# Strategies for QA & Information Retrieval

Ling573
NLP Systems and Applications
April 10, 2014

# Roadmap

- Shallow and Deep processing for Q/A
  - AskMSR, ARANEA: Shallow processing Q/A
    - Wrap-up
  - PowerAnswer-2: Deep processing Q/A

- Information Retrieval:
  - Problem:
    - Matching Topics and Documents
  - Methods:
    - Vector Space Model
  - Retrieval evaluation

# Redundancy-based Answer Extraction

- Prior processing:
  - Question formulation
  - Web search
  - Retrieve snippets – top 100
- N-grams:
  - Generation
  - Voting
  - Filtering
  - Combining
  - Scoring
  - Reranking

# N-gram Filtering

- Throws out ‘blatant’ errors
  - Conservative or aggressive?
    - Conservative: can’t recover error
- Question-type-neutral filters:
  - Exclude if begin/end with stopword
  - Exclude if contain words from question, except
    - ‘Focus words’ : e.g. units
- Question-type-specific filters:
  - ‘how far’, ‘how fast’: exclude if no numeric
  - ‘who’,’where’: exclude if not NE (first & last caps)

# N-gram Filtering

- Closed-class filters:
  - Exclude if not members of an enumerable list

# N-gram Filtering

- Closed-class filters:
  - Exclude if not members of an enumerable list
  - E.g. ‘what year ‘ -> must be acceptable date year

# N-gram Filtering

- Closed-class filters:
  - Exclude if not members of an enumerable list
  - E.g. ‘what year ‘ -> must be acceptable date year
- Example after filtering:
  - Who was the first person to run a sub-four-minute mile?

| Candidate | Score |
|---|---|
| Bannister | 137 |
| Roger | 114 |
| Roger Bannister | 103 |
| English | 26 |
| ... | ... |

# N-gram Combining

- Current scoring favors longer or shorter spans?

# N-gram Combining

- Current scoring favors longer or shorter spans?
  - E.g. Roger or Bannister or Roger Bannister or Mr.....

# N-gram Combining

- Current scoring favors longer or shorter spans?
  - E.g. Roger or Bannister or Roger Bannister or Mr.....
    - Bannister pry highest – occurs everywhere R.B. +
- Generally, good answers longer (up to a point)

# N-gram Combining

- Current scoring favors longer or shorter spans?
  - E.g. Roger or Bannister or Roger Bannister or Mr.....
    - Bannister pry highest – occurs everywhere R.B. +
- Generally, good answers longer (up to a point)
- Update score: $S_c$ += $\Sigma S_t$, where t is unigram in c
- Possible issues:

# N-gram Combining

- Current scoring favors longer or shorter spans?
  - E.g. Roger or Bannister or Roger Bannister or Mr.....
    - Bannister pry highest – occurs everywhere R.B. +
- Generally, good answers longer (up to a point)
- Update score: $S_c$ += $\Sigma S_t$, where t is unigram in c
- Possible issues:
  - Bad units: Roger Bannister was

# N-gram Combining

- Current scoring favors longer or shorter spans?
  - E.g. Roger or Bannister or Roger Bannister or Mr.....
    - Bannister pry highest – occurs everywhere R.B. +
- Generally, good answers longer (up to a point)
- Update score: $S_c \mathrel{+}= \Sigma S_t$, where t is unigram in c
- Possible issues:
  - Bad units: Roger Bannister was – blocked by filters
    - Also, increments score so long bad spans lower
- Improves significantly

# N-gram Scoring

- Not all terms created equal

# N-gram Scoring

- Not all terms created equal
  - Usually answers highly specific
  - Also disprefer non-units
- Solution

# N-gram Scoring

- Not all terms created equal
  - Usually answers highly specific
  - Also disprefer non-units
- Solution: IDF-based scoring

  $S_c = S_c$ * average_unigram_idf

# N-gram Scoring

- Not all terms created equal
  - Usually answers highly specific
  - Also disprefer non-units
- Solution: IDF-based scoring

  $S_c = S_c$ * average_unigram_idf

**After combining**

| Candidate | Score |
|---|---|
| Roger Bannister | 354 |
| Sir Roger Gilbert Bannister | 286 |
| Sir Roger Bannister | 280 |
| Bannister Sir Roger | 278 |
| ... | ... |

# N-gram Scoring

- Not all terms created equal
  - Usually answers highly specific
  - Also disprefer non-units
- Solution: IDF-based scoring

  $S_c = S_c$ * average_unigram_idf

**After combining**

| Candidate | Score |
|---|---|
| Roger Bannister | 354 |
| Sir Roger Gilbert Bannister | 286 |
| Sir Roger Bannister | 280 |
| Bannister Sir Roger | 278 |
| ... | ... |

**After scoring**

| Candidate | Score |
|---|---|
| Roger Bannister | 2377 |
| Englishman Roger Bannister | 1853 |
| Sir Roger Gilbert Bannister | 1775 |
| Sir Roger Bannister | 1768 |
| ... | ... |

# N-gram Reranking

- Promote best answer candidates:

# N-gram Reranking

- Promote best answer candidates:
  - Filter any answers not in at least two snippets

# N-gram Reranking

- Promote best answer candidates:
  - Filter any answers not in at least two snippets
  - Use answer type specific forms to raise matches
    - E.g. ‘where’ -> boosts ‘city, state’

- Small improvement depending on answer type

# Summary

- Redundancy-based approaches
  - Leverage scale of web search
  - Take advantage of presence of ‘easy’ answers on web
  - Exploit statistical association of question/answer text

# Summary

- Redundancy-based approaches
  - Leverage scale of web search
  - Take advantage of presence of 'easy' answers on web
  - Exploit statistical association of question/answer text
- Increasingly adopted:
  - Good performers independently for QA
  - Provide significant improvements in other systems
    - Esp. for answer filtering

# Summary

- Redundancy-based approaches
  - Leverage scale of web search
  - Take advantage of presence of ‘easy’ answers on web
  - Exploit statistical association of question/answer text
- Increasingly adopted:
  - Good performers independently for QA
  - Provide significant improvements in other systems
    - Esp. for answer filtering
- Does require some form of ‘answer projection’
  - Map web information to TREC document

# Deliverable #2

- Baseline end-to-end Q/A system:
  - Redundancy-based with answer projection

  also viewed as
  - Retrieval with web-based boosting

- Implementation: Main components
  - (Suggested) Basic redundancy approach
  - Basic retrieval approach (IR next lecture)

# Data

- Questions:
  - XML formatted questions and question series
- Answers:
  - Answer ‘patterns’ with evidence documents
- Training/Devtext/Evaltest:
  - Training: Thru 2005
  - Devtest: 2006
  - Held-out: …
- Will be in /dropbox directory on patas
- Documents:
  - AQUAINT news corpus data with minimal markup

# PowerAnswer2

- Language Computer Corp.
  - Lots of UT Dallas affiliates
- Tasks: factoid questions
- Major novel components:
  - Web-boosting of results
  - COGEX logic prover
  - Temporal event processing
  - Extended semantic chains
- Results: Best factoid system: 0.713 (vs 0.666, 03.329)

# Challenges: Co-reference

- Single, basic referent:

| Target 27 - *Jennifer Capriati* | |
|---|---|
| Q27.2 | Who is her coach? |
| Q27.3 | Where does she live? |

# Challenges: Co-reference

- Single, basic referent:

| Target 27 - *Jennifer Capriati* | |
| --- | --- |
| Q27.2 | Who is her coach? |
| Q27.3 | Where does she live? |

- Multiple possible antecedents:
  - Depends on previous correct answers

| Target 136 - *Shiite* | |
| --- | --- |
| Q136.1 | Who was the first Imam of the Shiite sect of Islam? |
| Q136.2 | Where is his tomb? |
| Q136.3 | What was this person's relationship to the Prophet Mohammad? |
| Q136.4 | Who was the third Imam of Shiite Muslims? |
| Q136.5 | When did he die? |

# Challenges: Events

- Event answers:
  - Not just nominal concepts

# Challenges: Events

- Event answers:
  - Not just nominal concepts
  - Nominal events:
    - Preakness 1998

# Challenges: Events

- Event answers:
  - Not just nominal concepts
  - Nominal events:
    - Preakness 1998
  - Complex events:
    - Plane clips cable wires in Italian resort

# Challenges: Events

- Event answers:
  - Not just nominal concepts
  - Nominal events:
    - Preakness 1998
  - Complex events:
    - Plane clips cable wires in Italian resort

  - Establish question context, constraints

# Handling Question Series

- Given target and series, how deal with reference?

# Handling Question Series

- Given target and series, how deal with reference?
- Shallowest approach:
  - Concatenation:
    - Add the ‘target’ to the question

# Handling Question Series

- Given target and series, how deal with reference?
- Shallowest approach:
  - Concatenation:
    - Add the 'target' to the question
- Shallow approach:
  - Replacement:
    - Replace all pronouns with target

# Handling Question Series

- Given target and series, how deal with reference?
- Shallowest approach:
  - Concatenation:
    - Add the 'target' to the question
- Shallow approach:
  - Replacement:
    - Replace all pronouns with target
- Least shallow approach:
  - Heuristic reference resolution

# Question Series Results

- No clear winning strategy

# Question Series Results

- No clear winning strategy
  - All largely about the target
    - So no big win for anaphora resolution
    - If using bag-of-words features in search, works fine

# Question Series Results

- No clear winning strategy
  - All largely about the target
    - So no big win for anaphora resolution
    - If using bag-of-words features in search, works fine

  - ‘Replacement’ strategy can be problematic
    - E.g. Target=Nirvana:
    - What is their biggest hit?

# Question Series Results

- No clear winning strategy
  - All largely about the target
    - So no big win for anaphora resolution
    - If using bag-of-words features in search, works fine

  - ‘Replacement’ strategy can be problematic
    - E.g. Target=Nirvana:
    - What is their biggest hit?
    - When was the band formed?

# Question Series Results

- No clear winning strategy
  - All largely about the target
    - So no big win for anaphora resolution
    - If using bag-of-words features in search, works fine

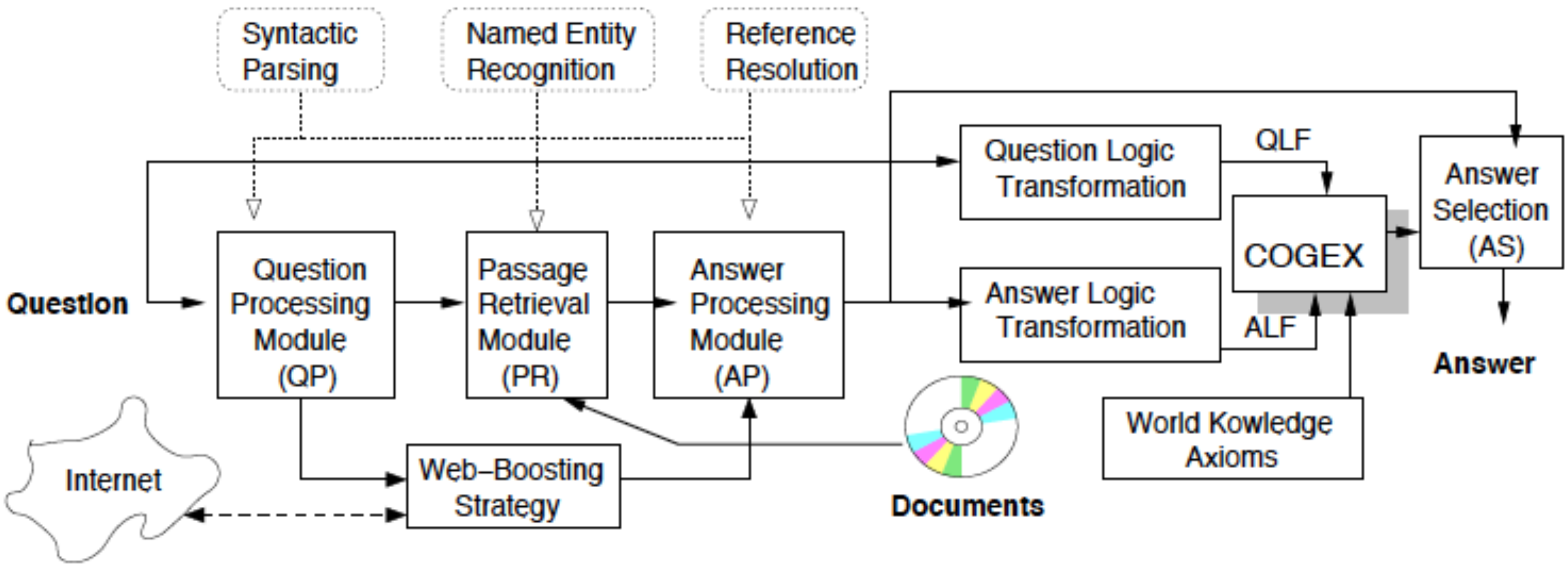
  - ‘Replacement’ strategy can be problematic
    - E.g. Target=Nirvana:
    - What is their biggest hit?
    - When was the band formed?
      - Wouldn’t replace ‘the band’

# Question Series Results

- No clear winning strategy
  - All largely about the target
    - So no big win for anaphora resolution
    - If using bag-of-words features in search, works fine
  - 'Replacement' strategy can be problematic
    - E.g. Target=Nirvana:
    - What is their biggest hit?
    - When was the band formed?
      - Wouldn't replace 'the band'
  - Most teams concatenate

# PowerAnswer-2

- Factoid QA system:



# PowerAnswer-2

- Standard main components:
  - Question analysis, passage retrieval, answer processing

# PowerAnswer-2

- Standard main components:
  - Question analysis, passage retrieval, answer processing
- Web-based answer boosting

# PowerAnswer-2

- Standard main components:
  - Question analysis, passage retrieval, answer processing
- Web-based answer boosting
- Complex components:

# PowerAnswer-2

- Standard main components:
  - Question analysis, passage retrieval, answer processing
- Web-based answer boosting
- Complex components:
  - COGEX abductive prover
  - Word knowledge, semantics:
    - Extended WordNet, etc
  - Temporal processing

# Web-Based Boosting

- Create search engine queries from question

# Web-Based Boosting

- Create search engine queries from question
- Extract most redundant answers from search
  - Cf. Dumais et al – AskMSR; Lin – ARANEA

# Web-Based Boosting

- Create search engine queries from question
- Extract most redundant answers from search
  - Cf. Dumais et al - AskMSR; Lin – ARANEA
- Increase weight on TREC candidates that match
  - Higher weight if higher frequency

# Web-Based Boosting

- Create search engine queries from question
- Extract most redundant answers from search
  - Cf. Dumais et al - AskMSR; Lin – ARANEA
- Increase weight on TREC candidates that match
  - Higher weight if higher frequency
- Intuition:
  - Common terms in search likely to be answer
  - QA answer search too focused on query terms

# Web-Based Boosting

- Create search engine queries from question
- Extract most redundant answers from search
  - Cf. Dumais et al - AskMSR; Lin – ARANEA
- Increase weight on TREC candidates that match
  - Higher weight if higher frequency
- Intuition:
  - Common terms in search likely to be answer
  - QA answer search too focused on query terms
  - Reweighting improves
- Web-boosting improves significantly: 20%

# Deep Processing: Query/Answer Formulation

- Preliminary shallow processing:
  - Tokenization, POS tagging, NE recognition, Preprocess

# Deep Processing: Query/Answer Formulation

- Preliminary shallow processing:
  - Tokenization, POS tagging, NE recognition, Preprocess
- Parsing creates syntactic representation:
  - Focused on nouns, verbs, and particles
    - Attachment

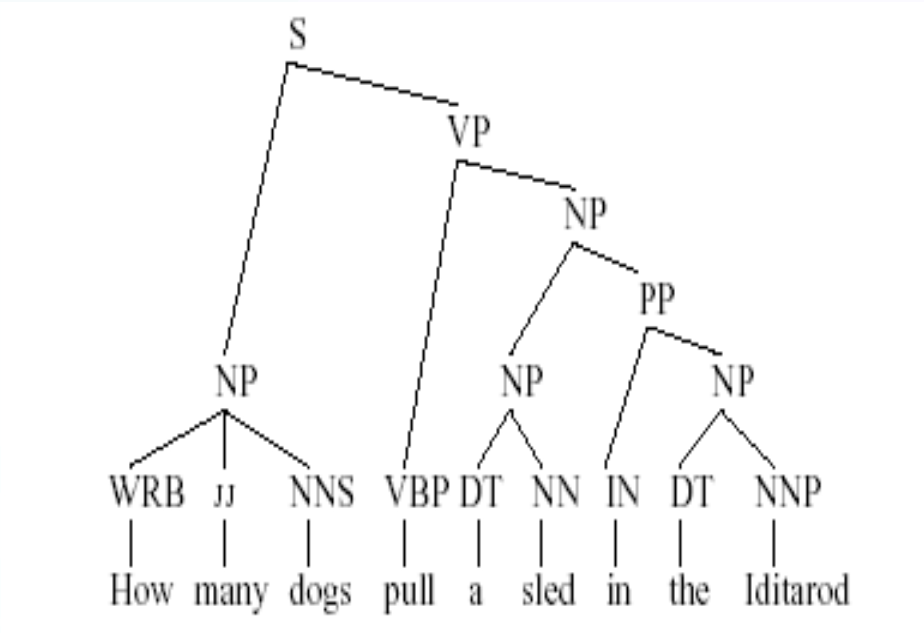
# Deep Processing: Query/Answer Formulation

- Preliminary shallow processing:
  - Tokenization, POS tagging, NE recognition, Preprocess
- Parsing creates syntactic representation:
  - Focused on nouns, verbs, and particles
    - Attachment
- Coreference resolution links entity references

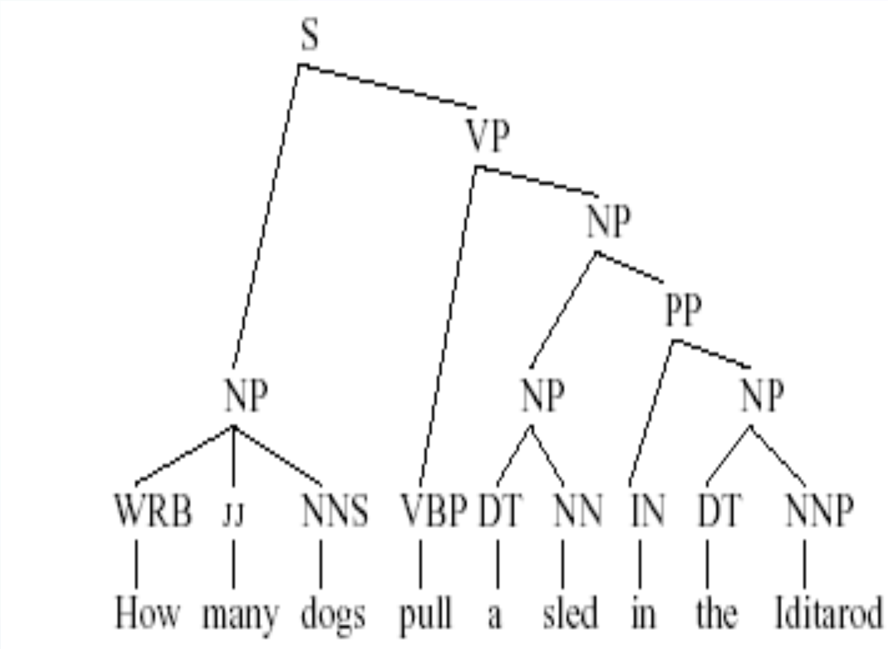
# Deep Processing: Query/Answer Formulation

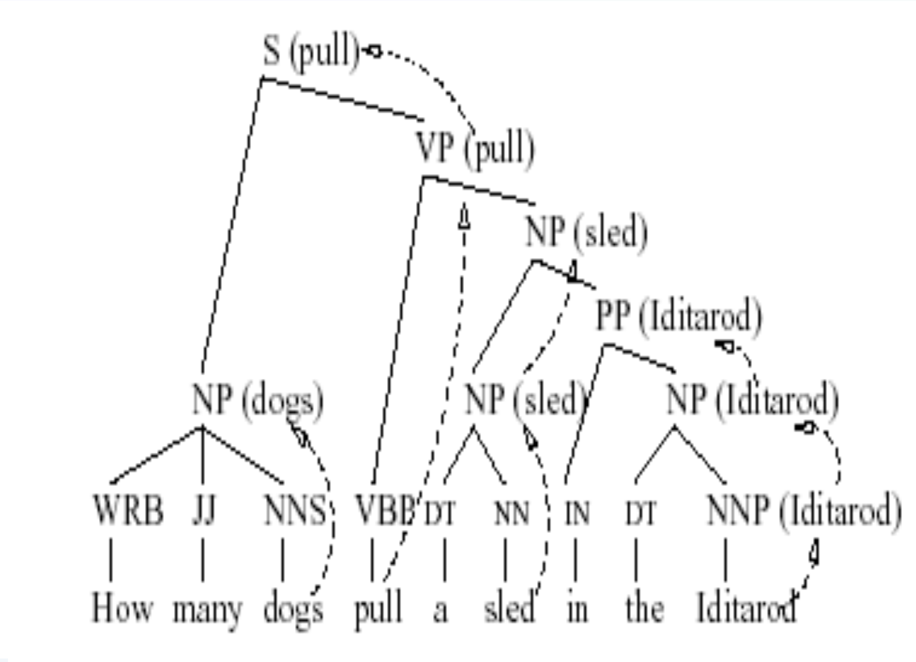
- Preliminary shallow processing:
  - Tokenization, POS tagging, NE recognition, Preprocess
- Parsing creates syntactic representation:
  - Focused on nouns, verbs, and particles
    - Attachment
- Coreference resolution links entity references
- Translate to full logical form
  - As close as possible to syntax

# Syntax to Logical Form

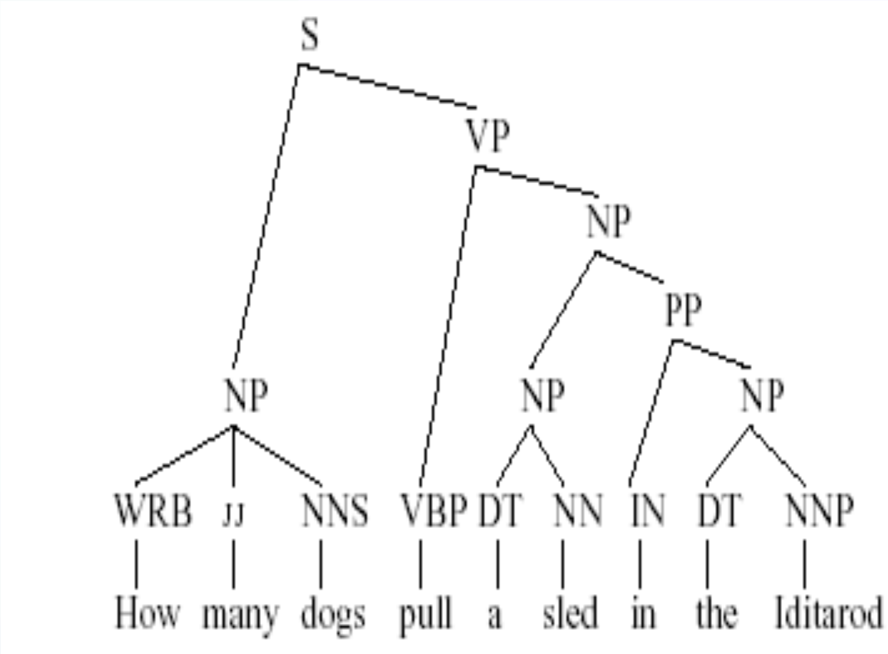


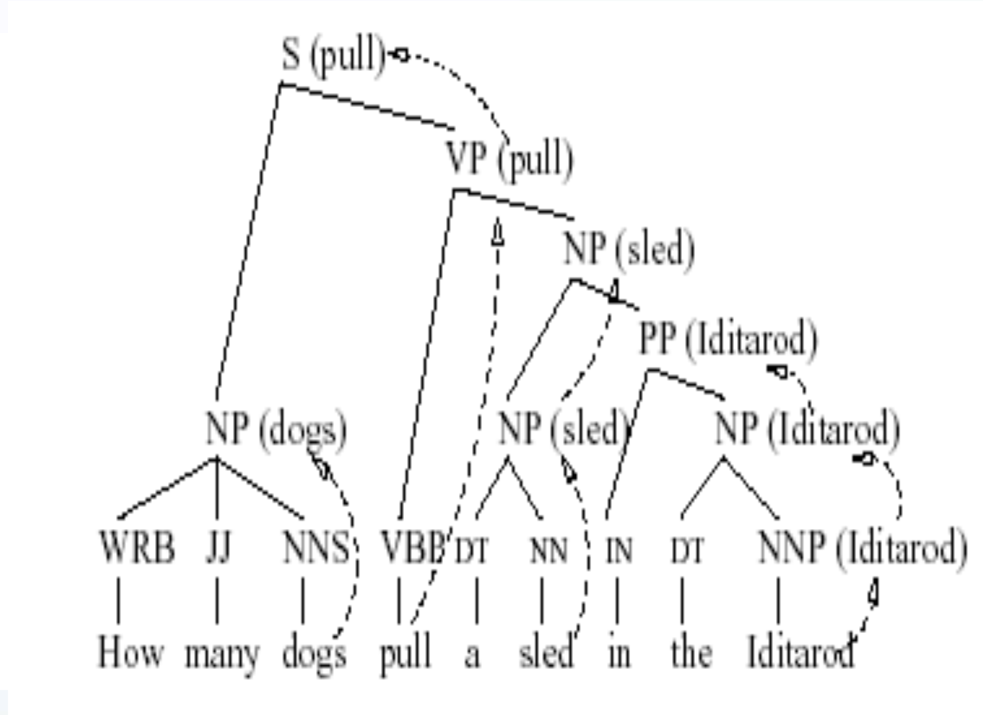
# Syntax to Logical Form

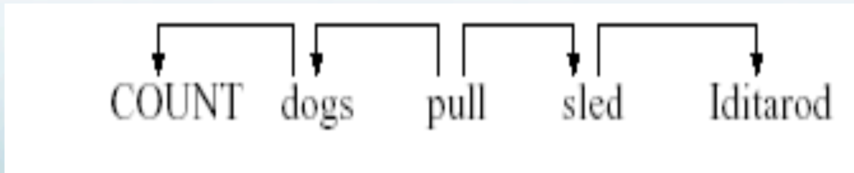


# Syntax to Logical Form



# Deep Processing: Answer Selection

- Cogex prover:
  - Applies abductive inference
    - Chain of reasoning to justify the answer given the question
    - Mix of logical and lexical inference

# Deep Processing: Answer Selection

- Cogex prover:
  - Applies abductive inference
    - Chain of reasoning to justify the answer given the question
    - Mix of logical and lexical inference
- Main mechanism: Lexical chains:
  - Bridge gap in lexical choice b/t Q and A
    - Improve retrieval and answer selection

# Deep Processing: Answer Selection

- Cogex prover:
  - Applies abductive inference
    - Chain of reasoning to justify the answer given the question
    - Mix of logical and lexical inference
- Main mechanism: Lexical chains:
  - Bridge gap in lexical choice b/t Q and A
    - Improve retrieval and answer selection
    - Create connections between synsets through topicality

# Deep Processing: Answer Selection

- Cogex prover:
  - Applies abductive inference
    - Chain of reasoning to justify the answer given the question
    - Mix of logical and lexical inference
- Main mechanism: Lexical chains:
  - Bridge gap in lexical choice b/t Q and A
    - Improve retrieval and answer selection
    - Create connections between synsets through topicality
- *Q: When was the internal combustion engine invented?*
- *A: The first internal-combustion engine was built in 1867.*

# Deep Processing: Answer Selection

- Cogex prover:
  - Applies abductive inference
    - Chain of reasoning to justify the answer given the question
    - Mix of logical and lexical inference
- Main mechanism: Lexical chains:
  - Bridge gap in lexical choice b/t Q and A
    - Improve retrieval and answer selection
    - Create connections between synsets through topicality
- *Q: When was the internal combustion engine invented?*
- *A: The first internal-combustion engine was built in 1867.*
- Yields 12% improvement in accuracy!

# Example

- How hot does the inside of an active volcano get?
- Get(TEMPERATURE, inside(active(volcano)))
- "lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit"
- Fragments(lava,TEMPERATURE(degrees(300)), belched(out, mountain))
  - Volcano ISA mountain; lava ISPARTOF volcano
  - Lava inside volcano
  - Fragments of lava HAVEPROPERTIESOF lava
- Knowledge derived from WordNet to proof 'axioms'

Ex. Due to D. Jurafsky

# Temporal Processing

- 16% of factoid questions include time reference

# Temporal Processing

- 16% of factoid questions include time reference
- Index documents by date: absolute, relative

# Temporal Processing

- 16% of factoid questions include time reference
- Index documents by date: absolute, relative
- Identify temporal relations b/t events
  - Store as triples of (S, E1, E2)
    - S is temporal relation signal – e.g. during, after

# Temporal Processing

- 16% of factoid questions include time reference
- Index documents by date: absolute, relative
- Identify temporal relations b/t events
  - Store as triples of (S, E1, E2)
    - S is temporal relation signal – e.g. during, after
- Answer selection:
  - Prefer passages matching Question temporal constraint
  - Discover events related by temporal signals in Q & As
  - Perform temporal unification; boost good As

# Temporal Processing

- 16% of factoid questions include time reference
- Index documents by date: absolute, relative
- Identify temporal relations b/t events
  - Store as triples of (S, E1, E2)
    - S is temporal relation signal – e.g. during, after
- Answer selection:
  - Prefer passages matching Question temporal constraint
  - Discover events related by temporal signals in Q & As
  - Perform temporal unification; boost good As
- Improves only by 2%
  - Mostly captured by surface forms

# Results

| | PowerAnswer-2 |
|---|---|
| Factoid | 0.713 |
| List | 0.468 |
| Other | 0.228 |
| Overall | 0.534 |

Table 2: Results in the main task.

# Matching Topics and Documents

- Two main perspectives:
  - Pre-defined, fixed, finite topics:
    - “Text Classification”

# Matching Topics and Documents

- Two main perspectives:
  - Pre-defined, fixed, finite topics:
    - "Text Classification"
  - Arbitrary topics, typically defined by statement of information need (aka query)
    - "Information Retrieval"
      - Ad-hoc retrieval

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval
      - Typically: Newspaper story, encyclopedia entry

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval
      - Typically: Newspaper story, encyclopedia entry
      - Alternatively: paragraphs, sentences; web page, site

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval
      - Typically: Newspaper story, encyclopedia entry
      - Alternatively: paragraphs, sentences; web page, site
- Query:
  - Specification of information need

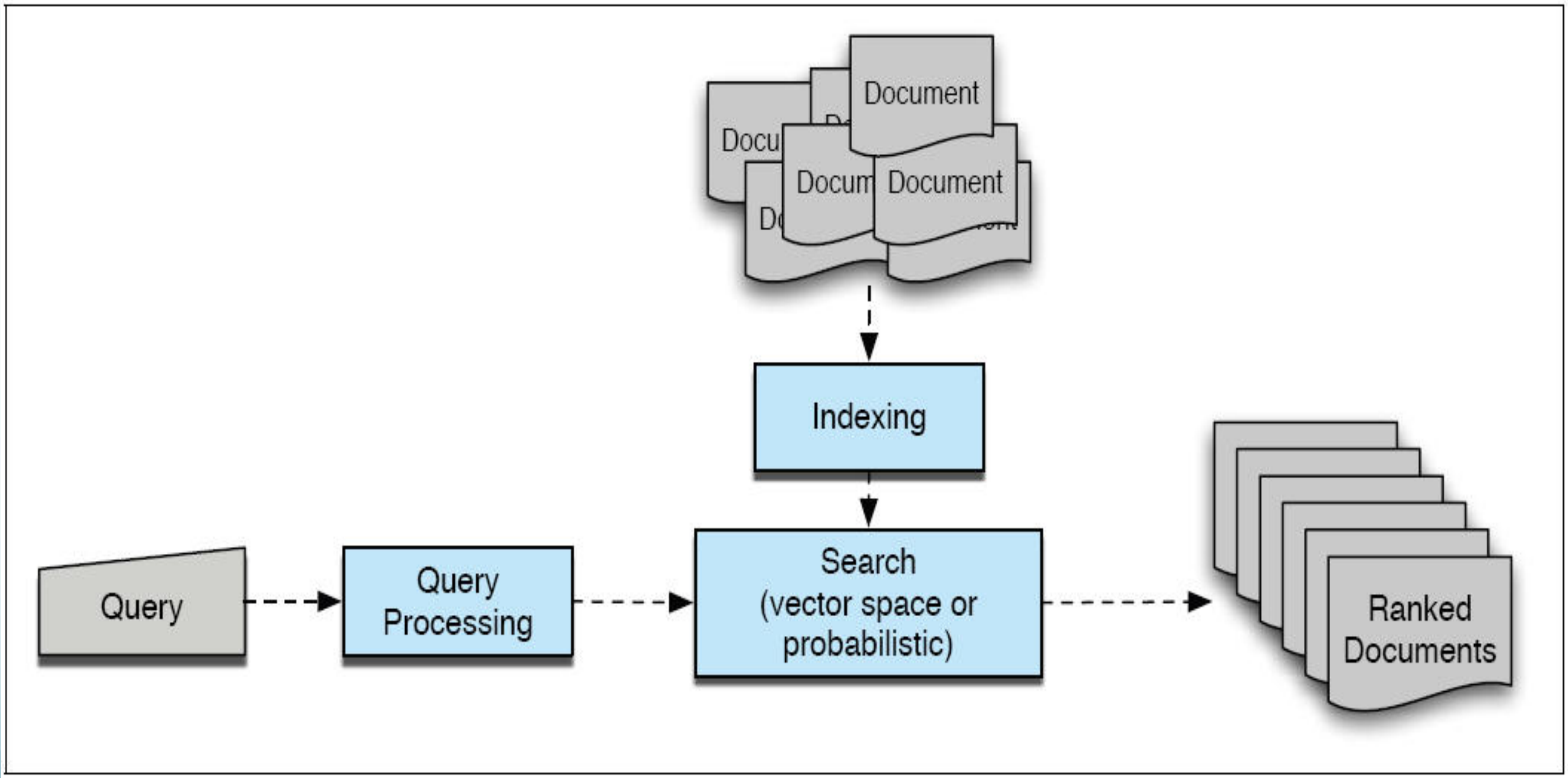
# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval
      - Typically: Newspaper story, encyclopedia entry
      - Alternatively: paragraphs, sentences; web page, site
- Query:
  - Specification of information need
- Terms:
  - Minimal units for query/document

# Information Retrieval Components

- Document collection:
  - Used to satisfy user requests, collection of:
  - Documents:
    - Basic unit available for retrieval
      - Typically: Newspaper story, encyclopedia entry
      - Alternatively: paragraphs, sentences; web page, site
- Query:
  - Specification of information need
- Terms:
  - Minimal units for query/document
    - Words, or phrases

# Information Retrieval Architecture



# Vector Space Model

- Basic representation:
  - Document and query semantics defined by their terms

# Vector Space Model

- Basic representation:
  - Document and query semantics defined by their terms
  - Typically ignore any syntax
    - Bag-of-words  (or Bag-of-terms)
      - Dog bites man == Man bites dog

# Vector Space Model

- Basic representation:
  - Document and query semantics defined by their terms
  - Typically ignore any syntax
    - Bag-of-words (or Bag-of-terms)
      - Dog bites man == Man bites dog

- Represent documents and queries as
  - Vectors of term-based features

# Vector Space Model

- Basic representation:
  - Document and query semantics defined by their terms
  - Typically ignore any syntax
    - Bag-of-words (or Bag-of-terms)
      - Dog bites man == Man bites dog

- Represent documents and queries as
  - Vectors of term-based features
  - E.g. $\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{N,j}); \vec{q}_k = (w_{1,k}, w_{2,k}, \ldots, w_{N,k})$
  - *N:*

# Vector Space Model

- Basic representation:
  - Document and query semantics defined by their terms
  - Typically ignore any syntax
    - Bag-of-words (or Bag-of-terms)
      - Dog bites man == Man bites dog
- Represent documents and queries as
  - Vectors of term-based features
  - E.g. $\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{N,j}); \vec{q}_k = (w_{1,k}, w_{2,k}, \ldots, w_{N,k})$
  - *N:*
    - # of terms in vocabulary of collection: Problem?

# Representation

- Solution 1:
  - Binary features:
    - w=1 if term present, 0 otherwise
  - Similarity:
    - Number of terms in common
    - Dot product

$$sim(\vec{q}_k, \vec{d}_j) = \sum_{i=1}^{N} w_{i,k} w_{i,j}$$

- Issues?

# VSM Weights

- What should the weights be?
- "Aboutness"
  - To what degree is this term what document is about?
  - Within document measure
  - Term frequency (tf): # occurrences of t in doc j
- Examples:
  - Terms: chicken, fried, oil, pepper
  - D1: fried chicken recipe: (8, 2, 7,4)
  - D2: poached chick recipe: (6, 0, 0, 0)
  - Q: fried chicken: (1, 1, 0, 0)

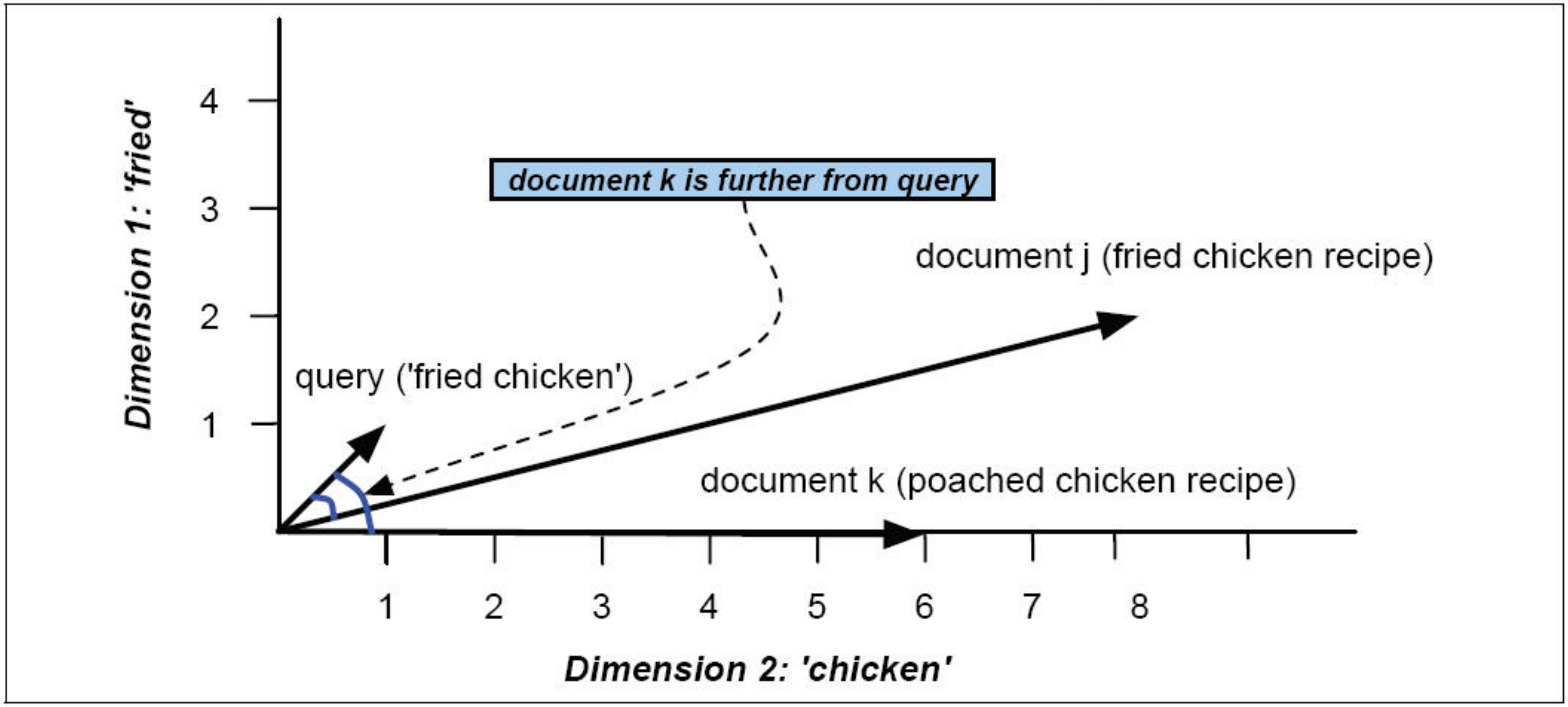
# Vector Space Model (II)

- Documents & queries:
  - Document collection: term-by-document matrix

$$A = \begin{pmatrix} 8 & 6 \\ 2 & 0 \\ 7 & 0 \\ 4 & 0 \end{pmatrix}$$

  - View as vector in multidimensional space
    - Nearby vectors are related
  - Normalize for vector length

# Vector Space Model



# Vector Similarity Computation

- Normalization:
  - Improve over dot product
    - Capture weights
    - Compensate for document length
  - :

# Vector Similarity Computation

- Normalization:
  - Improve over dot product
    - Capture weights
    - Compensate for document length
  - Cosine similarity

$$sim(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,k} w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,k}^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

# Vector Similarity Computation

- Normalization:
  - Improve over dot product
    - Capture weights
    - Compensate for document length
  - Cosine similarity

$$sim(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,k} w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,k}^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

    - Identical vectors:

# Vector Similarity Computation

- Normalization:
  - Improve over dot product
    - Capture weights
    - Compensate for document length
  - Cosine similarity

$$sim(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,k} w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,k}^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

    - Identical vectors: 1
    - No overlap:

# Vector Similarity Computation

- Normalization:
  - Improve over dot product
    - Capture weights
    - Compensate for document length
  - Cosine similarity

$$sim(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,k} w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,k}^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

    - Identical vectors: 1
    - No overlap: 0

# Term Weighting Redux

- “Aboutness”
  - Term frequency (tf): # occurrences of t in doc j

# Term Weighting Redux

- "Aboutness"
  - Term frequency (tf): # occurrences of t in doc j
    - Chicken: 6; Fried: 1 vs Chicken: 1; Fried: 6

# Term Weighting Redux

- “Aboutness”
  - Term frequency (tf): # occurrences of t in doc j
    - Chicken: 6; Fried: 1 vs Chicken: 1; Fried: 6
- Question: what about ‘Representative’ vs ‘Giffords’?

# Term Weighting Redux

- "Aboutness"
  - Term frequency (tf): # occurrences of t in doc j
    - Chicken: 6; Fried: 1 vs Chicken: 1; Fried: 6
- Question: what about 'Representative' vs 'Giffords'?
- "Specificity"
  - How surprised are you to see this term?
  - Collection frequency
  - Inverse document frequency (idf):

$$idf_i = \log\left(\frac{N}{n_i}\right)$$

# Term Weighting Redux

- "Aboutness"
  - Term frequency (tf): # occurrences of t in doc j
    - Chicken: 6; Fried: 1 vs Chicken: 1; Fried: 6
- Question: what about 'Representative' vs 'Giffords'?
- "Specificity"
  - How surprised are you to see this term?
  - Collection frequency
  - Inverse document frequency (idf):

$$idf_i = \log(\frac{N}{n_i}) \qquad w_{i,j} = tf_{i,j} \times idf_i$$

# Tf-idf Similarity

- Variants of tf-idf prevalent in most VSM

$$sim(\vec{q},\vec{d}) = \frac{\sum_{w \in q,d} tf_{w,q} tf_{w,d} (idf_w)^2}{\sqrt{\sum_{q_i \in q} (tf_{q_i,q} idf_{q_i})^2} \sqrt{\sum_{d_i \in d} (tf_{d_i,d} idf_{d_i})^2}}$$

# Term Selection

- Selection:
  - Some terms are truly useless

# Term Selection

- Selection:
  - Some terms are truly useless
    - Too frequent:
      - Appear in most documents

# Term Selection

- Selection:
  - Some terms are truly useless
    - Too frequent:
      - Appear in most documents
    - Little/no semantic content

# Term Selection

- Selection:
  - Some terms are truly useless
    - Too frequent:
      - Appear in most documents
    - Little/no semantic content
      - Function words
        - E.g. the, a, and,…

# Term Selection

- Selection:
  - Some terms are truly useless
    - Too frequent:
      - Appear in most documents
    - Little/no semantic content
      - Function words
        - E.g. the, a, and,…
  - Indexing inefficiency:
    - Store in inverted index:
      - For each term, identify documents where it appears
      - 'the': every document is a candidate match

# Term Selection

- Selection:
  - Some terms are truly useless
    - Too frequent:
      - Appear in most documents
    - Little/no semantic content
      - Function words
        - E.g. the, a, and,…
  - Indexing inefficiency:
    - Store in inverted index:
      - For each term, identify documents where it appears
      - 'the': every document is a candidate match
- Remove 'stop words' based on list
  - Usually document-frequency based

# Term Creation

- Too many surface forms for same concepts

# Term Creation

- Too many surface forms for same concepts
  - E.g. inflections of words: verb conjugations, plural
    - Process, processing, processed
    - Same concept, separated by inflection

# Term Creation

- Too many surface forms for same concepts
  - E.g. inflections of words: verb conjugations, plural
    - Process, processing, processed
    - Same concept, separated by inflection
- Stem terms:
  - Treat all forms as same underlying
    - E.g., ‘processing’ -> ‘process’; ‘Beijing’ -> ‘Beije’
- Issues:

# Term Creation

- Too many surface forms for same concepts
  - E.g. inflections of words: verb conjugations, plural
    - Process, processing, processed
    - Same concept, separated by inflection

- Stem terms:
  - Treat all forms as same underlying
    - E.g., ‘processing’ -> ‘process’; ‘Beijing’ -> ‘Beije’
- Issues:
  - Can be too aggressive
    - AIDS, aids -> aid; stock, stocks, stockings -> stock

# Evaluating IR

- Basic measures:  Precision and Recall

# Evaluating IR

- Basic measures:  Precision and Recall
- Relevance judgments:
  - For a query, returned document is relevant or non-relevant
    - Typically binary relevance: 0/1

# Evaluating IR

- Basic measures:  Precision and Recall
- Relevance judgments:
  - For a query, returned document is relevant or non-relevant
    - Typically binary relevance: 0/1
  - T: returned documents; U: true relevant documents
  - R: returned relevant documents
  - N: returned non-relevant documents

# Evaluating IR

- Basic measures:  Precision and Recall
- Relevance judgments:
  - For a query, returned document is relevant or non-relevant
    - Typically binary relevance: 0/1
  - T: returned documents; U: true relevant documents
  - R: returned relevant documents
  - N: returned non-relevant documents

$$\Pr ecision = \frac{|R|}{|T|}; \operatorname{Re} call = \frac{|R|}{|U|}$$

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: 'best' first
  - 10 relevant documents returned:

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first
  - 10 relevant documents returned:
    - In first 10 positions?

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first
  - 10 relevant documents returned:
    - In first 10 positions?
    - In last 10 positions?

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first
  - 10 relevant documents returned:
    - In first 10 positions?
    - In last 10 positions?
    - Score by precision and recall – which is better?

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first
  - 10 relevant documents returned:
    - In first 10 positions?
    - In last 10 positions?
    - Score by precision and recall – which is better?
      - Identical !!!
      - Correspond to intuition?  NO!

# Evaluating IR

- Issue: Ranked retrieval
  - Return top 1K documents: ‘best’ first
  - 10 relevant documents returned:
    - In first 10 positions?
    - In last 10 positions?
    - Score by precision and recall – which is better?
      - Identical !!!
      - Correspond to intuition?  NO!

- Need rank-sensitive measures

# Rank-specific P & R

| Rank | Judgment | $Precision_{Rank}$ | $Recall_{Rank}$ |
|---|---|---|---|
| 1 | R | 1.0 | .11 |
| 2 | N | .50 | .11 |
| 3 | R | .66 | .22 |
| 4 | N | .50 | .22 |
| 5 | R | .60 | .33 |
| 6 | R | .66 | .44 |
| 7 | N | .57 | .44 |
| 8 | R | .63 | .55 |
| 9 | N | .55 | .55 |
| 10 | N | .50 | .55 |
| 11 | R | .55 | .66 |
| 12 | N | .50 | .66 |
| 13 | N | .46 | .66 |
| 14 | N | .43 | .66 |
| 15 | R | .47 | .77 |
| 16 | N | .44 | .77 |
| 17 | N | .44 | .77 |
| 18 | R | .44 | .88 |
| 19 | N | .42 | .88 |
| 20 | N | .40 | .88 |
| 21 | N | .38 | .88 |
| 22 | N | .36 | .88 |
| 23 | N | .35 | .88 |
| 24 | N | .33 | .88 |
| 25 | R | .36 | 1.0 |

# Rank-specific P & R

- $\text{Precision}_{rank}$: based on fraction of reldocs at rank
- $\text{Recall}_{rank}$: similarly

# Rank-specific P & R

- $\text{Precision}_{rank}$: based on fraction of reldocs at rank
- $\text{Recall}_{rank}$: similarly
- Note: Recall is non-decreasing; Precision varies

# Rank-specific P & R

- $\text{Precision}_{rank}$: based on fraction of reldocs at rank
- $\text{Recall}_{rank}$: similarly
- Note: Recall is non-decreasing; Precision varies
- Issue: too many numbers; no holistic view

# Rank-specific P & R

- $Precision_{rank}$: based on fraction of reldocs at rank
- $Recall_{rank}$: similarly
- Note: Recall is non-decreasing; Precision varies
- Issue: too many numbers; no holistic view
  - Typically, compute precision at 11 fixed levels of recall
  - Interpolated precision:

$$IntPrecision(r) = \max_{i>=r} Precision(i)$$

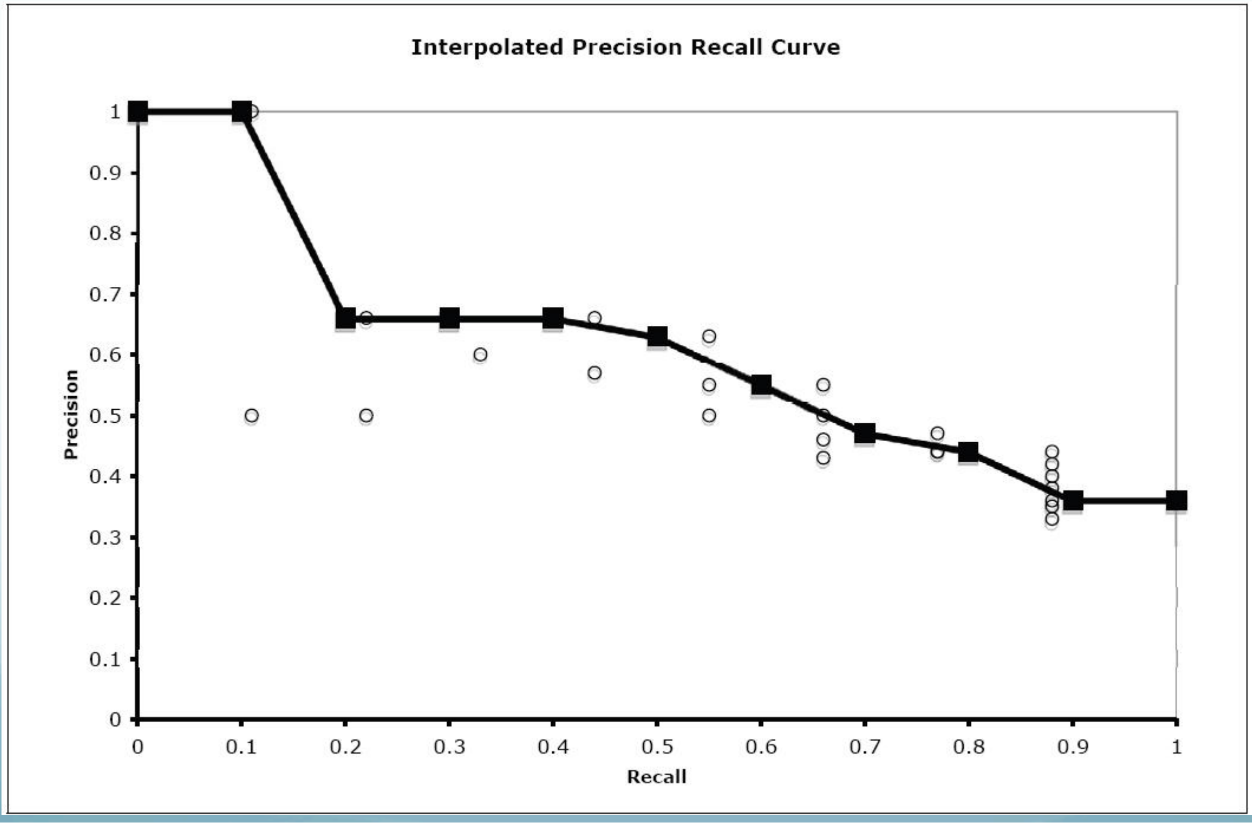
    - Can smooth variations in precision

# Interpolated Precision

| Interpolated Precision | Recall |
|---|---|
| 1.0 | 0.0 |
| 1.0 | .10 |
| .66 | .20 |
| .66 | .30 |
| .66 | .40 |
| .63 | .50 |
| .55 | .60 |
| .47 | .70 |
| .44 | .80 |
| .36 | .90 |
| .36 | 1.0 |

# Comparing Systems

- Create graph of precision vs recall
  - Averaged over queries
  - Compare graphs



# Mean Average Precision (MAP)

- Traverse ranked document list:
  - Compute precision each time relevant doc found

# Mean Average Precision (MAP)

- Traverse ranked document list:
  - Compute precision each time relevant doc found
    - Average precision up to some fixed cutoff
    - $R_r$: set of relevant documents at or above r
    - Precision(d) : precision at rank when doc d found

$$\frac{1}{|R_r|} \sum_{d \in R_r} \Pr ecision_r(d)$$

# Mean Average Precision (MAP)

- Traverse ranked document list:
  - Compute precision each time relevant doc found
    - Average precision up to some fixed cutoff
    - $R_r$: set of relevant documents at or above r
    - Precision(d) : precision at rank when doc d found

$$\frac{1}{|R_r|}\sum_{d \in R_r} \mathrm{Pr}\,ecision_r(d)$$

- Mean Average Precision: 0.6
    - Compute average over all queries of these averages

# Mean Average Precision (MAP)

- Traverse ranked document list:
  - Compute precision each time relevant doc found
    - Average precision up to some fixed cutoff
    - $R_r$: set of relevant documents at or above r
    - Precision(d) : precision at rank when doc d found

$$\frac{1}{|R_r|}\sum_{d \in R_r} \mathrm{Pr}\,ecision_r(d)$$

- Mean Average Precision: 0.6
    - Compute average of all queries of these averages
    - Precision-oriented measure

# Mean Average Precision (MAP)

- Traverse ranked document list:
  - Compute precision each time relevant doc found
    - Average precision up to some fixed cutoff
    - $R_r$: set of relevant documents at or above r
    - Precision(d) : precision at rank when doc d found

$$\frac{1}{|R_r|}\sum_{d \in R_r} \mathrm{Pr}\, ecision_r(d)$$

- Mean Average Precision: 0.6
    - Compute average of all queries of these averages
    - Precision-oriented measure
- Single crisp measure: common TREC Ad-hoc

# Roadmap

- Retrieval systems
- Improving document retrieval
  - Compression & Expansion techniques
- Passage retrieval:
  - Contrasting techniques
  - Interactions with document retreival

# Retrieval Systems

- Three available systems
  - Lucene: Apache
    - Boolean systems with Vector Space Ranking
    - Provides basic CLI/API (Java, Python)

  - Indri/Lemur: Umass /CMU
    - Language Modeling system  (best ad-hoc)
    - ‘Structured query language
      - Weighting,
    - Provides both CLI/API (C++,Java)

  - Managing Gigabytes (MG):
    - Straightforward VSM

# Retrieval System Basics

- Main components:
  - Document indexing
    - Reads document text
      - Performs basic analysis
        - Minimally – tokenization, stopping, case folding
        - Potentially stemming, semantics, phrasing, etc
    - Builds index representation

# Retrieval System Basics

- Main components:
  - Document indexing
    - Reads document text
      - Performs basic analysis
        - Minimally – tokenization, stopping, case folding
        - Potentially stemming, semantics, phrasing, etc
    - Builds index representation
  - Query processing and retrieval
    - Analyzes query (similar to document)
      - Incorporates any additional term weighting, etc
    - Retrieves based on query content
      - Returns ranked document list

# Example (I/L)

- indri-5.0/buildindex/IndriBuildIndex parameter_file
  - XML parameter file specifies:
    - Minimally:
      - Index: path to output
      - Corpus (+): path to corpus, corpus type
    - Optionally:
      - Stemmer, field information
- indri-5.0/runquery/IndriRunQuery query_parameter_file -count=1000 \

  -index=/path/to/index -trecFormat=true > result_file

  Parameter file: formatted queries w/query #

# Lucene

- Collection of classes to support IR
  - Less directly linked to TREC
    - E.g. query, doc readers
- IndexWriter class
  - Builds, extends index
  - Applies analyzers to content
    - SimpleAnalyzer: stops, case folds, tokenizes
    - Also Stemmer classes, other langs, etc
- Classes to read, search, analyze index
- QueryParser parses query (fields, boosting, regexp)