APPLICATIONS 8:

AUTOMATED TEXT SUMMARIZATION

Theme

Types of text summaries. The three steps. Topic identification methods. Multi-document summarization. Summary evaluation.

Summary of Contents

Why summarization? Because of the flood of information: In 2002, film, magnetic, and optical storage media produced about **5 exabytes (10^{18})** of new information. **92%** of the new information was stored on magnetic media, mostly in hard disks. To put this in context: if digitized with full formatting, the 17 million books in the Library of Congress contain about 136 terabytes of information; so **5 exabytes** of information is equivalent in size to the information contained in **37,000 new libraries** the size of the Library of Congress book collections! The amount of new information stored on paper, film, magnetic, and optical media has about **doubled** during 1999 – 2002 (see www2.sims.berkeley.edu/research/projects/how-much-info-2003/).

And that was in 2002.

In 2009, Jamie Callen from CMU crawled a sizeable portion of the web and created the ClueWeb09 text corpus, which is available at http://boston.lti.cs.cmu.edu/Data/clueweb09/ to qualifying people, for research purposes only. It contains 1 billion web pages (5TB compressed), in ten languages, collected in January and February 2009.

1. Types of Summary

Exercise: read a short text and summarize it (see last two pages). What makes this hard to do? For one thing, you don't know what the reader needs or wants. For generic (author's point of view) summarization you have to assume you can infer what is important to the author. But often the reader has some specific goal in mind, and would prefer a summary that speaks to that goal. Sometimes, your summary doesn't even need to consist of full sentences; perhaps some keywords, or even just a picture or a diagram, may be enough.

To know what summary to produce, you need more parameters: there is no one best summary. An initial typology was created by Karen Spärck Jones, and taken further by Hovy and Mani:

Input: characteristics of the source text(s)

- Source size: single-document *vs*. multi-document
- Specificity: domain-specific *vs*. general
- Genre and scale
- Language: monolingual *vs*. multilingual

Output: characteristics of the summary as a text

- Derivation: **Extract** *vs*. **abstract**: An extract is a collection of passages (ranging from single words to whole paragraphs) extracted from the input text(s) and produced verbatim as the summary. An abstract is a newly generated text, produced from some computer-internal representation that results after analysis of the input.

- Coherence: fluent *vs*. dysfluent
- Partiality: neutral *vs*. evaluative
- Conventionality: fixed *vs*. floating: A fixed-situation summary is created for a specific use, reader (or class of reader), and situation.

Purpose: characteristics of the summary usage

- Audience: **Generic** *vs*. **query-oriented**: A generic summary provides the author's point of view of the input text(s), giving equal import to all major themes in it. A query-oriented (or user-oriented) summary favors specific themes or aspect(s) of the text, in response to a user's desire to learn about just those themes in particular.
- Usage: **Indicative** *vs*. **informative**: An indicative summary provides merely an indication of the principal subject matter or domain of the input text(s) without including its contents. After reading an indicative summary, one can explain what the input text was about, but not necessarily what was contained in it. An informative summary reflects (some of) the content, and allows one to describe (parts of) what was in the input text.
- Expansiveness: Background *vs*. just-the-news

2. Methods of Summarization

2.1 Stages of Summarization Systems

To create a summary, you have to evaluate each portion (paragraph, sentence, word) of the text and decide whether to keep it or not. You may then reformulate what you have selected in a coherent form, and must then output it. Conceptually, the process consists of three stages:

1. **Topic identification**: identify the material to keep
2. **Interpretation/compaction**: combine and compress it, as far as possible
3. **Generation**: output it in the format required

Extraction systems perform the first step only; abstraction systems the first two, and usually also the third.

Topic Identification: Most systems concentrate on this, the first stage; many do not even perform the other two. The task is simply to assign importance scores to each portion (sentence, usually) of the text, rank the portions, and return the top *N*% of them (probably in document order rather than rank order, for coherence).

Extract-type summaries are relatively easy to produce and work surprisingly well for certain genres (news articles, for example). Their major problems are:

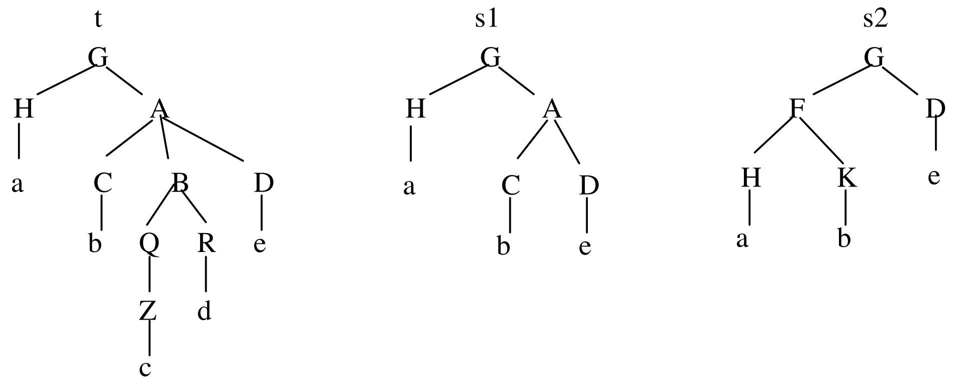
- lack of coherence: pronouns (*he*/*she*/*it*/*this*/*that*/etc.) may refer back to sentences that are not included in the extract
- unintended implications that may arise when two non-contiguous sentences are juxtaposed:

  Many people were fooled by the salesman's smooth talk. [But not everyone.] Mr. Barker, for example. He…

The principal topic identification methods are discussed in Section 2.2 below.

Interpretation/compaction: Converting the material identified as relevant into a shorter form can be easy or hard. The easy way is simply to compact it, by dropping unnecessary words, replacing full forms by abbreviations, etc. The sophisticated way is to re-interpret the material in a more general, and new, form.

Compaction: The simplest methods are trivial. But this can be quite sophisticated as well. In a prize-winning paper, (Knight and Marcu, 2000) describe how they learned the most likely compressions by counting the grammar production rules used to generate thousands of trees (sentences collected from the Ziff-Davis corpus and parsed automatically). Consider these trees:



The left tree shows the sentence structure *t* of the string *abcde*. If you want to compress this string, you might consider generating the string *abe*. Is that in fact a 'good' compression of tree *t*? Let's say *abe* could have two syntactic derivations, *s1* and *s2*: which is more likely, *s1* (the middle tree) or *s2* (the right tree)? Knight and Marcu use Bayes' Rule as follows: P(*s1* / *t*) = P(*t* / *s1*) . P(*s1*), and then compute the probabilities P(*s1*) (how likely a sentence is *s1* in the language?) and P(*t* | *s1*) (how likely is it that the full sentence *t* can be produced from *s1*?) They compute the former by counting how many times they see trees like *s1* in the corpus, namely by counting how many times they see trees containing the production G → H A compared to G → F D (and the same for all the variations of A, as in A → C B D, and so on). They compute the latter by counting how many times they see trees in which a constituent has been added by a production, for example the addition of B under A: A → C B D versus A → C D. Having learned these statistics for all production rules (even lexical, leaf productions such as K → b and C → b), they can then compute the overall probability of a tree, and of each subportion of that tree. So they can determine how likely tree *t* is, and how likely *s1*, and *s2*.

To compress a sentence, then, they simply take its tree and see how the probability changes when they discard every portion of it. For example, the following sentence is stripped of its words in the following order (numbers are inversely correlated with probability, the smaller the number, the more likely the tree):

beyond that basic level, the operations of the three products vary widely. (1514588)
beyond that level, the operations of the three products vary widely. (1430347)
beyond that basic level, the operations of the three products vary. (1333437)
beyond that level, the operations of the three products vary. (1249223)
beyond that basic level, the operations of the products vary. (1181377)
the operations of the three products vary widely. (939912)
the operations of the products vary widely. (872066)
the operations of the products vary. (748761)
the operations of products vary. (690915)
operations of products vary. (809158)
the operations vary. (522402)

operations vary. (662642)

Notice that since the rules have been trained on grammatical sentences, they tend not to produce ungrammatical compressions. Notice also that longer sentences sometimes have better probabilities.

Using somewhat more traditional methods, (Barzilay and McKeown, 1999; Jing and McKeown, 1999; McKeown et al., 1999) also compress sentences, and sometimes combine them in a variety of methods.

Interpretation: In order to produce an *abstract*-type summary, the system has to interpret, compress, and re-formulate the extracted output in ways that are new, using words not contained in the input document. For example:

> Joe was hungry. He found a suitable place, went in, sat down. The waiter brought him a menu. Joe ordered a hamburger, which he thoroughly enjoyed. After paying, he left, satisfied.

What extract could one make of this? But an abstract is immediate:

> Joe was hungry, so he enjoyed a hamburger at a restaurant.

For some genres, extracts seem forced while abstracts are natural. A good example is fables (Aesop's fables, for example) and children's stories.

Making an abstract requires a lot of a system: it needs world knowledge against which to interpret the extracted portions (it needs to know *sit down* + *waiter* + *menu* + *order* + *eat* + *pay* ⇒ *restaurant-visit*). Representing such knowledge and performing the abstraction operations is not trivial; see (Hahn and Reimer, 1999). Collecting enough such knowledge for general usage amounts to building a domain model of the world. For that reason, we are unlikely to see general-purpose abstracting summarizers for a while.

Generation: Generating an extract is obviously trivial. But one can perform additional operations on extracted material to compress it further, in order to increase the density of information. This work resembles sentence planning: it generally employs a parse tree of the sentence(s) to be compressed. Some criteria identify which portions of the parse tree to squeeze out, and some rules specify how the remaining tree is put together again.

There is an overlap between compaction and generation. Some nice work was done by Barzilay and McKeown (2005) and Barzilay and Lapata (2005).

2.2 Methods of Topic Identification

Finding centrality: Top-down and bottom-up methods.

Top-down: predefine schemas of what's important. This is Information Extraction (IE, as in the MUC conference series). Example template/schema:

Earthquake:
- Where? match criterion: <location>
- What time? <time of day>
- How large? <number> + "Richter scale"
- How many casualties? <number> + "casualties"|"dead"|"wounded"|"injured"|…
- How much damage? <dollar amount> + "damage"
- Rescue operations? Text near to "relief"

As shown in the MUC evaluations, IE systems perform at about 60%. But once you have a template correctly filled in, you can easily produce an abstract, even with template-level realization. What's more, you can produce one in many languages! So this method is very nice, when it works.

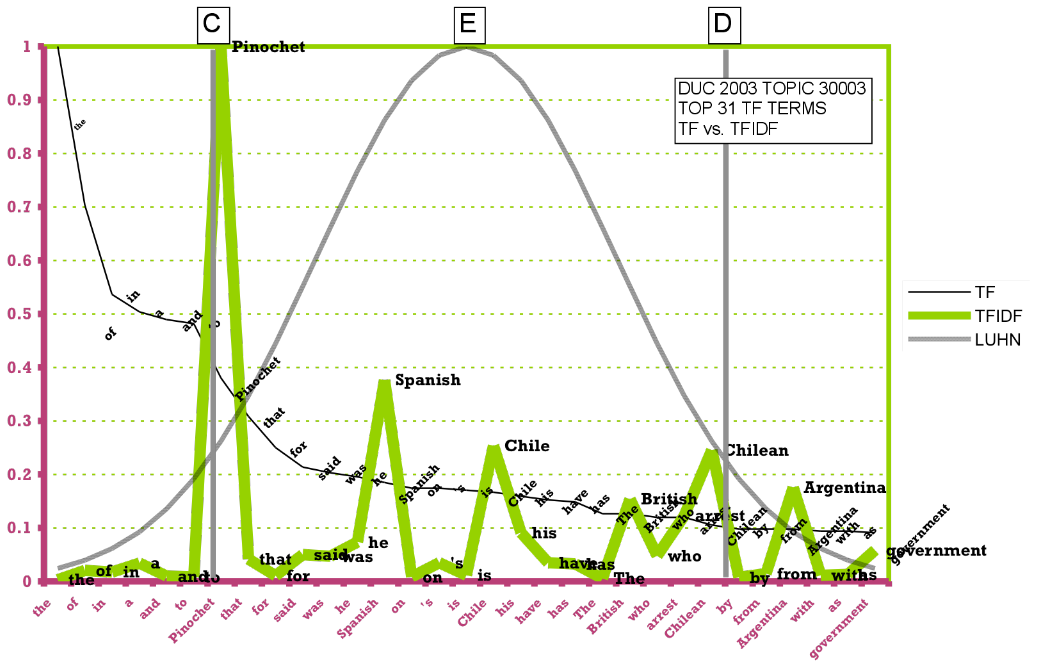
The problem is that template systems find only what they are told to look for; they cannot recognize anything surprising, and cannot notice when the text is not really about their own topic. In addition, templates are very knowledge-intensive to build and hard to maintain when new forms of events (say, new types of terrorism) are invented.

The earliest template systems were built as IE engines, not summarizers, in AI Projects, for example DeJong's FRUMP (DeJong, 1978).

Bottom-up: The other alternative is to follow the text, trying to estimate centrality using various clues. Generally these clues are at the word level, meaning that the systems are fairly unsophisticated. Almost all these methods work by assigning a score to each word and/or then each sentence according to various measures, integrating the scores for each sentence, and then ranking the sentences. The topmost *N* sentences are returned as extract summary. The most popular techniques are listed below.

A. Surface methods:

- **Word frequencies**. More important things tend to be mentioned more often: reward sentences that contain unusually frequent words (Luhn, 1959; Edmunson, 1969; Kupiec et al., 1995). Problem of synonyms and pertainyms. Concept signatures or phrase expansion sets. Experiments and results. (Strzalkowski et al., 1999) Problem: scaling up. Comparing *tf* and *tf.idf*:



Comparison of freq counts from DUC 2003: using *tf*, *tf.idf*, and mid-range words as per (Luhn, 1959).

- **Position**. In genres with fixed structure, exploit structure. For example, reward the title, abstract, (par 1, sent 1), etc. The Optimal Position Policy (OPP) is a method for learning the most rewarding sentences given a training corpus (Luhn, 1959; Lin and Hovy, 1997). Three steps:
  - Step 1: For each article, enumerate sentence positions (both → and ←).
  - Step 2: For each sentence, determine *yield* (= overlap between sentences and the index terms for the article).
  - Step 3: Create partial ordering over the locations where sentences containing important words occur: Optimal Position Policy (OPP).

  Examples: OPP for ZIFF corpus: $(T) > (P_2,S_1) > (P_3,S_1) > (P_2,S_2) > \{(P_4,S_1),(P_5,S_1),(P_3,S_2)\} > \ldots$
  (T=title; P=paragraph; S=sentence)
  OPP for *Wall Street Journal*: $(T) > (P_1,S_1) > \ldots$

- **Title words**. Content words in titles (and boldface, etc.) are important, and add scores to sentences that contain them (Luhn, 1959).

- **Cue phrase**. Some words and phrases indicate centrality—"it is important to note that...", "significantly", "in conclusion…", superlatives, etc. (Edmundson, 1969). Others are 'stigma phrases' that indicate their sentences should be scored *lower*—for example, quotations, sentences containing "for example", "hardly", "impossible", etc. — see (Paice, 1980; Teufel and Moens, 1999; Kupiec et al., 1995). Problem of finding them automatically.

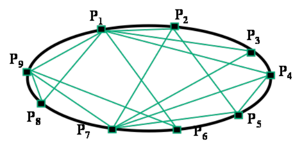
B. Mid-level methods (focus on cohesion across sentences):

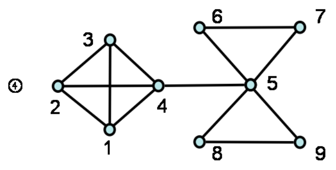
- **Lexical chains**. Link related words (identity, hypernym/hyponym, synonym/antonym, etc.), build chains, find the most highly connected sentences or paragraphs. See (Morris and Hearst, 1991).

- **Grammatical chains**: Link related words (coreference using pronouns etc.; elision/nominal substitution; conjunctions; etc.); also count connectedness of each sentence; score accordingly.

- **Semantic chains**. Link sentences based on the semantic relatedness of their words. Can use paraphrase, etc. — see below under Information Extraction.

For all three these methods, you convert the text into a graph, where each unit (say, sentence) is a node. You can look at simple connectedness or seek to find specific patterns.





- **Maximal Marginal Relevance** (MMR): Pick the central sentence, then compute the marginal relevance of all others using the MMR formula below. Select the one maximally relevant and add it to the summary. Recompute the marginal relevance of the remainder, and again add the best one to the summary. Stop when desired length is reached, and reorder summary sentences (Goldstein and Carbonell 99).

$$MMR(Q,R,S) = argmax_{D_i \in R\backslash S} [\, \lambda sim(D_i,Q) - (1-\lambda)\, max_{D_j \in R}\, sim(D_i,D_j) \,]$$

  where Q is the user's query, R is the whole set of sentences, S is the sentences already selected for the summary. The parameter λ specifies how similar the selected sentences must be to the query Q (when λ=1, they maximally resemble Q; when λ=0, they are maximally different from each other and pay no attention to Q).

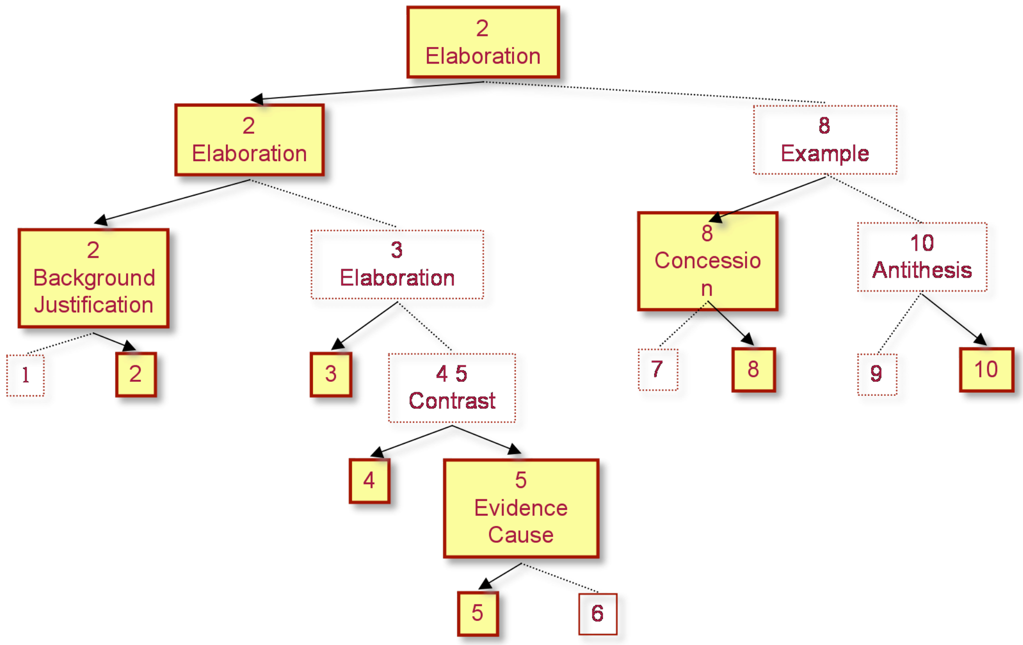
- **Partial parsing**. Perform a shallow parse of each sentence, to indicate at least some head information, and then use in conjunction with the rest.

C. Deep methods:

- **Discourse structure**. Discourse component nuclearity promotion using RST (Marcu, 1998). This is an important method that has had some influence in later work, despite the problems with discourse-level parsing. Claim: The multi-sentence coherence structure of a text can be constructed, and the 'centrality', typically the *nuclei* of the textual units in this structure, reflects their importance of each unit. To construct the discourse tree, use relations from *Rhetorical Structure Theory* (RST; Mann and Thompson, 1988). Each (or most) RST relations have two branches, the dominant one called the *nucleus* and the other one the *satellite*. When you have obtained the tree representing the text, then chop away all the satellite branches, and then the nucleus branches most distant from the root (the most nuclear sentence/clause).

[***With*** its distant orbit — 50 percent farther from the sun than Earth — and slim atmospheric blanket,[1]] [Mars experiences frigid weather conditions.[2]] [Surface temperatures typically average about –60 degrees Celsius (–76 degrees Fahrenheit) at the equator and can dip to –123 degrees C near the poles.[3]] [Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion,[4]] [***but*** any liquid water formed that way would evaporate almost instantly[5]] [***because*** of the low atmospheric pressure.[6]]

[***Although*** the atmosphere holds a small amount of water, and water-ice clouds sometimes develop,[7]] [most Martian weather involves blowing dust or carbon dioxide.[8]] [Each winter, ***for example***, a blizzard of frozen carbon dioxide rages over one pole, and a few meters of this dry-ice snow accumulate as previously frozen carbon dioxide evaporates from the opposite polar cap.[9]] [***Yet*** even on the summer pole , where the sun remains in the sky all day long, temperatures never warm enough to melt frozen water.[10]]



Summarization =
selection of the most important units +
determination of a partial ordering on them

2 > 8 > 3, 10 > 1, 4, 5, 7, 9 > 6

- **KR-based reasoning**. Do inference on parse results (Hahn and Reimer, 1999).
- **Information Extraction**: Use the frame/adage/script/skeletal structure underlying each 'unitary' message. Back to FRUMP, but extended.

2.3 Integrating Extracted Information

Having assigned scores to the fragments (sentences/nuggets) to be included in the summary, the problem is to integrate the various scores. Simple linear combination functions are easy to produce and test.

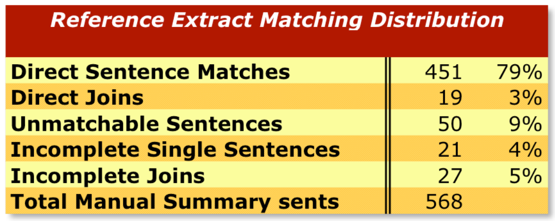
A more complex method is ranking using Naïve Bayes socring (Kupiec et al., 1995). Summarization is seen as a sentence ranking process based on the probability of a sentence would be included in a summary:

$$\begin{aligned} P(s \in S \mid F_1, F_2, \cdots, F_k) &= \frac{P(F_1, F_2, \cdots, F_k \mid s \in S) P(s \in S)}{P(F_1, F_2, \cdots, F_k)} \\ &= \frac{P(s \in S) \prod_{j=1}^{k} P(F_j \mid s \in S)}{\prod_{j=1}^{k} P(F_j)} \end{aligned}$$

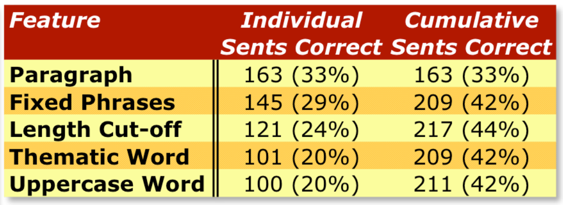
assuming statistical independence of the features, and $P(s \in S)$ is a constant (assuming uniform distribution for all s), where $P(F_j \mid s \in S)$ and $P(F_j)$ can be estimated from the training set. They used the following features:

- Sentence length cut-off (binary long sentence feature: only include sentences longer than 5 words)
- Fixed-phrase (binary cue phrase feature: such as "this letter …", total 26 fixed phrases)
- Paragraph (binary position feature: the first 10 and the last 5 paragraphs in a document)
- Thematic Word (binary key feature: sentences contain the most frequent content words or not)
- Uppercase Word (binary orthographic feature: capitalize frequent words not occur at the sentence-initial position)

Corpus: 188 OCRed documents and their manual summaries sampled from 21 scientific/technical domain.
Results:

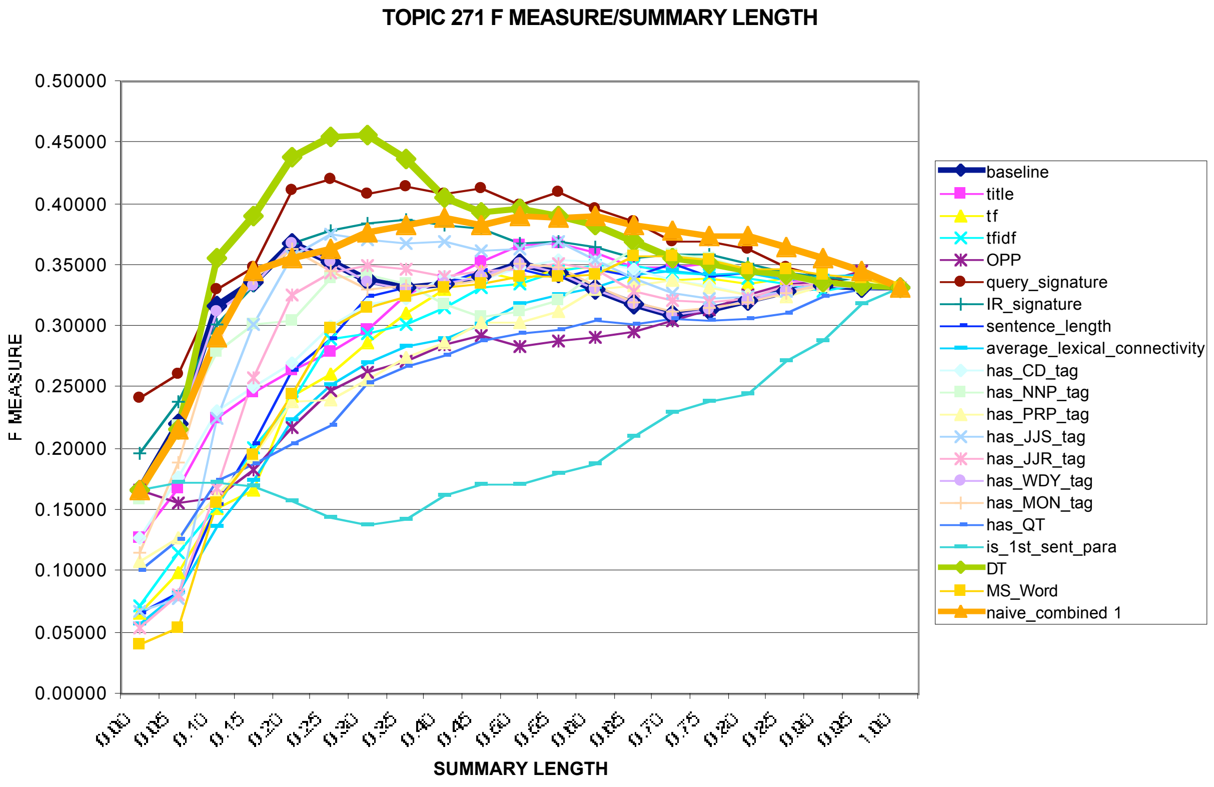


Reference Extract Matching Distribution		
Direct Sentence Matches	451	79%
Direct Joins	19	3%
Unmatchable Sentences	50	9%
Incomplete Single Sentences	21	4%
Incomplete Joins	27	5%
Total Manual Summary sents	568	



Feature	*Individual Sents Correct*	*Cumulative Sents Correct*
Paragraph	163 (33%)	163 (33%)
Fixed Phrases	145 (29%)	209 (42%)
Length Cut-off	121 (24%)	217 (44%)
Thematic Word	101 (20%)	209 (42%)
Uppercase Word	100 (20%)	211 (42%)

In later work, Hovy and Lin compared several methods (just OPP, just *tf.idf*, linear combination, C4.5 decision trees, etc.); see the graph below in (Hovy and Lin, 1999). Note the abysmal performance of the Microsoft Word summarizer.



Comparison of different identification and combination techniques.

3. Multi-Document Summarization

This is a newer area. The first DUC evaluation/competitions for multi-document summaries attracted about 10 systems. Top-scoring systems were built at ISI (Hovy and Lin, 2001; 2002) and Southern Methodist University (Harabagiu et al., 2001). Both of these simply extract and order important sentences. In contrast, the system built by Columbia University (McKeown et al. 2001; 2002)

The main problem is to avoid redundancy: when multiple documents/sentences about the same topic, from different sources and authors, have been identified as relevant, there is bound to be some overlap. But even if all redundancy has been identified and taken care of, one still has the problem of properly organizing the material to form a coherent (and correct!) story.

The NeATS algorithm for multi-doc summaries:

1. Extract and rank passages

Given the input documents, form a query, extract sentences, and rank them, using modules of the Webclopedia QA system

- identify key concepts for each topic group. Compute unigram, bigram, and trigram topic signatures (Lin and Hovy, 2000; Hovy and Lin, 1999) for each group, using the likelihood ratio λ (Dunning, 1993)
- remove from the signatures all words or phrases that occur in fewer than half the texts of the topic group

- save the signatures in a tree, organized by signature overlap, using the parse tree format of CONTEX (Hermjakob, 1997; 2000)
- use the Webclopedia query formation module to form queries, most specific first
- use Webclopedia's version of MG to perform sentence-level IR and return a ranked list of sentences

2. Filter for content

Using an OPP policy as developed for the SUMMARIST single-document summarizer (Lin and Hovy, 1997), remove extracted sentences too far from the high-importance regions:
2.a from the ranked list, remove all sentences with sentence position > 10 (simple OPP policy)
2.b also decrease ranking score of all sentence containing stigma words (day names; time expressions; sentences starting with conjunctions such as "but", "although"; sentences containing quotation marks; sentences containing the verb "say").

3. Enforce cohesion and coherence

Each remaining sentence is paired with a suitable introductory sentence:
3.a pair each sentence with the first sentence (lead) of its document; but if the first sentence contains fewer than 5 words, then take the next one. For example (where *x.y* stands for *document number . sentence number*):

4.3, 6.6, 2.5, 5.2… → 4.1, 4.3, 6.1, 6.6, 2.1, 2.5, 5.1, 5.2...

4. Filter for length

Select the required number of sentence pairs using a simplified version of CMU's MMR algorithm:
4.a include first pair
4.b using a simplified version of MMR (Goldstein et al., 1999), find the sentence pair most different from the included ones, and include it too. (In the DUC-2001 implementation, NEATS did not consider the sentence pair, just the sentence. This caused some degradation.)
4.c repeat 4.b until the summary length criterion is satisfied

→ 4.1, 4.3, 2.1, 2.5

5. Ensure chronological coherence

Reorder the pairs in publication order, and disambiguate all time words with explicit dates:
5.a reorder pairs in publication order

→ 2.1, 2.5, 4.1, 4.5

5.b for each time word ("today", "Monday", etc.) compute the actual date (from the dateline) and include it in the text in parentheses, in order to signal which day each "today" (etc.) is.

6. Format and printing results
Format and output the final result.

4. Other Summarization Challenges

Update summaries. In recent years, the DUC and TAC conferences have branched out to include new challenges. One of them is to produce 'update summaries': systems that follow a story as it unfolds over time to produce summaries that contain only the latest information. In this, the early thesis work of Radev is very valuable, and has been extended in the recent DUC and TAC evaluation conferences (see DUC 2007, TAC 2008).

As defined in DUC/TAC, the update task asks the system to create (update) summaries assuming that the user has read the previous documents (not the previous summaries of those documents); see http://www.nist.gov/tac/tracks/2008/summarization/update.summ.08.guidelines.html

Obviously, this leads to different summaries, since you (presumably) try to reduce redundancy between the update summary and the old documents (and not the old summaries). It is of course also possible to define the task the other way, namely that the user has been reading summaries alone. So there are two possible definitions of the update task:

i) Summary B avoids repetitions with the content of the articles in Document Cluster A.

ii) Summary B avoids repetitions with the content of Summary A.

The definition that has been used is i), and ii) is a common misunderstanding. The advantage of i) is that it can be evaluated in an even way for all teams because we must avoid repetitions with the same piece of text (cluster A is the same for all systems, but Summary A is different for different systems).

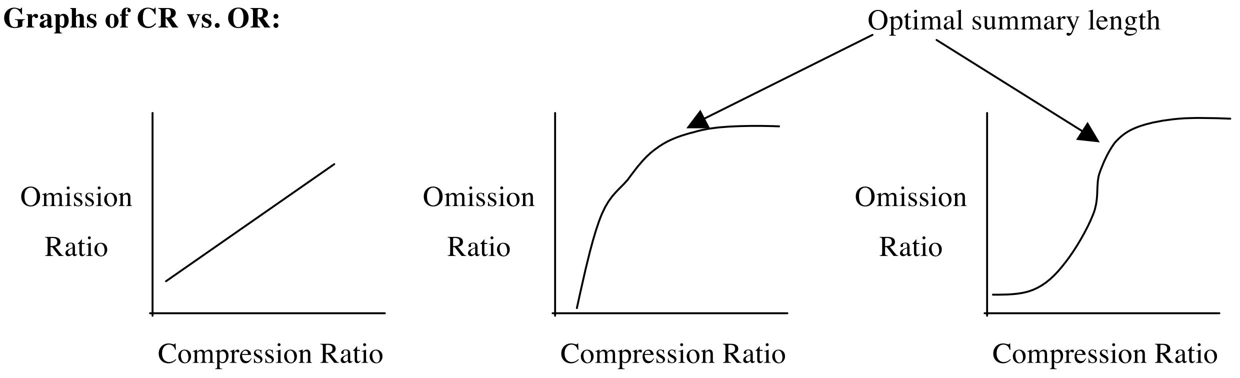
5. Summary Evaluation

What do you really want to know? Essentially, you want to keep all the important information and throw away the rest. So you have to measure the information retained and the length of the summary. Can define two measures (Hovy and Lin, 1999):

Compression ratio = (length of Summary) / (length of Original)

Omission Ratio = (information in Summary) / (information in Original)

Graphs of CR vs. OR:



Compression is easy: you count words or characters. But *how do you measure information*? How do you factor out the reader's knowledge? Here are some ideas:

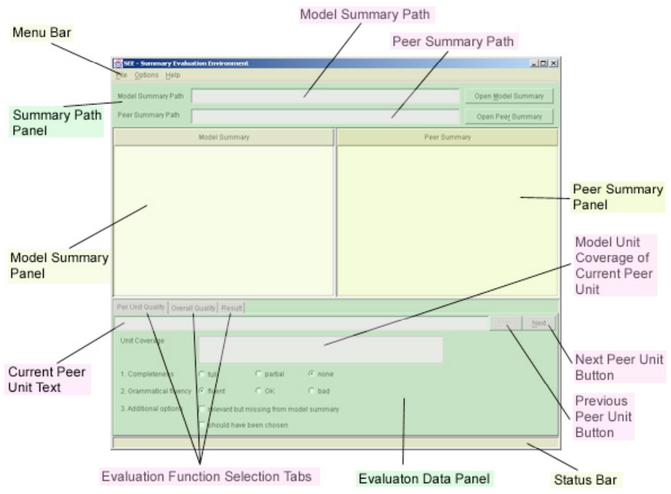
- **Information Theory measures: the Shannon Game**. Three guessers: some try to guess the text, having seen nothing; others read the summary, and then guess it; others read the text, put it away, and guess it. The ratios provide an indication of the informativeness of the summary.
- **Question/Answer method: the Question Game.** The Questioners make up questions about the text. The answerers first try to answer the questions, having seen nothing (baseline knowledge); they then read the summary, and then answer the same questions; finally, they read the text, put it away, and answer the questions again. The ratios provide an indication of the informativeness of the summary.

- **IR method 1: Categorization task.** Given a set of summaries or texts, do readers categorize them in the same bins?
- **IR method 2: Ad Hoc task.** Given a set of summaries created for a certain query, do readers judge them as relevant?
- **Task-based methods**. Use what Sparck Jones and Galliers (1996) call *extrinsic measures*—measure how well someone can perform a task with and without the help of the technology. For example, do the Ad Hoc task above, but measure the length of time taken by the user to handle *N* documents vs. *N* summaries.
- **BLEU revised, and ROUGE**: Can one use a modified form of the automated MT evaluation metric BLEU (Papineni et al. 01) for summarization evaluation? Experiments by Lin and Hovy (03) indicate one can. In the ROUGE package, Lin (04) has implemented about 10 different scoring methods, and has shown which ones give best correlation with human scores under which conditions. Generally, simple unigrams (BLEU1) are best to reflect content, but do not capture any grammaticality: a pure keyword extraction system is as good as one that produces beautiful prose with the same contents. Longest Common Substring (LCS) that rewards not fixed-length ngrams, but a combination of the maximal substrings of words found anywhere, works well in general, for both content and grammaticality. Simple word error rate (WER), not even using demorphing, works well too.
- **Pyramid method and nugget popularity**: Teufel and Van Halteren introduced a method of scoring short units for relevance: the more gold standard (reference) summaries that contain a unit, the higher its score. This method was called the Pyramid Scoring method by Nenkova and Passonneau (2005, 2006) and tried in the DUC 2005 and 2006 evaluations. Deciding which units 'mean the same' and can be put into the same equivalence class for scoring is a time-consuming and problematic step; the longer the unit, the more variation among gold-standard creators and output score assigners.
- **Basic Elements and Nuggets**: Instead of fixed-length word ngrams, use syntactically coherent elements (multi-word units) as the comparison elements. Also extend units with paraphrases. Gives slightly better performance than ROUGE. (See Hovy et al. 2005 and Zhou and Hovy 2006).

See online literature about the SUMMAC-96 evaluation and the DUC evaluation series, organized annually by NIST. The DUC organizers every year announce a task (headline summarization; just-the-news update summarization; topic-oriented question-answering summarization; etc.) and create the appropriate corpora as well as gold-standard reference summaries (by hand) against which system summaries are scored. At the annual meeting (the DUC conferences) the results are made available and techniques discussed.

The SEE manual evaluation interface (Lin 03) was used by NIST in its 2003 and subsequent evaluation. Humans compared a system's summary to humans' ideals. The compared sentences clause by clause: does the system include this specific clause? Also, overall fluency and grammaticality scores are given. See the figure below.

Subsequently, DUC used Lin's ROUGE package as one of the scoring mechanisms. Fully automated, several of the ROUGE metrics correlate well with human scoring.



The SEE manual evaluation interface (Lin 03).

It turns out that people are not very consistent: in single-document evaluation, from 5,921 judgments, 18% (1,076 judgments) differ for the same ideal–system evaluation pair (from the same assessor), and of these 2.4% (143 judgments) have *three* different coverage scores. And in the multi-document case, from a total of 6,963 judgments, 7.6% (528) contain different judgments for the same pair (from the same assessor, and 27 of them have three different coverage scores; see (Lin and Hovy, 2002 DUC workshop).

Automated evaluation methods

As MT flourished with BLEU, there was the dream of creating an automated method to evaluate summaries. Chin-Yew Lin and Hovy from ISI created the ROUGE system, which (like BLEU) compared a system output to a set of manually created gild-standard summaries, and scored the overlap using various ngram overlap measures. In later years, DUC used ROUGE as one of the scoring mechanisms, and it is one of the standards reported today in summarization research.

Fully automated, several of the ROUGE metrics correlate well with human scoring.

In brief, the automated scoring research to date is:

- **BLEU revised, and ROUGE**: Can one use a modified form of the automated MT evaluation metric BLEU (Papineni et al. 01) for summarization evaluation? Experiments by Lin and Hovy (03) indicate one can. In the ROUGE package, Lin (04) has implemented about 10 different scoring methods, and has shown which ones give best correlation with human scores under which conditions. Generally, simple unigrams (BLEU1) are best to reflect content, but do not capture any grammaticality: a pure keyword extraction system is as good as one that produces beautiful prose with the same contents. Longest Common Substring (LCS) that rewards not fixed-length ngrams, but a combination of the maximal substrings of words found anywhere, works well in general, for both content and grammaticality. Simple word error rate (WER), not even using demorphing, works well too.
- **Pyramid method and nugget popularity**: Teufel and Van Halteren introduced a method of scoring short units for relevance: the more gold standard (reference) summaries that contain a unit, the higher its score. This method was called the Pyramid Scoring method by Nenkova and Passonneau (2005, 2006) and tried in the DUC 2005 and 2006 evaluations. Deciding which units 'mean the same' and can be put into the same equivalence class for scoring is a time-consuming and problematic step; the longer the unit, the more variation among gold-standard creators and output score assigners.

- **Basic Elements and Nuggets**: Instead of fixed-length word ngrams, use syntactically coherent elements (multi-word units) as the comparison elements. Also extend units with paraphrases. Gives slightly better performance than ROUGE. (See Hovy et al. 2005 and Zhou and Hovy 2006).

ROUGE

ROUGE is a recall-based equivalent of BLEU (which is precision-based: it measures whether translated fragments are correct, whereas ROUGE measures whether fragments are included):

$$C_n = \frac{\sum_{C \in \{Model\ Units\}} \sum_{n\text{-}gram \in C} Count_{match}(n\text{-}gram)}{\sum_{C \in \{Model\ Units\}} \sum_{n\text{-}gram \in C} Count(n\text{-}gram)} \qquad Ngram(i,j) = \exp\left(\sum_{n=i}^{j} w_n \log C_n\right)$$

Count$_{match}$ *(n-gram)* is the max number of ngrams that co-occur in the system's and the ideal summary, and *Count (n-gram)* is the number of ngrams in the ideal.

ROUGE has had a big influence on summarization research. Lin implemented various alternatives:

- ROUGE-N: N-gram based co-occurrence statistics
- ROUGE-L: statistics on longest common substring (LCS)
- ROUGE-W: Weighted LCS-based statistics that favors consecutive LCSes
- ROUGE-S: Skip-bigram-based co-occurrence statistics
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics

Generally, the longer the sequence of overlap scored, the more grammatical and more correct — but, of course, the less chance of a match:

police killed the gunman

1. <u>police</u> kill the gunman
2. <u>the gunman</u> kill <u>police</u>
3. <u>the gunman</u> <u>police</u> <u>killed</u>

ROUGE-N: S1=S2 ("police", "the gunman" both match); S3>S1

ROUGE-L (longest common substring):

- S1=3/4 ("police the gunman")
- S2=2/4 ("the gunman")
- S1>S2; S2=S3

ROUGE-S:

- S1=3/6 ("police the", "police gunman", "the gunman")
- S2=1/6 ("the gunman")
- S3=2/6 ("the gunman", "police killed")
- S1>S3>S2

The variations of ROUGE were subjected to extensive evaluation to determine correlations:

Corpora

- DUC 01, 02, and 03 evaluation data
- Including human and systems summaries

Seven task formats

- Single doc 10 and 100 words, multi-doc 10, 50, 100, 200, and 400 words

Three versions

- CASE: the original summaries

- STEM: the stemmed version of summaries
- STOP: STEM plus removal of stopwords

Number of references
- Single and different numbers of multiple references

Quality criterion
- Pearson's product moment correlation coefficients between systems' average ROUGE scores and their human assigned mean coverage score

Metrics
- 17 ROUGE metrics: ROUGE-N with N = 1 to 9, ROUGE-L, ROUGE-W, ROUGE-S and ROUGE-SU (with maximum skip-distance of 0, 4, and 9)

Statistical significance
- 95% confidence interval estimated using bootstrap resampling

Results: Overall
- Using multiple references achieved better correlation with human judgment than just using a single reference.
- Using more samples achieved better correlation with human judgment (DUC 02 vs. other DUC data).
- Stemming and removing stopwords improved correlation with human judgment.
- Single-doc task had better correlation than multi-doc

Specific
- ROUGE-S4, S9, and ROUGE-W1.2 were the best in 100 words single-doc task, but were statistically indistinguishable from most other ROUGE metrics.
- ROUGE-1, ROUGE-L, ROUGE-SU4, ROUGE-SU9, and ROUGE-W1.2 worked very well in 10 words headline like task (Pearson's ρ ~ 97%).
- ROUGE-1, 2, and ROUGE-SU* were the best in 100 words multi-doc task but were statistically equivalent to other ROUGE-S and SU metrics.

Basic Elements

Strategy: decompose texts into minimal fragments, then create nuggets of whatever size.

BE definition: A *nugget* is predicated on either an *event* or an *entity*. Two parts for each nugget:

- *Content*: the head noun of the entity, or the head verb of the event and the head noun of its associated entity (if more than one entity is attached to the verb, then its subject)
- *Context*: a coherent piece of info associated with the content that helps define it uniquely

Nugget extraction is performed recursive to satisfy granularity needs. Each nugget is composed out of Basic Elements.

BEs: all *nouns*, all *verbs*, all modifier-noun pairs (e.g., *adjective–noun* or *noun–noun*); all modifier-verb pairs (e.g., *adverb–verb* or *headnoun–verb*). Example: "The girl working at the bookstore in Hollywood talked to the diplomat living in Britain."
Nuggets:

1) [girl] working at the bookstore in Hollywood
    [girl] working at the bookstore
    [bookstore] in Hollywood

2) {girl} [talked] to the diplomat living in Britain
{girl} [talked] to the diplomat
[diplomat] living in Britain

Extracting BEs automatically is easy:

- Parse sentence
- Apply regular expression matcher (tregex, etc.) to identify subtrees with appropriate structure

Evaluating summaries/answers with BEs, using humans for matching:

For *reference (gold standard) summaries* (per docset):

- System creates eNugs for all sentences in all docs in set
- Annotators group equivalent eNugs into groups
- Popularity scores automatically assigned to nugget groups
- Multiple appearances in one document are given 1 point
- Resulting nugget groups with scores form the set's gold standard

For *peer (system) summaries*:

- System creates eNugs for all sentences
- 2 annotators match/align peer's eNugs with reference nugget groups
- Peer summary score computed automatically

Of course, the matching step can be automated, but some accuracy is lost.

Optional further readings

<u>Overviews:</u>

NLP Handbook chapter: Hovy, E.H. 2005. Automated Text Summarization. In R. Mitkov (ed), *The Oxford Handbook of Computational Linguistics*, pp. 583–598. Oxford: Oxford University Press.

Encyclopedia article: Hovy, E.H. 2001. Text Summarization. In R. Mitkov (ed.), *Handbook of Natural Language Processing*. Oxford: Oxford University Press.

Book: Mani, I. 2001. *Automatic Summarization*. Amsterdam/Philadelphia: John Benjamins Press.

Book: Mani, I. and M. Maybury (eds). 1999. *Advances in Automatic Text Summarization*. Cambridge, MA: MIT Press.

DUC Workshops: Harman, D. and P. Over. 2002–07. *DUC Proceedings* 2002, 2003, 2004, 2005, 2006, 2007.

TAC conferences: Trang Dang, H. *TAC Proceedings* 2008.

<u>Interpretation:</u>

DeJong, G.J. 1978. *Fast Skimming of News Stories: The FRUMP System*. Ph.D. Dissertation, Yale University.

Hahn, U. and U. Reimer. 1999. Knowledge-Based Text Summarization: Salience and Generalization Operators for Knowledge Base Abstraction. In I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. Cambridge, MA: MIT Press, 215–232.

Extraction / Topic Identification Methods:

Edmundson, H.P. 1969. New Methods in Automatic Extraction. *Journal of the ACM* 16(2), 264–285. Also in I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. 1999. Cambridge, MA: MIT Press, 23–42.

Hovy, E.H., C.-Y. Lin, and L. Zhou. 2005. A BE-based Multi-document Summarizer with Sentence Compression. *Proceedings of the Multilingual Summarization Evaluation Workshop at the ACL 2005 conference*. Ann Arbor, MI.

Kupiec, J., J. Pedersen, and F. Chen. 1995. A Trainable Document Summarizer. *Proceedings of the Eighteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR),* Seattle, WA, 68–73. Also in I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. 1999. Cambridge, MA: MIT Press, 55–60.

Lin, C-Y. and E.H. Hovy. 1997. Identifying Topics by Position. *Proceedings of the Applied Natural Language Processing Conference (ANLP-97),* Washington, DC, 283–290.

Luhn, H.P. 1959. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*: 159–165. Also in I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. 1999. Cambridge, MA: MIT Press, 15–22.

Marcu, D. 1998. Improving Summarization through Rhetorical Parsing Tuning. *Proceedings of the COLING-ACL Workshop on Very Large Corpora*, Montreal, Canada, 10–16.

Strzalkowski, T., G. Stein, J. Wang, and B. Wise. 1999. A Robust Practical Text Summarizer. In I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. Cambridge, MA: MIT Press, 137–154.

Zhou, L., C.-Y. Lin, and E.H. Hovy. 2006. Summarizing Answers for Complicated Questions. Full paper. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC).* Genoa, Italy.

Sentence Compression / Generation

Barzilay, R. and K.R. McKeown. 2005. Sentence Fusion for Multidocument News Summarization. *Computational Linguistics*.

Barzilay, R. and M. Lapata. 2005. Collective Content Selection for Concept-To-Text Generation. *Proceedings of EMNLP conference*.

Jing, H. and K.R. McKeown, 1999. The Decomposition of Human-Written Summary Sentences. *Proceedings of the SIGIR-99 Conference*.

McKeown, K.R., J. Klavans, V. Hatzivassiloglou, R. Barzilay, E. Eskin. 1999. Towards Multi-Document Summarization by Reformulation. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Knight, K. and D. Marcu. 2000. Statistics-Based Summarization—Step One: Sentence Compression. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. Outstanding Paper award.

Multi-Document Summarization:

Harabagiu., S. et al. 2001. Multi-Document Summarization. *Proceedings of the DUC-0 Workshop on Multi-Document Summarization at the SIGIR Conference*. New Orleans, LA.

Lin, C.-Y. and E.H. Hovy. 2001. NeATS: A Multi-Document Summarizer. *Proceedings of the DUC-0 Workshop on Multi-Document Summarization at the SIGIR Conference*. New Orleans, LA.

Radev, D. and K.R. McKeown. 1998. Generating Natural Language Summaries from Multiple On-Line Sources. *Computational Linguistics* (special issue on language generation), 24(3), 469–500.

Zhou, L. and E.H. Hovy. 2005. Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats. *Proceedings of the conference of the Association for Computational Linguistics (ACL)*. Ann Arbor, MI.

Evaluation

Spärck Jones, K. and J.R.Galliers. 1996. *Evaluating Natural Language Processing Systems: An Analysis and Review*. New York: Springer.

Hovy, E.H. and C-Y. Lin. 1999. Automating Text Summarization in SUMMARIST. In I. Mani and M. Maybury (eds), *Advances in Automated Text Summarization*. Cambridge, MA: MIT Press, 81–97.

Papineni, K., S. Roukos, et al. 1991. BLEU. *Proceedings of the HLT Conference*. San Diego, CA. Also available as IBM Technical Report.

Lin, C.-Y. 2004. The ROUGE Summary Evaluation Package and Experiments.

Hovy, E.H., C.-Y. Lin, L. Zhou, and J. Fukumoto. 2006. Automated Summarization Evaluation with Basic Elements. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*. Genoa, Italy.

Tratz, S. and E.H. Hovy. 2008. Summarization Evaluation Using Transformed Basic Elements. *Proceedings of Text Analytics Conference (TAC-08)*. NIST, Gaithersburg, MD.

90 Soldiers Arrested After Coup Attempt In Tribal Homeland

MMABATHO, South Africa (AP)

About 90 soldiers have been arrested and face possible death sentences stemming from a coup attempt in Bophuthatswana, leaders of the tribal homeland said Friday.

Rebel soldiers staged the takeover bid Wednesday, detaining homeland President Lucas Mangope and several top Cabinet officials for 15 hours before South African soldiers and police rushed to the homeland, rescuing the leaders and restoring them to power.

At least three soldiers and two civilians died in the uprising.

Bophuthatswana's Minister of Justice G. Godfrey Mothibe told a news conference that those arrested have been charged with high treason and if convicted could be sentenced to death. He said the accused were to appear in court Monday.

All those arrested in the coup attempt have been described as young troops, the most senior being a warrant officer.

During the coup rebel soldiers installed as head of state Rocky Malebane-Metsing, leader of the opposition Progressive Peoples Party.

Malebane-Metsing escaped capture and his whereabouts remained unknown, officials said. Several unsubstantiated reports said he fled to nearby Botswana.

Warrant Officer M.T.F. Phiri, described by Mangope as one of the coup leaders, was arrested Friday in Mmabatho, capital of the nominally independent homeland, officials said.

Bophuthatswana, which has a population of 1.7 million spread over seven separate land blocks, is one of 10 tribal homelands in South Africa. About half of South Africa's 26 million blacks live in the homelands, none of which are recognized internationally.

Hennie Riekert, the homeland's defense minister, said South African troops were to remain in Bophuthatswana but will not become a "permanent presence."

Bophuthatswana's Foreign Minister Solomon Rathebe defended South Africa's intervention. "The fact that ... the South African government (was invited) to assist in this drama is not anything new nor peculiar to Bophuthatswana," Rathebe said. "But why South Africa, one might ask? Because she is the only country with whom Bophuthatswana enjoys diplomatic relations and has formal agreements."

Mangope described the mutual defense treaty between the homeland and South Africa as ``similar to the NATO agreement," referring to the Atlantic military alliance. He did not elaborate. Asked about the causes of the coup, Mangope said, "We granted people freedom perhaps ... to the extent of planning a thing like this."

The uprising began around 2 a.m. Wednesday when rebel soldiers took Mangope and his top ministers from their homes to the national sports stadium. On Wednesday evening, South African soldiers and police stormed the stadium, rescuing Mangope and his Cabinet.

South African President P.W. Botha and three of his Cabinet ministers flew to Mmabatho late Wednesday and met with Mangope, the homeland's only president since it was declared independent in 1977. The South African government has said, without producing evidence, that the outlawed African National Congress may be linked to the coup.

The ANC, based in Lusaka, Zambia, dismissed the claims and said South Africa's actions showed that it maintains tight control over the homeland governments. The group seeks to topple the Pretoria government. The African National Congress and other anti-government organizations consider the homelands part of an apartheid system designed to fragment the black majority and deny them political rights in South Africa.

If You Give a Mouse a Cookie
Laura Joffe Numeroff © 1985

If you give a mouse a cookie, he's going to ask for a glass of milk.

When you give him the milk, he'll probably ask you for a straw.

When he's finished, he'll ask for a napkin.

Then he'll want to look in the mirror to make sure he doesn't have a milk mustache.

When he looks into the mirror, he might notice his hair needs a trim.

So he'll probably ask for a pair of nail scissors.

When he's finished giving himself a trim, he'll want a broom to sweep up.

He'll start sweeping.

He might get carried away and sweep every room in the house.

He may even end up washing the floors as well.

When he's done, he'll probably want to take a nap.

You'll have to fix up a little box for him with a blanket and a pillow.

He'll crawl in, make himself comfortable, and fluff the pillow a few times.

He'll probably ask you to read him a story.

When you read to him from one of your picture books, he'll ask to see the pictures.

When he looks at the pictures, he'll get so excited that he'll want to draw one of his own. He'll ask for paper and crayons.

He'll draw a picture. When the picture is finished, he'll want to sign his name, with a pen.

Then he'll want to hang his picture on your refrigerator. Which means he'll need Scotch tape.

He'll hang up his drawing and stand back to look at it. Looking at the refrigerator will remind him that he's thirsty.

So…he'll ask for a glass of milk.

And chances are that if he asks for a glass of milk, he's going to want a cookie to go with it.