Part – D

Multilingual Automatic Summarization

Introduction:

Automatic summarization has been a very active field within computational linguistics, and researchers have looked at this problem from various angles. In the past, the focus has been on texts written in a single source language. However, in recent years multilingual summarization has been generating a lot of interest where texts written in multiple languages are used by summarization systems.

We can distinguish between single- and multidocument summarization. A summary can be driven by a specific query or provide a general summary of the document (or document collection). The summary itself can have different purposes. An **informative** summary, for example, is a compressed version of the original covering the most important facts reported in the input text(s) (e.g., summary of a journal article). A summary can be **indicative** of topics covered in the input text without providing further details (e.g., keywords for scientific papers). Another type of summary can be found in the form of reviews. Such an **evaluative** summary gives an opinion on the input text most often by comparing it to similar documents. An **elaborative** summary can provide more details of parts of a large document or the document linked to by the current document to help navigation through large documents or linked collections such as Wikipedia

Most basically, we can distinguish between a summary as being an extract or an abstract, with rather different implications. An extract is a summary constructed mostly by choosing the most relevant pieces of text, perhaps with some minor edits. An abstract is a gloss that describes the contents of a document without necessarily featuring any of that content. Most current summarization systems produce extracts, but some attempts have been made to produce abstracts [2] or propose solutions for sentence compressions that would keep only the important part of a (longer) sentence [3].

Recent variations include generation summaries for bibliographic information, update summarization (i.e., only report the latest changes of a developing story), or guided summarization in which the goal is to extract semantic information from the source documents depending on the text type (e.g., accidents/natural disasters).

Multilingual summarization inherits all features (and challenges) from monolingual automatic summarization and adds an additional dimension to the overall task. Loosely defined, multilingual summarization involves more than one language in the process of automatically summarizing a text.

To be more specific, summarization can be carried out on one source language (e.g., Arabic), and the resulting summary is presented in one target language (e.g., English). We call this specific multilingual summarization task **translingual** summarization.

Even more complex is a task called **crosslingual** summarization. Here the summarization task is spread out over multiple source languages, and the resulting summary is presented in one (or more) target languages.

Crosslingual summarization is the more challenging task because it requires the integration of multiple source documents coming from different languages. All multilingual summarization tasks, whether they involve two or multiple languages as source or target languages, face a host of problems.

The first problem is crossdocument coreference resolution. Named entities are often transcribed differently in different languages. *Al-Qaida*, *al-Qa'ida*, *el-Qaida*, or *al Qaeda* are different transliterations for *Al-Qaeda* in English and *El Kaida* in German, for example. A summarization system needs to normalize these variants and map them to a unique entity.

Similarly, anaphora resolution in a multilingual setting needs to be addressed. Languages encode number and gender agreement differently. English lacks grammatical gender, but other Indo-European languages, for example, use grammatical gender to indicate reference to antecedents with the respective pronoun (e.g., French: *la lune* (FEM)-*elle*; German: *der Mond* (MASC)-*er*).
Other problems a multilingual summarization system may encounter could be due to different discourse structures commonly used in different languages. Discourse relations may be expressed differently in different languages. Hence, it may be difficult to generate a coherent summary in the target language.

Even more complex problems are created if language-dependent concepts need to be summarized. The summarization of legal concepts in different languages, for instance, may be difficult, if not impossible, to carry out.

Many of these problems already exist in monolingual summarization (e.g., anaphora resolution), but they get more severe because languages encode anaphora, discourse structure, or concepts differently. The quality of a summary therefore hinges on the quality of the machine translation systems, which are still far from perfect. A general strategy of minimizing errors induced by machine translation is to find ways to minimize the impact of the problems described previously. Summarization systems can, for example, include knowledge-poor approaches to anaphora resolution [4] that rely on easy-to-extract features and graph-based approaches that cluster similar sentences according to word-based similarity metrics [5].

History SUMMARIST, one of the first summarization systems, which also generated summaries in languages other than English, was developed by Ed Hovy and Chin-Yew [6] in 1998. The system was able to produce extracts from newspaper articles written in English, Spanish, French, German, and Indonesian.[1]

1. Some older versions of the system also generated summaries in Arabic and Japanese.

In 2001, SummBank—the first crosslingual summarization framework for research in this field—was developed. This resource was the product of a Johns Hopkins Research Workshop [7] and comprises 360 multidocument, human-written summaries for 40 news clusters in English and Chinese.[2]

Approaches to Summarization

The Classics

Automatic summarization is the extraction and modification of material from a source document to create a more succinct description of the original content to satisfy the user's information need. If text is extracted verbatim (with minimal modifications), the summary is called an **extract**; if the text is abstracted to capture the gist of the content on a more abstract level, the summary is called an **abstract**. Most current automatic summarization systems produce extracts, not abstracts.

A wide body of research has focused on how user needs can be addressed. This research led to the introduction of different variations of the summarization task, such a multidocument summarization and query-based summarization. Multidocument summarization provides summaries of multiple documents concerned with the same topic, and query-based summarization guides the summarization process by a user query instead of providing a general-purpose summary. Query-based summarization can be carried out for a single document or multiple documents.

In general, every summarization system can be divided into three stages:

Analysis The source text is analyzed, and some internal representation is generated. This representation can be a collection of feature vectors (e.g., counts of most frequent words in a sentence) or a logical representation of the described content. For a translingual system, this part is particularly crucial because the representation needs to be in some way compatible across different languages.

Transformation The internal representation is manipulated in such a way that the content of the source document is condensed (e.g., ranking sentences according to a scoring function). Again, such transformations may be language-dependent on the way the internal representation is chosen.

Realization The summary is generated by producing a shorter piece of text than the source document. A shallow approach could just output the *n* highest-scoring sentences according to a scoring function, but most likely other operations are necessary to produce a coherent summary (e.g., coreference resolution). A multilingual summarization system has to employ a machine translation component at this point, if it has not done so earlier in the progress, to ensure that the summary is readable in the target language. Alternatively, language generation can produce the summary directly from an abstract semantic representation.

Research on automatic summarization can be traced back into the late 1950s with Luhn's work on summarization [21]. Luhn investigated the influence of frequent terms in a sentence and came up with a scoring function that computes a score for each sentence in the document.

Other early systems on summarization relied on surface-based features for extracting important sentences. In general, sentences at the beginning (or the end) of the document were found to be often important [22]. Hence, the position of a sentence in a document is often a good feature to determine the importance of a sentence because an author would like to put such sentences at a salient position in the document.

Many of the early approaches as well as recent ones used these surface features that are easy to extract [21, 22, 23].

• indicative phrases such as *in summary*

• distribution of terms

• overlap with words from headings, titles

• position of sentence in the text, paragraph, and so on

These general features can also easily be adopted for multilingual summarization. Term distribution and position are mostly language-independent features where position is more genre-dependent than language-dependent. For example, news messages tend to have important sentences at the beginning, whereas legal text tends to show summarization information at the end.

The problem that is often observed with these surface-based approaches is that the generated summaries are not always coherent. As a result, discourse theories that predict the coherence of a text were incorporated in the summarizer. Because discourse is often seen as a graph structure, graph-based theories are discussed in the following subsection.

Graph-Based Approaches

This section discusses approaches that model text in the form of graphs and how this representation can help to improve automatic summarization of text. On the one hand, discourse theories such as rhetorical structure theory (RST) [24] model the coherence of a text via a tree structure, and on the other hand, graph-based ranking approaches such as PageRank [25] have been shown to be helpful for scoring sentences according to their importance.

Whereas the former approaches require deep linguistic knowledge of the respective language, there are graph-based approaches that translate the text into a graph representation with similarity scores as weights for the links between nodes representing the sentences. The second part of this section focuses on such approaches that use PageRank as a scoring mechanism for extracting the summary.

Coherence and Cohesion

Extractive summaries automatically generated from the source document(s) often suffer from poor linguistic quality. Because sentences are extracted out of context, important connections in the form of anaphoric expressions (e.g., pronouns) or discourse structure (e.g., discourse markers such as *therefore*) can be broken and make the summary incoherent and hard to read.

Several approaches to improving the linguistic quality have been introduced. We discuss two main concepts that are important in this context. First, **cohesion** is the link that connects sentences in a meaningful way [26]. It is typically done via anaphoric (or cataphoric) references between sentences. Other linguistic phenomena that support cohesion include substitution, ellipsis, or lexical collocation.

John went to the bank. He wanted to swim in the river.

The two sentences are well connected because we can resolve the anaphoric reference *he* to John and can infer the meaning of *bank* (i.e., *sloping land, especially the slope beside a body of water*) correctly because of the lexical collocation with the words *swim* and *river*.

Related to the concept of cohesion that is often restricted to the sentence-sentence connection is **coherence**. Coherence is often used in the context of a discourse theory modeling how sentences are connected within the entire text. Summaries need to be coherent to be understood and helpful for a user. Hence, the question of how sentences in an extract should be ordered and possibly modified to make the summary more readable becomes important.

To maximize the coherence of a summary, several approaches have been proposed [27, 28, 29]. They are mostly grounded in discourse theories such as RST [24]. The main assumption of RST is based on the observation that text segments are connected via rhetorical relations. A rhetorical relation between text segments can either be explicitly marked via a discourse marker (e.g., *because*) or inferred from the context. Rhetorical relations can express causality between events, elaborate a situation, or move the narration forward.

Work by Marcu and Echihabi [30] built on RST and proposed a rhetorical parser that can also be used for summarization. One of the core ideas of RST is the generation of a discourse tree. The nodes of the tree can combine text spans. Two types of nodes are possible:

• Nucleus & Satellite (hypotactic): The Nucleus contains the more important information supported by the information extracted by the Satellite. The ELABORATION relation, for example, possesses a Nucleus and a Satellite, as in:
Lactose is milk sugar: the enzyme lactase breaks it down.

• Nucleus & Nucleus (paratactic): The multinuclear relation CONTRAST, on the other hand, establishes a contrast between two equally important facts, as in:
For want of lactose, most adults cannot digest milk. In populations that drink milk, the adults have more lactase, perhaps through natural selection.

Figure 12–1 contains an RST discourse tree for the following example text about Mars exploration, as analyzed by Marcu [31]:

[With its distant orbit [1]] [— 50 percent farther from the sun than Earth — [2]] [and slim atmospheric blanket, [3]] [Mars experiences frigid weather conditions. [4]] [Surface temperatures typically average about 60 degrees Celsius (76 degrees Fahrenheit) at the equator [5]] [and can dip to 123 degrees C near the poles. [6]] [Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion, [7]] [but any liquid water formed in this way would evaporate almost instantly [8]] [because of the low atmospheric pressure. [9]] [Although the atmosphere holds a small amount of water, [10]] [and water-ice clouds sometimes develop, [11]] [most Martian weather involves blowing dust or carbon dioxide. [12]] [Each winter, for example, a blizzard of frozen carbon dioxide rages over one pole, [13]] [and a few meters of this dry-ice snow accumulate [14]] [as previously frozen carbon dioxide evaporates from the opposite polar cap. [15]] [Yet even on the summer pole, [16]] [where the sun remains in the sky all day long, [17]] [temperatures never warm enough to melt frozen water.[18]]

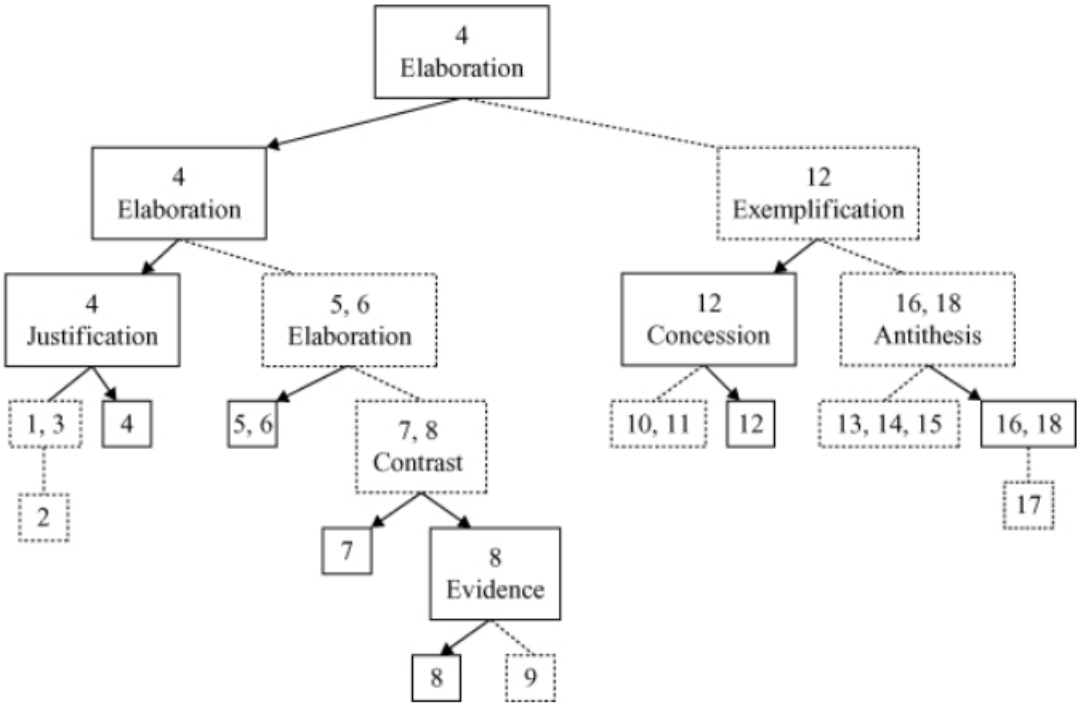

Figure 12–1. The RST structure of the Mars example text (Reproduced from Marcu [31])

The textual units into which this example text is broken are based on clauses and indicated by square brackets. Clause 12 (i.e., *most Martian weather involves blowing dust or carbon dioxide*), for example, states an example for the statement in clause 4 (i.e., *Mars experiences frigid weather conditions*). Here, clause 4 is the Nucleus and clause 12 is the Satellite. The Nucleus & Satellite node is defined in such a way that the Satellite could be deleted from the discourse tree, and the entire discourse would still be coherent. This feature can also be used for summarization purposes, and a discourse tree can be pruned to generate a more concise representation of the entire text [31, 28, 29]. The full text on Mars exploration could therefore be summarized as follows:

Mars experiences frigid weather conditions. Surface temperatures typically average about 60 degrees Celsius (76 degrees Fahrenheit) at the equator and can dip to 123 degrees C near the poles. Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion, but any liquid water formed in this way would evaporate almost instantly.

Most Martian weather involves blowing dust or carbon dioxide. Yet even on the summer pole, temperatures never warm enough to melt frozen water.

Even though these approaches based on discourse theories such as RST provide a coherent summary, they are not easily transferrable to other languages [31] [32].[6]

---

6. Learning discourse parsers from discourse markers, as suggested by Marcu and Echihabi [30] has proven to be difficult, as shown by Sporleder and Lascarides [33].

Discourse parsers for English [31], Japanese [28], and German [29] have been used in the context of summarization systems, but discourse parsers are not widely developed for many other languages.

Even the translation of discourse markers may be difficult because they may cover different semantics. The English marker *since*, for example, can have a causal or purely temporal meaning. To translate this marker correctly into German, the researcher has to choose between *weil* (causal) or *seit* (temporal).

Nevertheless, researchers must consider how coherence can be maintained in a different source language. This area is likely to be a new area to be explored.

Text as Graph

TextRank is an approach to summarization that also utilizes a graph representation [19]. TextRank is related to PageRank, but instead of generating a graph consisting of linked documents, the graph is derived from the text. The sentences in the text are represented as the vertices, whereas the edges between the vertices are weighted by a similarity metric. Similar to PageRank, vertices that are highly connected are “recommended” by other sentences and receive a high rank.

Formally, a graph representation is used for the text where a directed graph $G = (V, E)$ with a set of vertices V and a set of edges $E \subseteq V \times V$ represents links between sentences. The PageRank score is computed on the basis of the number of inlinks $in(V_i)$ pointing to a vertex and the number of outlinks $out(V_i)$ pointing away from a vertex. The PageRank score is then computed via the following formula, where d is the dampening factor indicating the probability of jumping to a new page:[7]

The parameter d is normally set to 0.85

$$S(V_i) = (1-d) + d * \sum_{j \in in(V_i)} \frac{1}{|out(V_j)|} S(V_j)$$

For TextRank, the directed graph becomes an undirected graph and consequently $in(V_i) = out(V_i)$. Instead, the links have weights w_{jk} based on similarity scores between the sentences. The similarity metric defined by Mihalcea and Tarau [19] counts the number of words two sentences share and normalizes by the length of the sentences.

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \wedge w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

The weighted PageRank score is defined as follows:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in in(V_i)} \frac{1}{\sum_{V_k \in out(V_j)} w_{jk}} WS(V_j)$$

A sample newspaper article is shown in Table 12–1. Given the equation in 12.2, similarity scores between all sentences can be computed. The resulting graph that connects all sentences via links contains the scores as the respective weights for each link, as illustrated in Figure 12–2.

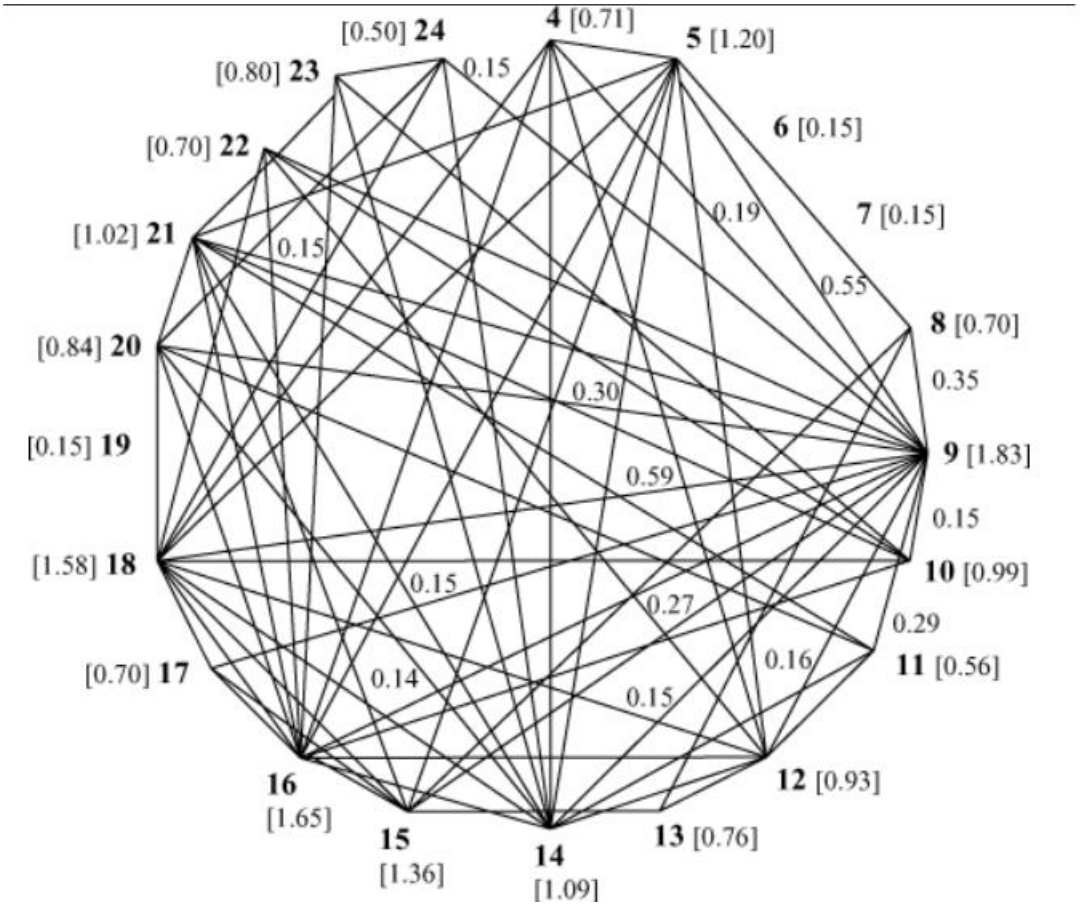

Figure 12–2. A sample graph generated from a text (Reproduced from Mihalcea and Tarau [19])

The PageRank scores for each sentence are derived from the graph, giving high scores to sentences that have links with high weights pointing to them.

TextRank was evaluated with the Document Understanding Conference (DUC) 2002 data, and it was shown that the system was comparable to the top systems within this competition. Given that this approach is unsupervised and does not require any further language-dependent tools except for a similarity metric, this approach would work also with other languages.

A similar approach, called LexPageRank, was developed by Erkan and Radev [5]. LexPageRank also utilizes PageRank, but the similarity scores are computed via cosine similarity, and thresholds for the weights are introduced. They offer LexPageRank as part of the MEAD summarization system, described in more detail in Section 12.2.4.

Learning How to Summarize

Starting with Kupiec, Pedersen, and Chen [23], the idea of learning a classifier that determines which sentences to be included in the summary was introduced. Many different approaches developed over the last decades fall into this category, and they all share the following components listed in Table 12–1 [34, 35]:

1. *An aligned corpus between summaries and their respective document(s).* Some method (e.g., word overlap) has to be employed to match summary sentences and sentences from the text to be summarized.

2. *Feature extractors that generate feature vectors for each sentence.* A feature may be the length of a sentence, the position in the text/paragraph, the overlap of words in the sentence with the title/headings, or the word frequency of the words in the sentence.

3. *A machine learning algorithm that classifies a sentence.* This can be a binary classifier, a multiclass classifier, or a regression model in which a sentence would receive an overall score.

In recent years, several methods for learning the rank of a sentence have evolved. They can be classified via these three categories that differ in terms of what is learned:

Score Each sentence from the training set is assigned a score. This score may be computed by the word overlap of document sentences with the sentences from model summaries. Given the sentence-score combinations, a regression model can be learned. Support vector regression (SVR) can be employed for learning a score for each sentence [36, 37].

Partial order Pairs of sentences are ranked in order to receive a partial order of sentences. A learning to rank method expects pairs of sentences that learns a partial order. Svore, Vanderwende, and Burges [38], for example, use RankNet [39] employing pairwise cross-entropy as a loss function similar to Amini et al. [40], who use an exponential loss function for XML summarization.

Ranks Another way of learning summary sentences is to learn how sentences are ranked in a list. Instead of pairwise ranking, sentences are ranked as a full-order list or at least in several "buckets." Approaches that chose this direction include ListNet [41] and web-based summarization by Wang et al. [42].

To give an example for one of the three machine learning approaches, we describe the partial order–based approach by Amini et al. [40] in more detail. The proposed learning framework uses a scoring function $h : R^n \rightarrow R$ reflecting the best linear combination of sentence features with respect to the learning criterion. The goal of the classification task is to minimize the error of a ranking loss function L_R. The ranking loss L_R is the average number of relevant sentences scored below irrelevant ones in every document $d \in D$.

$$L_R(h, D) = \frac{1}{\mid D \mid} \sum_{d \in D} \frac{1}{\mid S_d^{pos} \mid\mid S_d^{neg} \mid} \sum_{s \in S_d^{pos}} \sum_{s' \in S_d^{neg}} [[h(s) \geq h(s')]] \qquad (12.4)$$

where $[[h(s) \geq h(s')]]$ is a predicate that equals 1, $h(s) \geq h(s')$ and 0 otherwise. $L_R(h, D)$ iterates through all combinations of positive and negative sentences and increases the value for the loss function if the score for a positive sentence is lower than the score for a negative sentence.

Given this loss function, the goal of the ranking algorithm is to learn a scoring function *h* where higher scores are assigned to relevant sentences rather than to irrelevant sentences in the same document.

The loss function should be expressed as an exponential loss function, because [[]] cannot be differentiated. Moreover, the difference between the sentence scores can be computed via the difference in the feature representations of the sentences *s, s′* as in $\sum_{i=1}^{n} \beta_i (s_i - s'_i)$:

$$L_{exp}(D,B) = \frac{1}{\mid D \mid} \sum_{d \in D} \frac{1}{\mid S_d^{pos} \mid \mid S_d^{neg} \mid} \sum_{(s,s') \in S_d^{pos} \times S_d^{neg}} e^{\sum_{i=1}^{n} \beta_i (s'_i - s_i)}$$

Using the exponential loss function has an advantage regarding the computational complexity of the learning algorithm. Equation 12.5 can be simply rewritten, leading to a linear time complexity for computing the exponential loss function:

$$L_{exp}(D,B) = \frac{1}{\mid D \mid} \sum_{d \in D} \frac{1}{\mid S_d^{pos} \mid \mid S_d^{neg} \mid} \sum_{s' \in S_d^{neg}} e^{\sum_{i=1}^{n} \beta_i s'_i} \sum_{s \in S_d^{pos}} e^{\sum_{i=1}^{n} \beta_i s_i}$$

Input: $\bigcup_{d \in D} S_d^{pos} \times S_d^{neg}$, where *D* is a collection of documents and S^{pos} the set of positive (summary) sentence and S^{neg} the set of negative (non-summary) sentences.
Output: Each sentence vector *s* is normalized such that $\sum s_i = 1$; feature weights $F = (\beta_1 \ldots \beta_n)$ are set to arbitrary values; t=0;
repeat
for *i* = 1 **to** *n* **do**

$$\beta_i^{(t+1)} = \beta_i^{(t)} + \Sigma^t$$

end **for**
t = t+1
until Convergence of $L_{exp}(D, F)$
return B^F
Create a summary for each new document *d* by taking the first *n* sentences in *d* with regard to the linear combination of sentence features with B^F.

Amini et al. chose a linear ranking function $h(s, B)$ given a list of features represented as a feature weight vector $B = (\beta_1,\ldots,\beta_n)$ called LinearRank (Algorithm 12–1). The iterative scaling of the feature vector optimizes the loss function described in Equation 12.6 via an update

rule $B^{(t+1)} = B^{(t)} + \Sigma^t$. More precisely, the update function can be described as follows (see Amini et al. [40] for more details):

$$\beta_i^{(t+1)} = \beta_i^{(t)} + \frac{1}{2} log \frac{\sum_{d \in D} \frac{1}{|S_d^{neg}||S_d^{pos}|} \sum_{s' \in S_d^{neg}} e^{h(s', B^{(t)})} \sum_{s \in S_d^{pos}} e^{-h(s, B^{(t)})} (1 - s'_i + s_i)}{\sum_{d \in D} \frac{1}{|S_d^{neg}||S_d^{pos}|} \sum_{s' \in S_d^{neg}} e^{h(s', B^{(t)})} \sum_{s \in S_d^{pos}} e^{-h(s, B^{(t)})} (1 + s'_i - s_i)} \quad (12.7)$$

After generating training data using one of these three approaches, features need to be generated for each sentence. The feature engineering is important because it decides how well the classification task can be learned.

Assuming a setting of a query-based multidocument summarization such as in the DUC or Text Analysis Conferences (TAC) summarization competition, we can utilize frequency information from the overall topic, the query, and the document, as well as the other documents in the clusters. The DUC/TAC task includes at least a set of 25 to 50 documents grouped according to a topic (e.g., steps toward introduction of the Euro) and a query (e.g., describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999). Given this information, the following features have been used by past systems (e.g., Schilder and Kondadadi [37]):

Topic title frequency: ratio of number of words t_i in the sentence s that also appear in the topic title T to the total number of words $t_{1..|s|}$ in the sentence s: $\frac{\sum_{i=1}^{|s|} f_T(t_i)}{|s|}$, where

$$f_T = \begin{cases} 1 & : \quad t_i \in T \\ 0 & : \quad otherwise \end{cases}$$

Topic description frequency: ratio of number of words t_i in the sentence s that also appear in the topic description D to the total number of words $t_{1..|s|}$ in the sentence to the total number of s: $\frac{\sum_{i=1}^{|s|} f_D(t_i)}{|s|}$,

where $f_D = \begin{cases} 1 & : \quad t_i \in D \\ 0 & : \quad otherwise \end{cases}$

Content word frequency: the average content word probability $p_c(t_i)$ of all content words $t_{1..|s|}$ in a sentence s. The content word probability is defined as $p_c(t_i) = \frac{n}{N}$, where n is the number of times the word occurred in the cluster and N is the total number of words in the cluster: $\frac{\sum_{i=1}^{|s|} p_c(t_i)}{|s|}$

Document frequency: the average document probability $p_d(t_i)$ of all content words $t_{1..|s|}$ in a sentence *s*. The document probability is defined as $p_d(t_i) = \frac{d}{D}$, where *d* is the number of documents the word t_i occurred in for a given cluster and *D* is the total number of documents in the cluster: $\frac{\sum_{i=1}^{|s|} p_d(t_i)}{|s|}$

Other features that turned out to be useful include headline frequency (the average headline probability of all content words in a sentence *s*), sentence length, sentence position, TF-IDF score for the words, *n*-gram frequency, and named entity frequency in a sentence.
These machine learning approaches work well if an aligned corpus exists for all languages in a multilingual setting or could be easily generated. However, this may not always be the case. To work around the data problem, approaches have been proposed to bridge the gap between documents written in different languages. Ji and Zha [20] propose an algorithm that aligns the (sub-)topics of a pair of multilingual documents and summarizes their correlation by sentence extraction. They carry out this alignment via a weighted bipartite graph representing sentences of the two documents. Then, sentences are translated via a machine translation program into the other language, and a weight matrix is generated that consists of the similarity scores between the translated sentences and the sentences in the original language. Note that the machine-translated sentences do not need to be perfect translations, because the goal is to capture the similarity between the sentences.

From the weight graph, a subgraph of highly correlated sentences can be derived. These sentences present the main topic the two documents share. In addition, a biclustering algorithm is employed to derive further subtopics represented by clustered sentences from each document.

Multilingual Summarization

Challenges

We reviewed the most important approaches to summarization with pointers to how some of the techniques would fare with multilingual summarization and that most of the current summarization approaches often rely on language-dependent resources or tools (e.g., rhetorical parsers, cue phrase lexicons). Some approaches exist that allow for language-independent summarization by generating summaries from a representation that is abstracted from the source language.

The following list is a summary of features that need to be considered for a multilingual summarization system. In particular, these are challenges we must face when working with multiple languages.

Tokenization Because languages encode word boundaries differently, tokenization is a first obstacle to overcome when building a summarization for different languages. Languages such as English identify token boundaries via whitespace and punctuation, but other languages such as Chinese require a more complex segmenter to extract tokens from a stream of text that does

not contain any whitespaces. A token is a word in languages such as English but may be something else in different languages. Other languages (e.g., Arabic) that possess a rich morphology may require an even more fine-grained tokenization up to the morph level.

Anaphoric expressions The identification of anaphora (i.e., pronouns, discourse markers, and definite noun phrases) can help to make a summary more cohesive. Some techniques exist for monolingual summarization, but multilingual summarization also faces the challenges that names are written differently and that discourse markers have different semantics in different languages.

A knowledge-poor approach to anaphora resolution proposed by Mitkov [4] uses, in addition to number and gender agreement, a number of simple indicators (e.g., definiteness, givenness, verb classes) that are aggregated to generate a score for potential antecedent candidates.

Discourse structure The identification of document structure can help to improve the coherence of a summary. Different languages, however, may express the structure of a text differently.

Machine translation The state-of-the-art machine translation technology has not yet reached a level sufficient to guarantee high-quality translation. When designing a multilingual summarization system, developers must answer the question of when machine translation should be employed in the system. If text is generated at the start, components developed for the source language can be reused (e.g., tokenizer). If translation is done after identifying the summary-worthy sentences, language-dependent systems have to be used to preprocess the text accordingly.

Systems

There are three major summarization systems available that have some multilingual capabilities: MEAD, Summa, and NewsBlaster.

The **MEAD**[8] platform for multilingual summarization and evaluation offers several different summarization algorithms based on position, centroid, longest common subsequence, and keywords. MEAD is a publicly available platform for multilingual summarization and evaluation written in Perl. This framework can be used to easily adapt surface-based approaches discussed earlier or to train a classifier for detecting summary-worthy sentences. Moreover, it allows users to train their own summarization approach by offering support for machine learning algorithms such as decision trees, support vector machines, and maximum entropy.

The centerpiece of the MEAD framework is the centroid-based summarization approach. A centroid is a set of words that are significant for a cluster of documents. Relevant documents and summary sentences within these cluster of documents are extracted on the basis of the centroids they contain.

The algorithm that generates the clusters is called CIDR [43]. CIDR generates clusters of documents that share the same words among them. Starting with one document, the algorithm compares the similarity to other clusters. Documents are represented via word vectors. The values for the words are derived from the document frequency and the inverse document frequency (TF-IDF).

Each cluster has a centroid that can be described as a pseudodocument containing only the most important words with the highest TF-IDF scores. The word vector representing the cluster is composed of the weighted averages of the corresponding TF-IDF values from the documents that are already clustered.

The algorithm starts with the first document and puts it in a cluster containing only this document. New documents are compared to the word vector representing the cluster and are assigned to the respective cluster if the cosine similarity between the cluster vector and the document vector falls below a predefined threshold.

The cosine similarity is the cosine of the angle between two (word) vectors:

$$\text{sim}(A, B) = \text{cosine}\,\theta = \frac{A \cdot B}{|A||B|} \qquad (12.8)$$

If the document cannot be assigned to any cluster, a new cluster is formed containing so far only this single document.

Another graph-based algorithm also comes with MEAD: LexPageRank [5] is based on computing the PageRank score of the sentences in the lexical connectivity matrix with a defined threshold. LexPageRank is similar to TextRank in using PageRank for scoring the sentences. It differs from PageRank in the way the weights for the graph are generated. LexPageRank uses the cosine similarity to generate the weights between sentences, and it allows for using thresholds on the cosine similarity: a link between sentences is generated only if the cosine similarity is above a certain threshold (e.g., 0.1).

An alternative summarization system is **SUMMA**—a summarization tool that can run as a GATE plug-in or as a standalone application. The tool is extendable and allows users to add their own scoring functions in addition to the provided position- or centroid-based scoring functions. Various similarity metrics such as cosine and *n*-gram similarity are also included in this tool.

Finally, the **NewsBlaster** summarization system from Columbia University (see Figure 12–3) possesses a multilingual extension [9][9] and enables users to browse news written in multiple languages from multiple sites on the Internet. The approach utilizes a well-tested approach to document clustering [44] on machine-translated English text.

9. The website http://newsblaster.cs.columbia.edu summarizes news (and images) from (English) news websites.

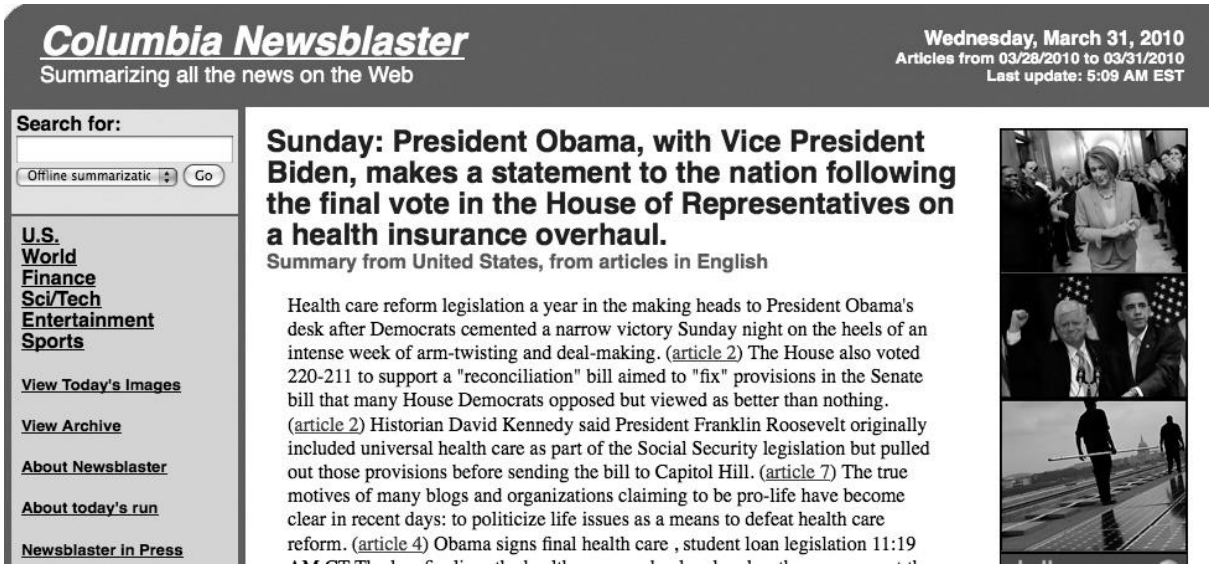

Figure 12–3. A sample page generated by NewsBlaster (Courtesy Columbia University)

The summarization is carried out by the Columbia summarizer [45] that clusters machinetranslated text from non-English documents. The English version is online, but no multilingual version seems to be available.

Evaluation

Determining the quality of system-generated summaries is one major challenge in summarization research. There are two ways to measure summary quality. Indirectly, **extrinsic** evaluations measure the usefulness of summaries by measuring how much they can help in performing another information-processing task. However, **intrinsic** evaluations are being actively pursued because they directly measure and reflect summary quality and can be used in various stages in a summarization development cycle. Summary comparison (i.e., content coverage calculation) is used as the preferred intrinsic evaluation method. Comparisons are performed by assessing how much information from a reference summary is included in a peer summary. **Reference** summaries are usually written by humans to serve as the goldstandard summaries in a comparison. A **peer** summary can refer to any summary whose quality is being measured and thus can be a system-generated summary if we are measuring the quality of a summarization system or can be a human-written summary if we are analyzing the reference summaries for their qualities.

Two directions in designing evaluation methodologies for summarization are manual evaluation and automated evaluation. Naturally, there is a great amount of confidence in manual evaluations because humans can infer, paraphrase, and use world knowledge to relate text units with similar meanings but different wording. Human efforts are preferred if the evaluation task is easily conducted and managed and does not need to be performed repeatedly. However, when resources are limited, automated evaluation methods become more desirable. Creating such an automatic evaluation methodology that can be readily applied to a wide range

of summarization tasks proves to be quite complicated. This section discusses both manual and automatic evaluation methodologies in detail.

12.3.1. Manual Evaluation Methodologies

One fascinating aspect of automatic summarization is the high degree of freedom when summarizing or compressing information presented in either single or multiple input texts. The process of choosing and eliminating informational pieces largely depends on task definition, topic in question, and domain and prior knowledge. When asked to perform the summarization task, even human summarizers would disagree on what goes into the summaries from original texts. This curious fact was discovered and analyzed through performing manual evaluation exercises.

Three most noticeable efforts in manual evaluation are the Summary Evaluation Environment (SEE) of Lin and Hovy ([46] and [47]), Factoid of Van Halteren and Teufel [48], and Pyramid of Nenkova and Passonneau [49].

Summary Evaluation Environment

SEE provides a user-friendly environment in which human assessors evaluate the quality of a system-produced peer summary by comparing it to an ideal reference summary. Summaries are represented by a list of summary units (sentences, clauses, etc.). Assessors can assign full or partial content coverage scores to peer summary units in comparison to the corresponding reference summary units. Grammaticality can also be graded unit-wise. One unique feature of this work is that it allows systematic identification of summary units and facilitates partial matches.

The Factoid Method

The goal of the Factoid work is to compare the information content of different summaries written on the same text and determine the minimum number of summaries needed to achieve stable consensus among human-written summaries. Van Halteren and Teufel studied 50 summaries that were created on the basis of one single text. For each sentence, its meaning is represented by a list of atomic semantic units called **factoids**. Here semantic atomicity means the amount of information associated with a factoid can vary from a single word to an entire sentence. Factoids from all summaries are collected and analyzed. Those expressing the same meaning or carrying the same amount of information across multiple summaries are identified manually as semantically similar. As the number of manually created summaries increases and reaches a certain level, the set of factoids stabilizes and remains largely unaffected even when new summaries and their corresponding factoids are added to the set. Ideally, the gold-standard human-written summaries should be a stable set before we start using them in measuring the quality of the system-generated summaries through content comparison. The Factoid method shows 15 summaries are needed to achieve stable consensus among reference summaries. In practice, the number of human-written summaries is much lower than the required quantity because of the resource-intensive nature of information-processing tasks.

The Pyramid Method

The Pyramid method is an extension of the Factoid method carried out on a larger scale. Nenkova and Passonneau show that only six summaries are required to form stable consensus from reference summaries. This lowered requirement on the number of reference summaries is empirically proven by the authors and achieves the primary goal of reliably differentiating the scoring of summaries. Summarization content units (SCUs) were originally defined as units that are not bigger than a clause but later were redefined as larger than a word but smaller than a sentence because clauses can still carry multiple semantic units.

The process of finding similar SCUs starts by finding similar sentences and proceeds to identifying finer-grained inspection of more tightly related subparts. After all the SCUs are identified and compared, they can be partitioned into a pyramid structure, as illustrated in Figure 12–4. A specific level in the pyramid indicates the number of summaries in which this level's SCUs occurred. SCUs that appeared in all summaries appear at the top of the pyramid because there are only a small number of them. SCUs from lower levels of the pyramid can be used to show the differences in human summary writers' understanding, interests, and knowledge of the summary topics. And the large number of these SCUs from the bottom levels demonstrates the difficulty in summarization. Using the example from Nenkova and Passonneau [49], the variety of summaries can be shown by the following summaries where underlined text indicates the SCUs that are shared (these four sentences come from four different summaries):

A. In 1998 two Libyans indicted in 1991 for the Lockerbie bombing were still in Libya.

B. Two Libyans were indicted in 1991 for blowing up a Pan Am jumbo jet over Lockerbie, Scotland, in 1988.

C. Two Libyans, accused by the United States and Britain of bombing a New Yorkbound Pan Am jet over Lockerbie, Scotland, in 1988, killing 270 people, for 10 years were harbored by Libya who claimed the suspects could not get a fair trial in America or Britain.

D. Two Libyan suspects were indicted in 1991.

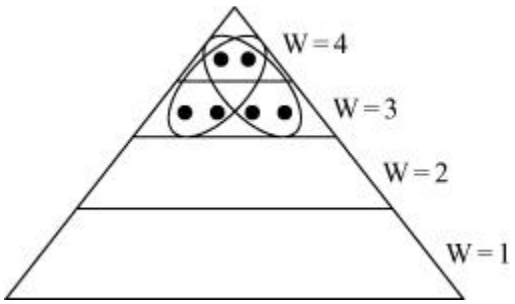

Figure 12–4. A pyramid of four levels (Reproduced from Nenkova and Passonneau [49])

we obtain two SCUs:

SCU1(weight=4): two Libyans were officially accused by the Lockerbie bombing

A. [two Libyans] [indicted]

B. [Two Libyans were indicted]

C. [Two Libyans,] [accused]

D. [Two Libyan suspects were indicted]

SCU2(weight=3): the indictment of the two Lockerbie suspects were in 1991

A. [in 1991]

B. [in 1991]

D. [in 1991]

The scoring of peer summaries is precision based. A peer summary would receive a score that is a ratio of the sum of the weights of its SCUs to the sum of the weights of an optimal summary with the same number of SCUs. Suppose we created a pyramid of SCUs from a set of reference summaries and found 10 SCUs in the peer summary, and only one SCU from the peer summary is found to have matched the SCUs from the pyramid. The SCU that is found in the pyramid had a weight of 1 (only one occurrence in all of the reference summaries). Then the pyramid score for the peer summary is 1/10 = 0.1.

Although the Pyramid method shares the same high cost as other manual evaluation exercises, it has gained community acceptance as the preferred manual evaluation methodology. It has been carried out by conference and task participants in the DUC and DUC's follow-up program, TAC.

Another positive side effect from running the Pyramid experiment on a large scale and on a repeated basis is the large set of human-written summaries and their corresponding semantic units. In designing and tuning summarization systems, the semantic units can be used as gold-standard data that can facilitate research beyond sentence-level extraction.

Responsiveness

In addition to providing content coverage evaluations through manual efforts, at TAC a score of 1 to 5 (1 to 10 since TAC 2009) on responsiveness is also given for a summary in question. This score does not reflect a comparison with the reference summary but solely reflects the quality of the evaluated summary on both content coverage and linguistic quality (two qualities that were measured separately at previous DUCs).

12.3.2. Automated Evaluation Methods

Evaluation systems that take reference and peer summaries as input and perform text comparison to produce evaluation results are called automated evaluations. To test and validate the effectiveness of an automatic evaluation metric, researchers must show that the automatic evaluation results correlate with human assessments highly, positively, and consistently [50].

ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [50, 51] is one of the first automatic summarization evaluation metrics proposed. It is an automatic evaluation package that measures a number of *n*-gram co-occurrence statistics between peer and reference summary pairs. ROUGE was inspired by BLEU [52], which was adopted by the machine translation community for automatic machine translation evaluation. While BLEU is precision related, ROUGE is a recall-oriented metric.

Instead of producing one figure that quantifies the summary quality, ROUGE produces a set of scores that can be used to interpret system-generated results:

• *ROUGE-N* : computes the *n*-gram recall score between a peer summary over all of its corresponding reference summaries. The following formula shows how to compute the ratio of matched *n*-grams from the peer summary and the total number of *n*-grams from the reference summaries:

$$ROUGE-N = \frac{\sum_{S \in \{\text{Reference Summaries}\}} \sum_{\text{gram}_n \in S} \text{Count}_{match}(\text{gram}_n)}{\sum_{S \in \{\text{Reference Summaries}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \quad (12.9)$$

• *ROUGE-L*: matches the longest common sequence (LCS) between two textual units, which finds the longest in-sequence matches instead of consecutive matches. It is not necessary to predefine a length limit as with *n*-gram matches. This metric also reflects the possible differences in sentence structure among the units being compared. To estimate the similarity between two summaries *X* of length *m* and *Y* of length *n*, a LCS-based F-measure is computed as follows:

$$R_{\text{lcs}} = \frac{\text{LCS}(X,Y)}{m} \quad (12.10)$$

$$P_{\text{lcs}} = \frac{\text{LCS}(X,Y)}{n} \quad (12.11)$$

$$F_{\text{lcs}} = \frac{(1+\beta^2) R_{\text{lcs}} P_{\text{lcs}}}{R_{\text{lcs}} + \beta^2 P_{\text{lcs}}} \quad (12.12)$$

LCS(*X*, *Y*) is the longest common subsequence between *X* and *Y*, and $\beta = P_{\text{lcs}}/R_{\text{lcs}}$ when $\partial F_{\text{lcs}}/\partial R_{\text{lcs}} = \partial F_{\text{lcs}}/\partial P_{\text{lcs}}$.

• *ROUGE-W* : computes the weighted LCS. While *ROUGE-L* finds the LCS, it does not penalize matches that have gaps in them. It simply indicates words appeared in the same sequence if there is a match. *ROUGE-W* differentiates matched word sequences that are consecutive and those that are nonconsecutive by charging a gap penalty. This gap penalty is reflected in a weighting function that gives higher reward to consecutive matches than to nonconsecutive matches.

• *ROUGE-S* : calculates the skip-bigram co-occurrence statistics. A skip-bigram is any two ordered words that occurr in a sentence, with arbitrary gaps between them. With a gap of 0, it is equivalent to *ROUGE-N* where *n*=2 on bigram matchings. Skipbigram-cased F-measure is computed as follows:

$$R_{\text{skip2}} = \frac{\text{SKIP2}(X,Y)}{C(m,2)} \tag{12.13}$$

$$P_{\text{skip2}} = \frac{\text{SKIP2}(X,Y)}{C(n,2)} \tag{12.14}$$

$$F_{\text{skip2}} = \frac{(1+\beta^2)R_{\text{skip2}}P_{\text{skip2}}}{R_{\text{skip2}} + \beta^2 P_{\text{skip2}}} \tag{12.15}$$

• *ROUGE-SU* : supplements *ROUGE-S* with a fallback-counting strategy, unigram matching, to address the case where two sentences do not have any skip-bigram matches. Without this supplement, *ROUGE-S* punishes sentences that have different word-order occurrences but could have the same content.

Evaluation results produced by ROUGE are highly correlated with human-based evaluation such as responsiveness. Correlations are expressed by Spearman ranking and Pearson correlation coefficients. Spearman correlation coefficient is used to show the correlation between the ranking orders:

when there are *n* rank pairs (x_i, y_i)

$$p = 1 - \frac{6\sum(x_i - y_i)^2}{n(n^2-1)} \tag{12.16}$$

Pearson correlation coefficient is computed on raw scores (X_i, Y_i) instead of ranks, as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{12.17}$$

One added advantage with ROUGE as an automatic evaluation package is that it does not have dependencies on other language-processing tools, such as various types of parsers, though it provides options to have stemming and part-of-speech tagging activated if so desired.

Basic Elements (BE)

BE [53] was proposed as an approach to automatic evaluation based on the concept of minimal semantic units. A basic element is a semantic unit extracted from a sentence such as subject-object relation, modifier-object relation. Several different syntactic and dependency parsers can be utilized to produce BEs: Charniak's parser [54], Collins's parser [55], Minipar [56], and Microsoft Logical Forms [57]. The BE breaker model takes in a parse tree and applies a set of heuristics to extract from the tree a set of smaller constituents, or BEs. Collins's and Charniak's parse trees are syntactic and do not include the semantic relations between head and modifier words. Minipar, however, is a dependency parser that automatically produces <*head*; *modifier*; *relation*> triples. The default version of the BE package creates BEs from Minipar's dependency trees. Systems are graded by measuring the overlap between system-generated summary BEs and the human-written summary BEs.

To show that BE is effective in evaluating summarization systems, it was tested on DUC 2003 results, comparing system-produced peer summaries against gold-standard reference summaries. Correlation figures are calculated by comparing rankings and average coverage scores of systems (peers and baselines) of those documented by DUC and produced by BE. Spearman rank-order correlation coefficient and Pearson correlation coefficient are computed for the validation tests. While multiple options are available when running the BE package, the authors show that running BE-F, where Minipar is used to extract BEs, without differentiating relations between the head and modifier, and taking the lemma of all words, has the highest correlations on both Spearman and Pearson when evaluating multidocument results.

When developing and tuning a summarization system, it is common to run both ROUGE and BE to analyze system-produced results thoroughly. However, because of BE's dependency on parsers, researchers may not be able to run the BE package in a multilingual scenario if the underlining parser is not available in that particular language.

Related Work

Both ROUGE and BE are used frequently because of their simplicity and high correlations with human judgments. However, the textual unit comparisons between peer and reference summaries are still limited to the matching of lexical identities. Efforts have been made to move toward measuring semantic closeness using paraphrases and synonyms. The ParaEval [58] method facilitates paraphrase matching in an overall three-level comparison strategy. At the top level, favoring higher coverage in reference, it performs a greedy search to find multiword to multiword paraphrase matches between phrases in the reference summary (usually human-written) and those in the peer summary (system-generated). The nonmatching fragments from the previous level are then searched by a greedy algorithm to find single-word paraphrase/synonym matches. At the third and the lowest level, ROUGE-1 matching is run on the remaining texts. This tiered design for summary comparison guarantees at least a ROUGE-1 level of summary content matching if no paraphrases are found. Compared to the initial release of ROUGE, ParaEval gained slightly on correlations. The paraphrases were extracted using machine translation alignment data based on the assumption that phrases that are frequently translated interchangeably are likely to be paraphrases of each other [59]. This

method is most effective for machine translation evaluations [60] because in translation the goal is to produce text in a target language that completely corresponds to the original text without any compression and redundancy complications that result in more word choices.

12.3.3. Recent Development in Evaluating Summarization Systems

In 2004, Filatova and Hatzivassiloglou [61] defined **atomic events** as major constituents of actions described in a text linked by verbs or action nouns. They believe that major constituents of events are marked as named entities in text, and an atomic event is a triplet of two named entities connected by a verb or an action-denoting noun all extracted from the same sentence. Atomic events are used in creating event-based summaries and are not modeled into any evaluation method.

Tratz and Hovy [62] provide an improvement to the original BE method, which facilitates surface transformation of the basic elemental text units, called BE with Transformations for Evaluation (BEwTE). This work is based on the idea that naïve lexical identity matching does not take into account the equivalency between text units that have exact or similar words but are structured differently syntactically or semantically. To perform the transformation automatically, a set of transformation heuristics are written, and the sequence in which they are to be run is defined. This rule-building process is manually performed. The scoring on BEs in the peer summaries is produced on the basis of a greedy matching algorithm and is normalized by total weights of corresponding reference BEs. Correlation tests performed on past DUCs and TACs results show higher correlations compared to the original BE method and ROUGE. In addition to the dependency of language-processing tools, BEwTE requires significant human efforts and linguistics knowledge to create the transformation rules and the order of executing them when operating in a multilingual environment.

Louis and Nenkova [63] show that automatic summarization evaluations can be run without manually written reference summaries. This work hypothesizes that gold-standard summaries have low divergence with texts on word probability distributions where low divergence indicates high similarity. Kullback Leibler (KL) and Jensen Shannon (JS) divergence between reference and peer summaries are used as summary scores. Topic signatures [64], a useful summary creation feature, are shown to be also indicative in terms of summary evaluation where a high concentration of signatures show higher summary content quality.

Another innovative work, AutoSummENG (Automatic Summary Evaluation based on *n*-gram Graphs [65]), creates summary graphs where nodes are *n*-grams and edges are relations between the *n*-grams. Summary comparison is a comparison between the reference and peer summary graphs. Relations are modeled by taking the fixed-length windows of context information surrounding the *n*-grams. Relation edges are weighted to indicate the distance between the *n*-gram nodes and the number of occurrences in text. This work shows higher correlations than other automatic methods. An added advantage of this work is its language neutrality, not needing language-dependent tools.

12.3.4. Automatic Metrics for Multilingual Summarization

Summarization is a complex natural language processing task, and its evaluation presents challenges that facilitate active research development. Even though most of the automatic evaluation methods are lexical-identity matching based, it is important to keep in mind that statistically they do provide a reliable measure of system-generated summaries' quality. While good systems and bad systems are easily distinguishable by these methods, they have difficulties in identifying nuances presented by comparable systems. Given the imperfections from these various evaluation methods, it is essential for summarization system designers to understand the task being undertaken and conduct their own detailed error analyses. When moving toward multilingual summarization, there are even fewer evaluation packages that can be used. Table 12–2 summarizes the language neutrality of all the automated methods discussed in this section.

Table 12–2. Evaluation metrics used for summarization and their requirements on language-dependent processing tools

Method Name	Language-Processing Tool Dependent	Notes
ROUGE	No	
BE	Yes	Syntactic and/or dependency tree parsers
ParaEval	No	Machine translation alignment data
BEwTE	Yes	Basic element dependencies and linguistic knowledge
Divergence ([63])	No	
AutoSummENG	No	

12.4. How to Build a Summarizer

This section provides a blueprint for building a summarization system. We do not assume any particular programming language or development framework, because a summarizer can be built in any programming language. This section contains pointers to different tools and frameworks that can be used for building a multilingual summarization system either from scratch or on an already existing framework.

A general schema of a multilingual summarization system is shown in Figure 12–5. The general schema reflects the three stages commonly used for summarization described in Section 12.2.1. First, the documents must be analyzed. Depending on what type of multilingual summarization system we intend to build, our input documents are of one language (i.e., translingual) or multiple languages (i.e., crosslingual). Multiple multilingual corpora are listed in Section 12.5. The choice of languages influences which subsequent tools we need (§12.4.2).

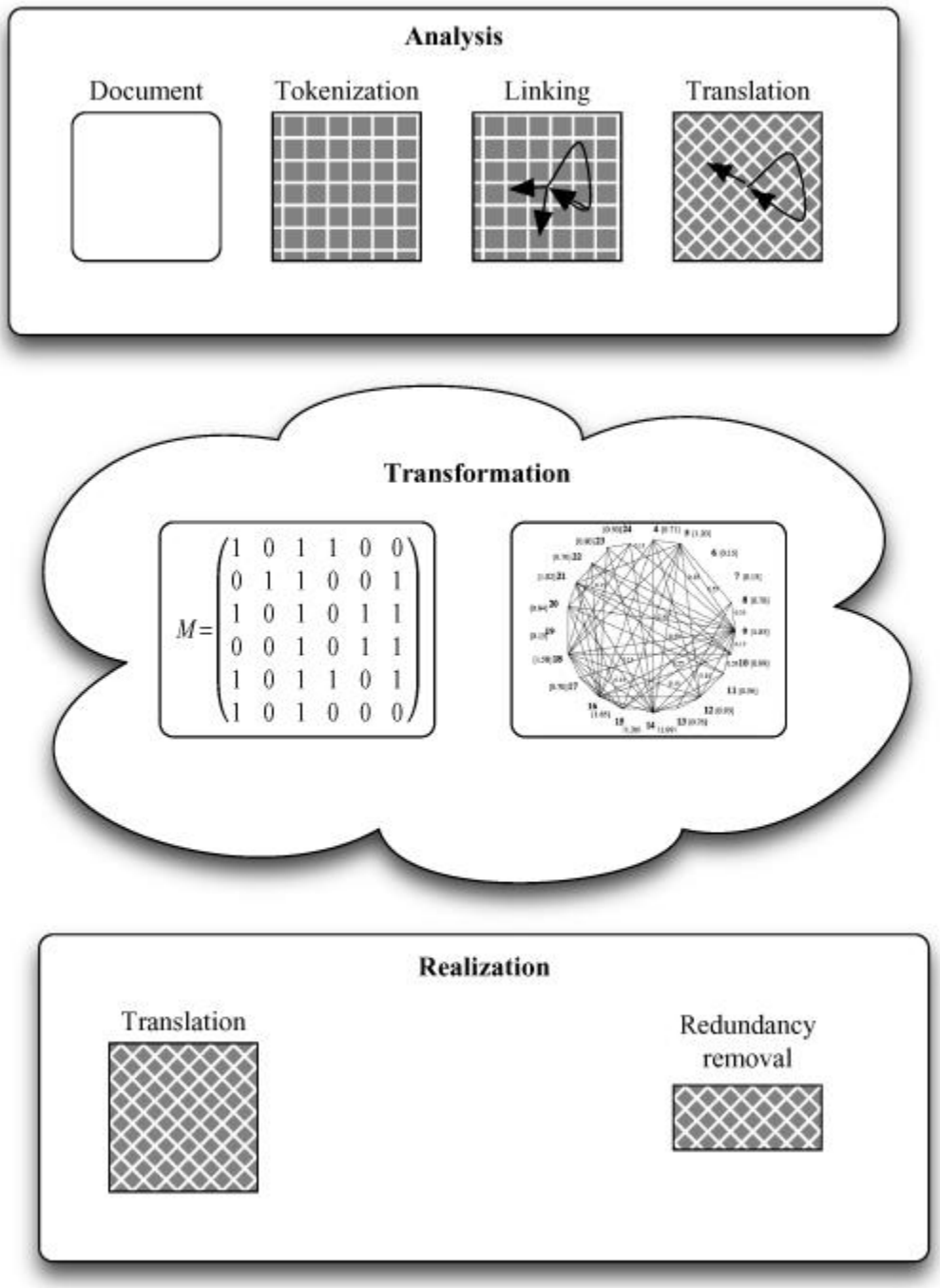

Figure 12–5. A blueprint for a multilingual summarization system

After collecting the input data, tokenization tools need to be applied. Tokenization will probably go beyond simple whitespace tokenization, particularly for languages that do not separate words with whitespace or punctuation (e.g., Chinese). It may also be necessary to tokenize into finer categories than words if the language in question possesses a rich morphology (e.g., Arabic) or allows for generating many compound expressions such as compound nouns in German. This analysis step also includes other separation and chunking techniques, such as sentence splitting, chunking, and parsing. During the tokenization process, some bookkeeping regarding the frequency of all tokens, *n*-grams, chunks, and so on, needs to be carried out.

The next step involves further analysis of the tokenized text and leads to linking tokens via coreference resolution (i.e., *Microsoft-the company*) or simply the preceding or succeeding context (e.g., *Today–Microsoft–announced*).

The final component in the analysis step is concerned with translating the input content. This may happen after or before the tokenization and linking. It may also be postponed until after the transformation step.

The second step in a summarization system is concerned with the transformation of the analyzed text. The choices made in the analysis text determine what kind of representation goes into the transformation modules. Section 12.2 offered different approaches to summarization to choose from. The output of this step consists of a reranked list of sentences/chunks where top-ranked text units are most summary-worthy.

Eventually, the summary is generated in the final realization step. Redundancies in the output text can be removed by applying different redundancy removal techniques described in the literature (e.g., QR, cosine similarity). If no machine translation has been applied to the input text, it now needs to be carried out to produce text in the target language.

12.4.1. ***Ingredients***

To develop a multilingual automatic summarization program, data in various forms is a prerequisite. Ideally, we could draw from the different summarization corpora provided by NIST[10] and the LDC.[11]

In most cases, however, we will not have any data available, and creating gold-standard data may be too expensive. In such cases, we can either train and test on available data and transfer the system to this new domain or try to adapt the system to the new domain. Clearly, new domain adaptation is an interesting research direction on its own, and we refer to recent efforts, as described by Daumé and Marcu [66] or presented at the Association for Computational Linguistics 2010 workshop on domain adaptation.[12]

A good starting point for training your summarizer is the collection of data provided by the National Institute of Standards and Technology for DUC and TAC:

The system can be implemented in different frameworks. We list some here as possible suggestions but do not want to imply in any way that this is an exclusive list:

• **UIMA**[13] stands for Unstructured Information Management Architecture. It was developed by IBM and is now an Apache project. It has a component architecture and software framework implementation for the analysis of unstructured content like text, video, and audio data. This framework is Java-based but also available in C++.

GATE,[14] a General Architecture for Text Engineering, was developed and first released in 1996 by the University of Sheffield. GATE is a general framework for natural language

processing that contains many different language processing tools mainly written in Java that are useful for developing a summarization system. Moreover, the SUMMA toolkit developed by Horacio Saggion can be used as a GATE plug-in.

NLTK,[15] the Natural Language ToolKit, offers natural language processing tools written in Python. The toolkit was developed for teaching natural language processing techniques in Python and offers a wide range of packages covering different taggers, stemmers, parsers, as well as corpus processing tools and classification and clustering algorithms.

R[16] is a free software environment for statistical computing and graphics and not a natural language processing tool, but it offers easy access to many of the techniques discussed in Section 12.2. There are many machine learning packages[17] as well as tools for graph processing,[18] including implementations for running PageRank.

In addition to the general frameworks that provide some initial support for writing your own summarization system, you may also start with one of the open source summarization systems introduced earlier: MEAD and SUMMA.

12.4.2. Devices

A number of tools are required for building a summarization system. In particular, at least a machine translation program is needed if more than one language is used that is different from the target language.

The following is an "ingredients" list that offers lots of suggestions on how to substitute one ingredient with another one.

Tokenizer/sentence splitter Different tools mentioned earlier (e.g., NLTK, GATE) contain tokenizers and sentence splitters. Here are some other natural language processing tools that offer similar functionalities:

lingPipe offers several different Java libraries for the linguistic analysis of human language, including sentence segmentation, Chinese word segmentation, and tokenization for English.

openNLP combines different open source projects related to natural language processing and offers a sentence splitter and tokenizer.

Machine translation program Several machine translation toolkits are available that allow researchers to train their own statistical machine learning model:

Feature extractors To run machine learning experiments or to generate a graph representation, researchers must extract features from the documents for each sentence (or word). There are tools available that facilitate this process and offer out-of-the-box feature extractors that researchers are likely to use (e.g., *n*-gram-based features). A tool that works with the UIMA

framework was developed by the University of Boulder, Colorado, and is called ClearTK[19] [67].

Instructions

The two previous sections contain enough pointers to get all the necessary ingredients and devices for creating your own summarization system. At the end of this chapter, we discuss how to utilize the blueprint of a summarization system (see Figure 12–5).

First, you must decide where to use the machine translation part. There are two possibilities. You can either translate first or translate later. The latter method has two advantages: a summary system that adopts this approach would probably be faster because only the best summary sentences need to be translated, a fact you should consider when large documents need to be summarized. Moreover, translation errors may degrade the summarization process if that is done on the target language only. This will surely be true if summarization techniques require high-level linguistic features such as a parse tree, but it may not be such a problem if clustering or graph-based approaches based on bag-of-words or *n*-gram features are chosen. Depending on this decision, the other components need also to be available and need to produce output of a certain quality (e.g., tokenizer, chunker).

Second, the overall approach needs to be determined. We summarized different approaches in Section 12.2 and pointed in particular to clustering and graph-based approaches that should work well with crosslingual or translingual summarization. The decision for this part of the system should be guided by the available language resources and the quality of the machine translation component. The output will be a ranked list of sentences with the best sentences being the best summary sentences.

An additional (optional) component would address the generation part of the system. For multidocument summarization, this component would have to make sure that no redundant sentences are selected, and for a multilingual summarization system, it must ensure that the correct translation for an entity or concept has been selected. Some systems offer a solution for this problem of redundancy removal (e.g., Carbonell and Goldstein [18]) or the selection of names in other languages (e.g., Mani, Yeh, and Condon [14]).

Finally and most importantly, you must to determine which evaluation methods should be used. The method should depend on how the system is used and may be intrinsic or extrinsic. It is recommended to set up experiments with the different parameters of your system and record the achieved results. These records will help you to decide on which parameters lead to the best system configuration.

12.5. Competitions and Data Sets

12.5.1. Competitions

DUC The Document Understanding Conference was carried out by the National Institute of Standards and Technology from 2001 to 2007 and focused on making progress in summarization research and providing a forum for researchers to participate in large-scale experiments. The tasks investigated include single summarization as well as multidocument summarization, mostly in English with the exception of DUC 2003, which involved summarization from Arabic to English translations.

TAC The Text Analysis Conference is the successor meeting for DUC since 2008. Tasks include query-based multidocument summarization, opinion-based summarization, as well as update summarization. In 2011, TAC included a multilingual pilot task. In 2012, the TAC entity linking task will be truly multilingual, covering English, Chinese and Spanish.

MSE The Multilingual Summarization Evaluation (MSE) 2005 and 2006 focused on multidocument summarization of the English and Arabic portions of the TDT-4 corpus, which contains 41,728 Arabic documents and 23,602 English documents. As for the DUC 2003 summarization task, summarizations were generated from the Arabic translations from the original English news articles. First, clusters were created by running a clustering algorithm developed by the University of Columbia over the the TDT4 corpus. Next, the Information Sciences Institute's machine translation system was used to translate the Arabic data. Interestingly enough, the best system in 2005 used only English sentences as input.

12.5.2. Data Sets

• SummBank (Cantonese, English) consists of 18,147 aligned bilingual (Cantonese and English) article pairs from the Information Services Department of the Hong-Kong Special Administrative Region of the People's Republic of China: Multilingual Summarization Evaluation (Arabic, English)

• Crossdocument Structure Theory Bank (CSTBank) (English): data annotated according to crossdocument structure theory (CST), a functional theory for multidocument discourse structure related to rhetorical structure theory

The New York Times Annotated Corpus (English) contains over 1.8 million articles written and published by the *New York Times* between January 1, 1987, and June 19, 2007, with article metadata provided by the New York Times Newsroom. The corpus provides over 650,000 article summaries written by library scientists. Although it is a monolingual corpus, it offers a normalized index for people, organizations, locations, and topic descriptors, which could be helpful for mapping entities across documents.

The Language Understanding Annotation Corpus (Arabic, English) contains over 9,000 words of English text (6,949 words) and Arabic text (2,183 words) annotated for committed belief, event and entity coreference, dialog acts, and temporal relations.

Topic Detection and Tracking (TDT) corpora covered multiple years of data creation (English, Arabic, Mandarin Chinese). TDT2 Multilanguage Text corpus contains news data collected daily from nine news sources in two languages (American English and Mandarin Chinese) over a period of six months (January through June 1998).

sources found in TDT2 plus two additional English television sources. For this corpus, the daily collection took place over a period of three months (October through December 1998).

English, Arabic, and Chinese news text (broadcast news transcripts, and newswire data) used in the 2002 and 2003 Topic Detection and Tracking technology evaluations.