, V. Rus and P. Morarescu. "FALCON: Boosting knowledge for answer engines", *Proceedings of TREC-9, pp. 479-488* (2000).
- A. Hickl, K. Roberts, B. Rink, J. Bensley, T. Jungen, Y. Shi, J. Williams, "Question answering with LCC's Chaucer-2 at TREC 2007", *Proceedings of TREC-16* (2007).
- J. Luo, B. Meng and X. Tu "Selecting good expansion terms based on similarity kernel function", *NLP-KE, pp. 156-160* (2011).
- B. Magnini, M. Speranza and V. Kumar. "Towards Interactive Question Answering: An Ontology-Based Approach", *IEEE ICSC '09, pp. 612-617 (2009).*
- B. J. Oommen, R. Khoury and A. Schmidt, "Text Classification Using Novel Anti-Bayesian Techniques", ICCCI-7 (2015).
- M. Razmara, A. Fee and L. Kosseim. "Concordia University at the TREC 2007 QA track"; *Proceedings of TREC-16* (2007).
- N. Schlaefer, R. Gieselmann, T. Schaaf, A. Waibel, A pattern learning approach to question answering within the Ephyra framework. *LNAI,* 4188:687-694. P. Sojka, I. Kopecek, K. Pala, (eds.): Springer. (2006)
- D. Shen, R. Pan, J.-T. Sun, J.J. Pan, K. Wu, J. Yin, and Q. Yang. "Q2C@UST: our winning solution to query classification in KDDCUP 2005", *ACM SIGKDD, vol. 7,* no. 2, pp. 100-110 (2005).
- S. Song and Y.-N. Cheah, "An unsupervised center sentence-based clustering approach for rule-based question answering", *IEEE ISCI, pp. 125-129* (2011).

Thank You