Indexing

The task of finding terms that describe documents well.

Manual vs. automatic indexing: how to manipulate and weight terms?

Vocabularies:

- Fixed:
  - High training effort
  - Out of band scheme changes
  - High precision searches that work well for closed collections (e.g. library)
- Free:
  - Index term set inferred, not predefined
  - Must deal with synonymy and polysemy, capitalisation, stemming
  - Might use phrases rather than simple words

Inverted files: mapping of search terms to document IDs, occurrence frequency term offset

Boolean Model:

- Presence of a term in a document is necessary and sufficient for match
- Does not typically provide for ranking: relevance is a binary decision

Vector Space Model:

- Documents and queries represented in vector space whose dimensions are terms
- Proximity measures:
  - Must be symmetric and minimal for identity
  - Distance measures are additionally non-negative and have the triangle property
  - Binary:
    - $rawoverlap(X, Y) = |X \cap Y|$
    - $dice(X, Y) = \frac{2|X \cap Y|}{|X|+|Y|}$
    - $jacc(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$
    - $overlapcoefficient(X, Y) = \frac{|X \cap Y|}{min(|X|,|Y|)}$
    - $cosine(X, Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}$
  - Weighted:
    - Dice and Jaccard measures don't work well for IR due to the use of sparse vectors with differing lengths
    - $\overrightarrow{w_i}$is a vector of terms occurring in document i, t is the number of index terms and $w_{i,j}$ is the weight of term j in the terms of document i
    - $cos(\overrightarrow{w_i}, \overrightarrow{w_k}) = \frac{\overrightarrow{w_i}\overrightarrow{w_k}}{|\overrightarrow{w_i}||\overrightarrow{w_k}|}$ where $||$ denotes the distance metric rather than vector length
    - $euclidean(\overrightarrow{w_i}, \overrightarrow{w_k}) = \sqrt{\sum_{j=1}^{d}(w_{i,j} - w_{k,j})^2}$
    - $manhattan(\overrightarrow{w_i}, \overrightarrow{w_k}) = \sum_{j=1}^{d} |w_{i,j} - w_{k,j}|$
  - TF*IDF:
    - $freq_{w,d}$ is the frequency of word w in document d
    - $n_{w,D}$ is the number of documents in the collection D that contain word w
    - $idf_{w,D} = \log \frac{|D|}{n_{w,D}}$
    - $tf_{w,d} = freq_{w,d}$
    - $(tf * idf)_{w,d,D} = tf_{w,d} idf_{w,D}$
    - $tf_{norm,w,d} = \frac{freq_{w,d}}{max_{l \in d} freq_{l,d}}$
    - $(tf * idf)_{norm,w,d,D} = tf_{norm,w,d} idf_{w,D}$

Zipf's law:

- $frequency(word_i) = \frac{1}{i^\theta} frequency(word_1)$ where $word_i$is the word with the ith rank and typically $1.5 < \theta < 2$
- Mid-frequency words are the best indicators of what the document is about

Porter stemmer:

- Works without a dictionary and deals with inflectional and derivational morphology, but not with root changes
- Parse words as $[C](VC)\{m\}[V]$ where m is called the measure
- Rules:
  - Are of the form $(condition)S \rightarrow S'$ where S is a suffix
  - Conditions can be on measure excluding the suffix of the rule under consideration, the shape of the word piece or involve logical operations

Evaluation

Goal is to test system parameters (methods of term choice, matching algorithms etc) while ignoring environment variables (particular documents and users).

Test collections consist of:

- A large document set
- Queries or topics with a short description of the information need

- Relevance judgments, ideally made by the person who created the query

Recall: $\frac{|relevant \cap retrieved|}{|relevant|}$

Precision: $\frac{|relevant \cap retrieved|}{|retrieved|}$

Accuracy: $\frac{|relevant \cap retrieved| + |notretrieved \cap notrelevant|}{|everything|}$ (conflates two measures)

Typically recall rises with document retrieved and precision falls, so a graph of one against the other describes a curved frontier

Most tasks are precision-critical (e.g. in web search where there is much redundant information) but other tasks can be accuracy-critical (e.g. patent search)

Relevance is subjective and situational (e.g. due to novelty): this is mitigated by the use of guidelines and extensive sampling

Pooling:

- Allows relevance judgments to be made on very large document collections
- The top N results from the systems under test are pooled together along with those gathered from unrestricted manual runs
- Relevance judgments are made on this pool, with those outside the pool considered irrelevant

Measures:

- Sensible precision/recall data points are those after each relevant document has been seen
- $F_\alpha = \frac{PR}{(1-\alpha)P + \alpha R}$, commonly used with $\alpha = \frac{1}{2}$
- $MAP = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{Q_j} \sum_{i=1}^{Q_j} P(doc_i)$ where Q_j is the number of relevant documents for query j, N is the number of queries, $P(doc_i)$ is the precision at the ith relevant document
- $P_{11pt} = \frac{1}{11} \sum_{j=0}^{10} \frac{1}{N} \sum_{i=1}^{N} \tilde{P}_i(r_j)$ where $r_j = \frac{j}{10}$ and $\tilde{P}_i(r_j) = \begin{cases} max(r_j \leq r \leq r_{j+1}) P_i(R = r) & \text{if } P_i(R = r) \text{ exists} \\ \tilde{P}_i(r_{j+1}) & \text{otherwise} \end{cases}$

Search Engines

Backlink counting:

- Return the pages with the most backlinks
- Does not take account of page quality

PageRank:

- Simulation of a random surfer, where links are followed randomly with probability q and occasionally jumps are made to another random page with probability $1 - q$
- Rank sources are added to counteract rank sinks (pages with no outgoing links)
- $\overrightarrow{r} = c(qA + \frac{1-q}{N}\mathbf{1})\overrightarrow{r}$ where $A_{uv} = \begin{cases} \frac{1}{N_v} & \text{if } \exists v \to u \\ 0 & \text{otherwise} \end{cases}$ and $\overrightarrow{r}$ is the PageRank vector
- Iterate to find $\overrightarrow{r}$ and terminate when the magnitude of the movement falls below a threshold
- There is a difference between linking behavior and actual usage data

Named Entity Recognition

ENAMEX (person/organization/location), TIMEX (time/date) etc

Approaches include regular expressions, name gazetteers (although these are impractical for last names, due to productiveness, overlap with nouns/verbs/adjectives and ambiguity of name types)

Cascading NER:

- Use machine learning to decide the type of NE, using the internal phrase structure of a name (substrings, suffixes, prefixes)
- Make high precision decisions first
- Assume one name type per discourse
- Mikheev Algorithm:
  1. Apply sure fire rules to tag definite NEs
  2. Use machine learning to find variants of the tagged names in the entire text
  3. Apply relaxed rules to tag further NEs
  4. Use machine learning again to find new variants
  5. Apply specialized maximum-entropy model to the title

Information Extraction

Lexico-semantic patterns:

- "Flattened-out" semantic representations with lexemes hard-wired into them

- Matching is string based, with pattern generalization being accomplished only by hand

Template mining:

- Apply hard (selectional) and soft (semantic preferences) constraints for slot fillers
- Given filled template plus raw text find trigger words and enabling conditions
- Heuristics take the form of syntactic patterns (e.g. active-verb prep <np>) which are filled with trigger words by the algorithm
- Typically followed by human evaluation of suggested patterns

Snowball

Bootstrapped algorithm which makes use of a working table of <organization, location> tuples (<o, l> tuples). To begin, the table is seeded with a number of examples by humans who have inspected the corpus and a named entity recognizer is run over the corpus. Finally, an algorithm is launched to iteratively discover patterns which let it fill in the table further:

1. Identify occurrences of the example <o, l> tuples it knows about so far in the documents
2. Extraction patterns identified based on the found contexts which contain known <o, l> tuples
3. Patterns are evaluated to find the best ones, and used to select new <o, l> tuples
4. New tuples are evaluated and if they are found to be reliable they are entered into the table of known <o, l> tuples
5. Repeat from step 1 unless no new sufficiently reliable <o, l> tuples are found in step 4

Occurrences of known tuples are found by scanning the input documents for instances of the words that occur within some fixed number of words from each other (e.g. 10 words). The context is then changed into a pattern of the form <left, tag1, middle, tag2, right>, where the tags are either "location" or "organization". The left, middle and right parts are lists of words with associated weights (assigned as a function of frequency of the word in context). These vectors are normalized and then scaled by constants W_{left},W_{middle} and W_{right}.

Degree of matching between two tuples can be defined for two tuples with the same words and tags as the sum of inner products of the word weightings: we call this $Match(P_1, P_2)$. Given this matching formula, the vectors can be clustered by a minimum similarity threshold τ_{sim}, to obtain cluster centroids which are word weighting vectors. These, combined with the original tags, are used to form the generalized pattern, which is further associated with the contexts it came from and how far each context was from the centroid.

In order to evaluate these generalized patterns, we seek productive and reliable patterns. Productivity is enforced by discarding patterns with less than τ_{sup}tuples in support (in the corresponding cluster). Reliability is estimated by computing the ratio of the number of times the pattern produces evidence that agrees with what is in our database (times the log of itself, to favor productive patterns) with the number of times the pattern produces any tuple. This is further normalized. This normalized confidence measure of a pattern P is called $Conf(P)$. If desired, patterns with low confidence can then be discarded. We can then define the confidence of a tuple T:

$$Conf(T) = 1 - \prod_{i=0}^{|P|}(1 - Conf(P_i)Match(C_i, P_i))$$

Where:

- $P = \{P_i\}$ is the set of patterns that generated the tuple
- C_i is the context associated with an occurrence of T for P_i
- $Match(C_i, P_i)$ is the goodness of a match between P_i and C_i

This works due to the assumption that for a tuple T to be valid at least one pattern must match the currently discovered text of T. The final step in the algorithm is to discard those tuples with confidence less than some threshold (say τ_t), and then update a confidence attached to every pattern in the table by the formula:

$$Conf_{i+1}(P) = Conf(P)\frac{1}{2} + Conf_i(P)\frac{1}{2}$$

Where i records the iteration number, reflecting the fact that the confidence of a pattern we generate via this algorithm will alter with every iteration as we consider new evidence.

Question Answering

Evaluation:

- Wrong answers are OK if they are supported by the text
- Ambiguous answers are considered incorrect
- Mean Reciprocal Rank (MRR): look at top five answers, per question score $RR_i$is $\frac{1}{r_i}$where $r_i$is the rank at which first correct answer occurs, then MRR is mean of this over all documents i
- Accuracy: $\frac{|SubmittedAnswers \cap CorrectAnswers|}{|CorrectAnswers|}$

- Confidence Weighted Score: $\frac{1}{Q}\sum_{i=1}^{Q}\frac{|Correct \cap AnswersUpTo_i|}{i}$ where Q is the number of questions and $AnswersUpTo_i$ is the i question answers with highest confidence

Expected Answer Type:

- Many questions expect a NE as their answer, so try and identify that
- FST rules for expected answer type, e.g. Name NP(city|country|company) → CITY|COUNTRY|COMPANY
- Direct matching of question words, e.g. who/whom → PERSON

SMU System:

- Parse question for keywords, answer type, question predicate structure
- Index into documents with keywords, refine results by answer type and then compare to predicate structure
- Feedback loops at every stage
  - Morphological: add keywords to query based on answer type, e.g. if includes "who" with "invent" add "inventor"
  - Lexical: add keywords based on relationships such as synonymy, e.g. add "distance" if question asks how "far" something is
  - Semantic alternations and paraphrases

MS System:

- Deep reasoning only necessary if search ground is restricted
- Minimal processing: remove question words, add morphological variations, reorder words
- Query generation linearly moves the finite verb (e.g. is) through the query and outputs a confidence and position in the text where the answer is expected
- Sum weighted n-grams from Google query and then combine similar ones
- Back project answers into the TREC corpus by using a traditional IR engine

Summarization

Informative/indicative and abstract (generated)/extract dimensions.

Ideally truth preserving and coherent

Deep model: go via semantic representation of full text, but bottleneck in text analysis

Fact extraction:

- Reason with templates which are generated by combining others
- The most important template is chosen for generation
- Template combination/modification rules:
  - Change of perspective: if information from a source conflicts over time report both pieces
  - Contradiction: conflicting information resolved by choosing that from independent sources
  - Addition: include additional information from later articles
  - Refinement: prefer specific information
  - Agreement/no information: report this to give reader information
  - Superset: combine incomplete information from many sources
  - Trend: report similar patterns over time as one statement
- Limited by domain dependence of templates and rules

Text extraction:

- Split text into units which are assigned importance using sentential/relational features
- Those units with the highest score are extracted verbatim
- Sentential features:
  - The number of concepts in a unit (according to $tf * idf$)
  - The number of concepts occurring in the title that appear in a unit
  - How close the text is to the top
  - Give extra weight to the first sentence in a paragraph
  - Sentence length
- Feature combination:
  - Manual weighting of feature scores

- Naive-Bayes classification: $P(s \in S|F_1, \ldots, F_k) \approx \frac{P(s \in S) \prod_{j=1}^{k} P(F_j|s \in S)}{\prod_{j=1}^{k} P(F_j)}$ where S is the summary, s is the sentence and F_i are the features chosen

Problems with coherence (e.g. dangling anaphora, but can be fixed by resolution and inclusion/discarding) and cohesion (e.g. concepts and agents are not introduced, narrative flow is lost)

Evaluation:

- Solicit subjective judgments: need to ensure all subjects understand the same judging criteria
- Gold standard comparison: unfortunately humans do not agree on what a good summary is
- Task-based evaluation: how well can humans perform a task with this summary? Expensive!