Chapter 5

Language Modeling

Katrin Kirchhoff

5.1 Introduction

Many applications in human language technology involve the use of a statistical language model—a model that specifies the a priori probability of a particular word sequence in the language of interest. Given an alphabet or inventory of units Σ and a sequence $W = w_1 w_2 \dots w_t \in \Sigma^*$, a language model can be used to compute the probability of W based on parameters previously estimated from a training set. Most commonly, the inventory Σ (also called **vocabulary**) is the list of unique words encountered in the training data; however, as we will see in this chapter, selecting the units over which a language model should be defined can be a rather difficult problem, particularly in languages other than English.

A language model is usually combined with some other model or models that hypothesize possible word sequences. In speech recognition, a speech recognizer combines acoustic model scores (and possibly other scores, such as pronunciation model scores) with language model scores to decode spoken word sequences from an acoustic signal. In machine translation, a language model is used to score translation hypotheses generated by a translation model. Language models have also become a standard tool in information retrieval [1], authorship identification [2], and document classification [3]. In several related fields, language models are used that are defined not over words but over acoustic units or isolated text characters. One of the core approaches to language identification, for example, relies on language models over phones or phonemes [4]; in optical character recognition, language models predicting character sequences are used [5, 6]. In this chapter, however, the focus is on language models over natural language words or wordlike units, which we define for now as units delimited by whitespace. We first present the fundamental n-gram modeling approach to statistical language modeling, as well as a range of more advanced modeling techniques, before discussing problems arising from the characteristics of particular languages, such as morphologically rich languages or languages without explicit word segmentation. The chapter concludes with a presentation of multilingual and crosslingual language modeling approaches.

5.2 *n*-Gram Models

The probability of a word sequence W of nontrivial length cannot be computed directly because unrestricted natural language permits an infinite number of word sequences of variable lengths. The probability $P(W)$ can be decomposed into a product of component probabilities according to the chain rule of probability:

$$P(W) = P(w_1 \dots w_t) = P(w_1) \prod_{i=1}^{t} P(w_i | w_{i-1} w_{i-2} \dots w_2 w_1) \tag{5.1}$$

Because the individual terms in this product are still too difficult to be computed directly, statistical language models make use of the **n-gram approximation**, which is why they are also called n-gram models. The assumption is that all previous words except for the $n-1$ words directly preceding the current word are irrelevant for predicting the current word, or, alternatively, that they are equivalent. Given this assumption of "history equivalence classes," the n-gram model is defined as:

$$P(W) \approx \prod_{i=1}^{t} P(w_i | w_{i-1}, \dots w_{i-n+1}) \tag{5.2}$$

Depending on the length of n, we can distinguish between unigrams ($n=1$), bigrams ($n=2$), trigrams ($n=3$), or 4-grams, 5-grams, and so on. An n-gram model is also often called an $(n-1)$-th order Markov model, because the approximation in Equation 5.2 embodies the Markov assumption of the independence of the current word given all other words except the $n-1$ preceding words.

5.3 Language Model Evaluation

Before describing parameter estimation methods and further refinements of the basic n-gram modeling approach, let us consider the problem of judging the performance of a language model. According to the basic definition given previously, a language model computes the probability of a word sequence W. How can we tell whether a language model is successful at estimating word sequence probabilities? Typically, two criteria are used: **coverage rate** and **perplexity** on a held-out test set that does not form part of the training data. The coverage rate measures the percentage of n-grams in the test set that are represented in the language model. A special case of this is the out-of-vocabulary rate (or OOV rate), which is 100 minus the unigram coverage rate, or, in other words, the percentage of unique word types *not* covered by the language model. The second criterion, perplexity, is an information-theoretic measure. Given a model p of a discrete probability distribution, perplexity can be defined as 2 raised to the entropy of p:

$$PPL(p) = 2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)} \tag{5.3}$$

In language modeling, we are often more interested in the performance of a language model q on a test set of a fixed size, say t words ($w_1 w_2 \dots w_t$). Then, language model perplexity can be computed as

$$PPL(p,q) = 2^{H(p,q)} = 2^{-\sum_{i=1}^{t} p(w_i) log_2 q(w_i)} \tag{5.4}$$

or simply

$$2^{-\frac{1}{t}\sum_{i=1}^{t} log_2 q(w_i)} \tag{5.5}$$

where $q(w_i)$ computes the probability of the $i'th$ word. If $q(w_i)$ is an n-gram probability, the equation becomes

$$2^{-\frac{1}{t}\sum_{i=1}^{t} log_2 p(w_i | w_{i-1}, \dots, w_{i-n+1})} \tag{5.6}$$

When comparing different language models, especially models based on different ways of decomposing a text into language modeling units (e.g., words vs. morphemes), care must be taken to normalize their perplexities with respect to the same number of units in order to obtain a meaningful comparison.

Perplexity can be thought of as the average number of equally likely successor words when transitioning from one position in the word string to the next. If the model has no predictive power at all, perplexity is equal to the vocabulary size. A model achieving perfect prediction, by contrast, has a perplexity of one. The goal in language model development is most often to minimize the perplexity on a held-out data set representative of the domain of interest.

However, it should be noted that sometimes the goal of language modeling is not to predict word sequence probabilities but to distinguish between "good" and "bad" word sequence hypotheses generated by some frontend such as a machine translation system or a speech recognizer. In this case, the language model should assign maximally distinct scores to word sequences that are errorful, ungrammatical, or otherwise unacceptable, as opposed to those that are correct. Optimizing for minimum perplexity does not necessarily achieve this goal; we return to this point in Section 5.6.3.

5.4 Parameter Estimation

5.4.1 Maximum-Likelihood Estimation and Smoothing

The standard procedure in training n-gram models is to estimate n-gram probabilities using the maximum-likelihood criterion in combination with parameter smoothing. The maximum-likelihood estimate is obtained by simply computing relative frequencies:

$$P(w_i | w_{i-1}, w_{i-2}) = \frac{c(w_i, w_{i-1}, w_{i-2})}{c(w_{i-1}, w_{i-2})} \tag{5.7}$$

where $c(w_i, w_{i-1}, w_{i-2})$ is the count of the trigram $w_{i-2}w_{i-1}w_i$ in the training data. It is obvious that this method fails to assign nonzero probabilities to word sequences that have not been observed in the training data; on the other hand, the probability of sequences that were observed might be overestimated. The process of redistributing probability mass such that peaks in the n-gram probability distribution are flattened and zero estimates are floored to some small nonzero value is called **smoothing**. The most common smoothing technique is **backoff**. Backoff involves splitting n-grams into those whose counts in the training data fall below a predetermined threshold τ and those whose counts exceed the threshold. In the former case, the maximum-likelihood estimate of the n-gram probability is replaced with an estimate derived from the probability of the lower-order $(n-1)$-gram and a backoff weight. In the latter case, n-grams retain their maximum-likelihood estimates, discounted by a factor that redistributes probability mass to the lower-order distribution. Thus, the backed-off probability P_{BO} for w_i given w_{i-1}, w_{i-2} is computed as follows:

$$P_{BO}(w_i|w_{i-1}, w_{i-2}) = \begin{cases} d_c P(w_i|w_{i-1}, w_{i-2}) \text{ if } c > \tau \\ \alpha(w_{i-1}, w_{i-2}) P_{BO}(w_i|w_{i-1}) \text{ otherwise} \end{cases} \tag{5.8}$$

where c is the count of (w_i, w_{i-1}, w_{i-2}), and d_c is a discounting factor that is applied to the higher-order distribution. The normalization factor $\alpha(w_{i-1}, w_{i-2})$ ensures that the entire distribution sums to one and is computed as

$$\alpha(w_{i-1}, w_{i-2}) = \frac{1 - \sum_{w_i : c(w_i, w_{i-1}, w_{i-2}) > \tau} d_c P(w_i|w_{i-1}, w_{i-2})}{\sum_{w_i : c(w_i, w_{i-1}, w_{i-2}) \leq \tau} P_{BO}(w_i|w_{i-1})} \tag{5.9}$$

The way in which the discounting factor is computed determines the precise smoothing technique. Well-known techniques include Good-Turing, Witten-Bell, Kneser-Ney, and others; see the study by Chen and Goodman [7] for an in-depth description and comparison of various smoothing techniques. For example, in Kneser-Ney smoothing, a fixed discounting parameter D is applied to the raw n-gram counts before computing the probability estimates:

$$P_{KN}(w_i|w_{i-1}, w_{i-2}) = \begin{cases} \frac{max\{c(w_i, w_{i-1}, w_{i-2}) - D, 0\}}{\sum_{w_i} c(w_i, w_{i-1}, w_{i-2})} \text{ if } c > \tau \\ \alpha(w_{i-1}, w_i) P_{KN}(w_i|w_{i-1}) \text{ otherwise} \end{cases} \tag{5.10}$$

In modified Kneser-Ney smoothing, which is one of the most widely used techniques, different discounting factors D_1, D_2, D_{3+} are used for n-grams with exactly one, two, or three or more counts:

$$Y = \frac{n_1}{n_1 + 2 * n_2} \tag{5.11}$$

$$D_1 = 1 - 2Y\frac{n_2}{n_1} \tag{5.12}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2} \tag{5.13}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3} \tag{5.14}$$

where $n_1, n_2, \ldots$ are the counts of n-grams with one, two, ..., counts.

Another common way of smoothing language model estimates is linear model interpolation [8]. In linear interpolation, M models are combined by

$$P(w_i|w_{i-1}, w_{i-2}) = \sum_{m=1}^{M} \lambda_m P(w_i|h_m) \tag{5.15}$$

where λ is a model-specific weight. The component models may use different conditioning variables, such as histories of different lengths, or they may have been estimated from different data sets, such as a large set of general data or a smaller set of domain-specific data (see Section 5.5). The following constraints hold for the model weights: $0 \leq \lambda \leq 1$, and $\sum_m \lambda_m = 1$. Weights are estimated by maximizing the log-likelihood (minimizing the perplexity) on a held-out data set that is different from the training set for the component models (and also different from the final evaluation or test set). This is typically done using the expectation-maximization (EM) procedure [9].

5.4.2 Bayesian Parameter Estimation

Bayesian probability estimation is an alternative parameter estimation method whereby the set of parameters of a model is itself viewed as a random variable governed by a prior statistical distribution. Given a training sample S and a set of parameters θ, $P(\theta)$ denotes a prior distribution over different possible values of θ, and $P(\theta|S)$ is the posterior distribution, which can be expressed using Bayes's rule as

$$P(\theta|S) = \frac{P(S|\theta)P(\theta)}{P(S)} \tag{5.16}$$

In language modeling, the set of parameters is the vector of word probabilities, that is, $\theta = \langle P(w_1), \ldots, P(w_K) \rangle$ (where K is the vocabulary size) for a unigram model, or, more generally, $\theta = \langle P(w_1|h_1), \ldots, P(w_K|h_K) \rangle$ for an n-gram model with K n-grams and history h of a specified length. The training sample S is a sequence of words, $w_1 \ldots w_t$. We require a point estimate of θ given the constraints expressed by the prior distribution and the training sample. This can be done using either the maximum a posteriori (MAP) criterion or the Bayesian criterion. The former finds the value that maximizes the posterior probability given in Equation 5.16:

$$\theta^{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(\theta|S) = \underset{\theta \in \Theta}{\operatorname{argmax}} P(S|\theta)P(\theta) \tag{5.17}$$

where Θ is the space of all possible assignments for θ. The Bayesian criterion finds the *expected* value of θ given the sample S:

$$\theta^B = E[\theta|S] = \int_\Theta \theta P(\theta|S) d\theta \tag{5.18}$$

$$= \frac{\int_\Theta \theta P(S|\theta)P(\theta) d\theta}{\int_\Theta P(S|\theta)P(\theta) d\theta} \tag{5.19}$$

Under the assumption that the prior distribution is a uniform distribution, the MAP estimate of the probability for a given word w is equivalent to the maximum-likelihood

estimate, whereas the Bayesian estimate is equivalent to the maximum-likelihood estimate with Laplace smoothing:

$$\theta_w^B = \frac{c(w)+1}{\sum_w c(w)+K} \tag{5.20}$$

Different choices for the prior distribution lead to different estimation functions. The most commonly used prior distribution in language model is the Dirichlet distribution. The Dirichlet distribution is the conjugate prior to the multinomial distribution (i.e., prior and posterior distribution have the same functional form). It is defined as

$$p(\theta) = D(\alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_k^{\alpha_k - 1} \tag{5.21}$$

where Γ is the gamma function and $\alpha_1, \ldots, \alpha_K$ are the parameters of the Dirichlet distribution (or hyperparameters), which can also be thought of as counts derived from an a priori training sample. The MAP estimate under the Dirichlet prior is

$$\theta^{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}} \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_k^{n_k + \alpha_k - 1} \tag{5.22}$$

where n_k is the number of times word k occurs in the training sample. The result is another Dirichlet distribution, parameterized by $n_k + \alpha$. The MAP estimate of $P(\theta|W, \alpha)$ thus is equivalent to the maximum-likelihood estimate with add-m smoothing, where $m_k = \alpha_k - 1$; that is, pseudocounts of size $\alpha_k - 1$ are added to each word (or n-gram) count. Hyperparameters offer a convenient way of integrating different information sources into the language model estimation process. This approach has most successfully been used for language model adaptation (e.g., [10]), where the prior is computed from a large out-of-domain data set and the observed counts are computed from a small in-domain data set; see Section 5.5 for further details on Bayesian language model adaptation. Early approaches to building language models entirely based on Bayesian estimation [11] did not perform as well as standard n-gram models estimated with the techniques described in Section 5.4.1. However, with recent progress in Bayesian statistics, alternative models have been developed that yield results comparable to those of Kneser-Ney smoothed n-gram models. In particular, this includes models that assume a latent topic structure of documents and use Bayesian estimation techniques to model this structure. These models are discussed more fully in Section 5.6.8.

5.4.3 Large-Scale Language Models

Recently, there has been much interest in scaling language models to very large data sets. The amount of available monolingual data increases daily, and for many languages, models can now be built from sets as large as several billions or trillions of words. Scaling language models to data sets of this size requires modifications to the ways in which language

models are trained, stored, and integrated into real-world systems (e.g., speech recognition decoders). It also affects parameter estimation in that exact probability computations may no longer be feasible.

Several sites [12, 13] have proposed distributed approaches to large-scale language modeling. Their common feature is that the entire language model training data is subdivided into several partitions, and counts or probabilities derived from each partition are stored in separate physical locations (i.e., they are distributed over a cluster of independent compute nodes in a client-server architecture). During runtime, clients can request statistics from a set of data partitions through a language model server, thus producing (possibly interpolated) probability estimates on demand. The advantage of distributed language modeling is that it scales to very large amounts of data and large vocabulary sizes and allows new data to be added dynamically without having to recompute static model parameters. Desired parameters, such as the n-gram model order or the specific mixture of different data partitions, can be chosen and requested at runtime, which allows dynamic decoding approaches to be used. The drawback of distributed approaches, however, is the slow speed of networked queries.

In Brants et al. [13], a nonnormalized form of backoff is introduced that differs from standard backoff (Equation 5.8) in that it uses the raw relative frequency estimate instead of a discounted probability if the n-gram count exceeds the minimum threshold (in this case 0):

$$S(w_i|w_{i-1}, w_{i-2}) = \begin{cases} P(w_i|w_{i-1}, w_{i-2}) \text{ if } c > 0 \\ \alpha S(w_i|w_{i-1}) \text{ otherwise} \end{cases} \tag{5.23}$$

The α parameter is fixed for all contexts rather than being dependent on the lower-order n-gram, as in Equation 5.8. The result is no longer a normalized probability distribution but a set of unnormalized scores (denoted by S rather than P for probabilities) that are used in the same manner as standard probabilities. The advantage of this scheme is that unnormalized scores are easier to compute in a distributed framework because summing over all n-gram contexts (which are stored in different physical locations and are therefore expensive to query) is no longer required. Interestingly, the authors found that the model performs almost as well as a model trained with standard Kneser-Ney smoothing on large amounts of data.

An alternative possibility is to use large-scale distributed language models at a second-pass rescoring stage only, after first-pass hypotheses have been generated using a smaller language model [14, 15]. Yet another approach is to store large-scale language models in the working memory of a single machine but to use efficient though possibly lossy data structures. Talbot and Osborne [16] investigate the use of Bloom filters for this purpose. Under this approach, corpus statistics (n-gram frequency counts, context counts, etc.) are represented in a highly memory-efficient, randomized data structure (a Bloom filter) in a quantized fashion. If $c(w_1 \ldots w_n)$ is the count of n-gram $w_1 \ldots w_n$, the quantized count $q(w_1 \ldots w_n)$ is defined as

$$q(w_1 \ldots w_n) = 1 + \lfloor \log_b c(w_1 \ldots w_n) \rfloor \tag{5.24}$$

At test time, the necessary statistics are queried from the filter, and the true frequency is approximated by the expected count given the quantized count:

$$E[c(w_1 \dots w_n)|q(w_1 \dots w_n) = j] = \frac{b^{j-1} + b^j - 1}{2} \tag{5.25}$$

In this framework, frequencies will never be underestimated but may possibly be overestimated, although the probability of overestimation decreases exponentially with the size of the estimation error. The advantage of this method is that, although the raw frequency counts may be inaccurate, querying the data structure is fast, thus enabling the model to compute smoothed probabilities on the fly. In practice, it was found that the performance of a Bloom filter language model on a machine translation task was close to that of a language model based on exact parameter estimation while providing memory savings of a factor of 4 to 6 [16].

The overall trend in large-scale language modeling is to abandon exact parameter estimation of the type described in the previous section in favor of approximate techniques. This development is likely to continue, leading to more powerful and refined approximation techniques as the number and size of text data collections continue to increase.

5.5 Language Model Adaptation

It is often the case that the amount of language model training data is insufficient, particularly when porting a speech or language processing system to a new domain, topic, or language. For this reason, much effort has been invested in **language model adaptation**, that is, designing and tuning a language model such that it performs well on a new test set for which little equivalent training data is available.

The most commonly used adaptation method is that of mixture language models, or model interpolation. Usually, an in-domain language model is trained using a small amount of in-domain data, and a larger background or generic model is trained using a large amount of out-of-domain data. These models are then interpolated according to Equation 5.15, optimizing the interpolation weights on a small development set. Naturally, this approach generalizes to more than two models, and several variations of basic model interpolation have been developed.

One popular method is topic-dependent language model adaptation. Seymour and Rosenfeld [17] show how documents can first be clustered into a large number of different topics, and individual language models can be built for each topic cluster. The desired final model is then fine-tuned by choosing and interpolating a smaller number of topic-specific language models.

A form of dynamic self-adaptation of a language model is provided by **trigger models**. The idea is that, in accordance with the underlying topic of the text, certain word combinations are more likely than others to co-occur; some words are said to "trigger" others—for example, the words *stock* and *market* in a financial news text. For clustering words according to topic and using them as trigger pairs, latent semantic analysis (LSA) [18] and probabilistic latent semantic analysis (PLSA) [19] have also been been used [20, 21, 22]. LSA, originally

formulated for information retrieval, represents a collection of texts as a document-word co-occurrence matrix, where rows correspond to words, columns correspond to documents, and individual cells represent the (possibly weighted) frequency of the word in the document. Singular-value decomposition applied to this matrix maps words to a low-rank continuous vector space. The semantic similarity within this space can then be computed using, for example, the cosine similarity of the corresponding word vectors. A language model can be adapted dynamically as follows:

$$P(w_i|h_i, \widetilde{h}_i) = \frac{P(w_i|h_i)\rho(w_i, \widetilde{h}_i)}{Z(h_i, \widetilde{h}_i)} \tag{5.26}$$

Here, $\widetilde{h}_i$ represents the global document history in LSA space up to word i, and ρ represents the similarity function measuring the compatibility between the current word and the semantic history. The idea is that the probabilities of semantically similar words get boosted by a factor proportional to their similarity with the global document history. Trigger relationships can also be incorporated into a language model in the form of constraints within constraint-based modeling frameworks (such as the maximum entropy [MaxEnt] model discussed in Section 5.6.5 [23] or the discriminative language model discussed in Section 5.6.3 [24]).

PLSA extends the basic, nonprobabilistic LSA approach by assuming a more sophisticated latent class model to decompose the word-document co-occurrence matrix instead of using simple singular-value decomposition. Given a latent class c, the probability of each word-document co-occurrence (w, d) can be expressed as:

$$P(w, d) = \sum_c P(c)P(w|c)P(d|c) = P(d)\sum_c P(c|d)P(w|c) \tag{5.27}$$

However, a potential concern with PLSA is its tendency to overfit to the training data. A more recent form of topic-based clustering is latent Dirichlet allocation (LDA) [25], which can be interpreted as a regularized version of PLSA. Topic models based on LDA and its extensions are described in Section 5.6.8.

A further variant of the standard adaptation framework is **unsupervised adaptation**, which is primarily relevant for speech recognition applications. Rather than using written text or perfectly transcribed speech as adaptation data, it is possible to directly use the output of a speech recognizer instead [26]. Several studies (e.g., [27, 28]) have shown that this method achieves roughly half of the improvement obtained by using perfect transcriptions for adaptation.

Recently, it has become common to utilize the Internet as a resource for additional language model data. If available text data for a given topic, domain, or language is insufficient, the web can be queried and additional domain-related data can be retrieved. After preprocessing and possibly filtering the data, it is added to the pool of existing data, or a separate model is trained on the web data and is subsequently interpolated with an existing baseline language model. Several rapid adaptation methods based on this general procedure have been described [29, 30, 31, 32].

Finally, alternative probability estimation schemes for language model adaptation have also been investigated. One of these is **maximum a posteriori (MAP) adaptation** [10].

Here, the counts collected separately from the generic out-of-domain (OD) and the in-domain adaptation (ID) data are combined as follows:

$$P(w|h) = \frac{c_{OD}(w,h) \cdot \varepsilon c_{ID}(w,h)}{c_{OD}(h) \cdot \varepsilon c_{ID}(h)} \tag{5.28}$$

where h and w are the history and predicted word respectively, c_{OD} is the count obtained from the out-of-domain data, and c_{ID} is the count from the in-domain data. The ε parameter ranges between 0 and 1 and represents the weight assigned to the adaptation data: because the amount of out-of-domain data is usually going to outweigh the amount of available adaptation data, the contributions from both sets can be balanced by appropriately setting the ε parameter, which is done empirically. A comparison of MAP and mixture models [33] has shown that mixture models are less robust than MAP adaptation toward variability in the adaptation data.

Although much of the work on language model adaptation has been performed in the context of speech recognition, several studies have also been done on adapting language models for machine translation. In Eck, Vogel, and Waibel [34] and Zhao, Eck, and Vogel [35], queries constructed from first-pass translation hypotheses are used to select additional sentences from a large corpus of target language data, which are then used as additional training data. Models built from this data are interpolated with the baseline language model and are used to retranslate the source language input text. Additional techniques for adapting language models using crosslingual data are described in Section 5.8.2.

5.6 Types of Language Models

Although n-gram models are still the most widely used type of statistical language model by far, a range of alternative models have been developed that have shown additional benefits in practical applications, often when used in combination with n-gram models.

5.6.1 Class-Based Language Models

Class-based language models [36] are a simple way of addressing data sparsity in language modeling. Words are first clustered into classes, either by automatic means [37] or based on linguistic criteria, for example, using part-of-speech (POS) classes. The statistical model makes the assumption that words are conditionally independent of other words given the current word class. If c_i is the class of word w_i, a class-based bigram model can be defined as follows:

$$P(w_i|w_{i-1}) = \sum_{c_i,c_{i-1}} P(w_i|c_i)p(c_i|c_{i-1},w_{i-1})P(c_{i-1}|w_{i-1}) \tag{5.29}$$

$$= \sum_{c_i,c_{i-1}} P(w_i|c_i)P(c_i|c_{i-1})P(c_{i-1}|w_{i-1}) \tag{5.30}$$

under the assumption that c_i is independent of w_{i-1} given c_{i-1}. Usually, a class will contain more than one word, such that the model simplifies to

$$P(w_i|w_{i-1}) = P(w_i|c_i)P(c_i|c_{i-1}) \tag{5.31}$$

Goodman [38] compared this decomposition to the following model:

$$P(w_i|w_{i-1}) \approx P(w_i|c_i,c_{i-1})P(c_i|c_{i-1}) \tag{5.32}$$

where the current word is conditioned not only on the current word class but also on the preceding word classes. Experiments on the North American Business News Corpus (with the training size ranging between 100,000 and 284 million words), using 20,000 test sentences and a vocabulary of 58,000 words showed that the model in Equation 5.32 worked better unless the training data size was at the lower end. Class-based models have been successful in reducing perplexity as well as practical performance in a wide range of language processing systems; however, they typically need to be interpolated with a word-based language model.

5.6.2 Variable-Length Language Models

In standard language modeling, vocabulary units are defined by simple criteria, such as whitespace delimiters, and the prediction of the probability of the next word is based on an invariable fixed-length history (save for backoff). Several modifications to this basic approach have been developed that aim at redefining vocabulary units in a data-driven way, resulting in merged units composed out of a variable number of basic units. These approaches are termed **variable-length** n-gram models. The challenge in these models is to find the best segmentation of the word sequence $w_1 w_2 \ldots w_t$ into language modeling units in addition to estimating the language model probabilities. Deligne and Bimbot [39] model the segmentation as a hidden variable and use an ME procedure to find the best segmentation. A variable-length model of order 7 yielded slight improvements in perplexity compared to a standard word-based bigram model, but no application results were reported.

A simpler approach is to start with the initial whitespace-based segmentation suggested by the standard orthography of the language and to merge selected units, rather than attempting to rederive the entire vocabulary segmentation from scratch. A limited number of merged units corresponding to frequently observed short phrases can thus be added to the language model vocabulary. A common criterion for identifying potential phrasal candidate units is the mutual information between adjacent words (e.g., [40]). The actual selection of phrasal units is then made using a greedy iterative algorithm: in each iteration, those candidates are selected that reduce the perplexity of a development corpus the most [41, 42]. In Zitouni, Smaili, and Haton [42], word class information is used to identify candidate phrasal units in that mutual information is computed over pairs of classes rather than pairs of words, which was shown to reduce perplexity by roughly 10% compared to word-based selection of candidate pairs. This model also achieved an 18% relative reduction in word-error rate on a medium-scale French automatic speech recognition (ASR) task.

5.6.3 Discriminative Language Models

Standard n-gram models embody a generative model for assigning a probability to a given word sequence W. However, in practical applications like machine translation or speech recognition, the task of a language model is often to separate good sentence hypotheses from bad sentence hypotheses. For this reason, it would be desirable to train language model parameters discriminatively, such that word strings of widely differing quality receive maximally distinct probability estimates. Recent attempts at such **discriminative**

language modeling include Roark et al. [43], Collins, Saraçlar, and Roark [44], Shafran and Hall [45], and Arisoy et al. [46]. Here, the language model is applied to an existing set of competing sentence hypotheses $\mathcal{Y}$ generated for an input x (e.g., an acoustic sequence in the case of speech recognition, or the source language string in the case of machine translation) by some generation function $\text{GEN}(x)$. Arbitrary feature functions $\phi(x, y)$ can be defined jointly over the input and each output $y \in \mathcal{Y}$. These are then used in a global linear model that selects the best hypothesis:

$$F(x) = \underset{y \in \text{GEN}(x)}{\text{argmax}}\ \phi(x, y)\alpha \tag{5.33}$$

where α is a weight vector. In the most basic case, the feature functions are the raw n-gram counts obtained from the training data. However, additional feature functions representing statistics over word classes or subword units may be integrated (see §5.7.1). The parameter vector α can be trained by the perceptron algorithm [47] or a conditional log-linear model [43]. The perceptron algorithm, for example, iterates through all training samples (for several epochs) and selects the currently best-scoring hypothesis for each sample. If it differs from the true reference hypothesis, it updates the current weights by adding the counts of the features in the correct hypothesis and subtracting their counts in the chosen hypothesis. Such a training procedure directly minimizes the desired objective function, such as word-error rate in a speech recognition system. Thus, the weights with which the counts of different n-grams contribute to the final model are trained to optimize system performance rather than minimize the perplexity criterion described in Section 5.3. Roark, Saraçlar, and Collins [48] reported a 1.8% absolute improvement in word-error rate (from 39.2% to 37.4%) on a large-vocabulary speech recognition task for a one-pass system and a 0.9% reduction in word-error rate for a multipass recognizer. Recently, discriminative language modeling has also been applied to statistical machine translation [49], where it yielded an improvement of 1 to 2 BLEU points over a state-of-the-art baseline system. As we shall see, a discriminative language model also offers a convenient way of integrating additional linguistic information, such as morphological features.

5.6.4 Syntax-Based Language Models

A well-known drawback of n-gram language models is that they cannot take into account relevant words in the history that fall outside the limited window of the directly preceding $n-1$ words. However, natural language exhibits many types of **long-distance dependencies**, where the choice of the current word is dependent on words that are relatively far removed in terms of sentence position. In the following example, the plural noun *Investors* triggers the plural verb *were* but is not taken into account as a conditioning variable by an n-gram model, where n is usually no larger than 4 or 5.

Investors, *who still showed confidence in financial markets last week,* **were** *responsible for today's downturn.*

To address this problem, several approaches to syntax-based language modeling have been developed, whose goal is to explicitly model such syntactic relationships and use them to

estimate better probabilities. Most of these approaches use a statistical parser to construct the syntactic representation S of the sentence and define a probability model that incorporates S. Chelba and Jelinek's **structured language model** [50] computes the joint probability of a word sequence and its parse S, $P(W, S)$ and decomposes it into a product of component probabilities involving various elements of the the word sequence proper, the head words from the parse structure, and the POS tags in the parse structure. Results reported in [50] indicate that a structured language model interpolated with a trigram model achieved a relative reduction in perplexity of 8% on the Wall Street Journal Continuous Speech Recognition (CSR) and Switchboard corpora. When used for lattice rescoring in a speech recognition system, it yielded a 6% relative reduction on the Wall Street Journal corpus and a 0.5% absolute reduction (41.1% to 40.6%) on Switchboard.

Another syntax-based model is the "almost-parsing" language model by Wang and Harper [51] (also called SuperARV model), which is based on a constraint-dependency grammar. Here, sentences are annotated with so-called SuperARVs, which are rich tags combining lexical features and syntactic information pertaining to a word (entry in the lexicon). A joint language model (the SuperARV language model) is defined over the word sequence and the sequence of tags as follows:

$$P(w_1, \ldots, w_N, t_1, \ldots, t_N) = \prod_{i=1}^{N} P(w_i t_i | w_1 \ldots, w_{i-1}, t_1 \ldots t_{i-1}) \tag{5.34}$$

$$= \prod_{i=1}^{N} P(t_i | w_1 \ldots, w_{i-1}) P(w_i | w_1 \ldots w_{i-1}, t_1 \ldots t_{i-1}) \tag{5.35}$$

$$\approx \prod_{i=1}^{N} P(t_i | w_{i-2}, w_{i-1}, t_{i-1}, t_{i-1}) P(w_i | w_{i-2}, w_{i-1}, t_{i-2}, t_{i-1}) \tag{5.36}$$

The model is smoothed by recursive linear interpolation of higher-order and lower-order models. The SuperARV model was evaluated on the Wall Street Journal Penn Treebank and CSR tasks and was compared to other parser-based language models, including the structured language model described previously, as well as to standard trigram and POS-based models. It was found that the SuperARV achieved the lowest perplexity scores out of all. When used for lattice rescoring on the CSR task, the SuperARV model yielded relative word-error rate reductions between 3.1% and 13.5% and again outperformed all other models.

5.6.5 MaxEnt Language Models

One shortcoming of maximum-likelihood-based probability estimation for language models is that the constraints imposed by estimates solely derived from the training data are too strong. MaxEnt models represent an alternative where these constraints are relaxed. Rather than setting the probability of a given n-gram to its relative frequency in the training data (modulo smoothing), the requirement of MaxEnt modeling is that the model agree,

on average, with the observed counts of events in the training data. The MaxEnt model is formulated as follows:

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_k \lambda_k f_k(x,y)\right) \tag{5.37}$$

where $f(x,y)$ is a feature function defined both on the input and the predicted variables, λ is a feature function specific weight, and $Z(x)$ is a normalization factor, computed as

$$Z(x) = \sum_{y \in Y} \exp\left(\sum_k \lambda_k f_k(x,y)\right) \tag{5.38}$$

Once appropriate feature functions have been defined, the expected value of f_k is

$$E(f_k) = \sum_{x \in X, y \in Y} \tilde{p}(x) p(y|x) f_k(x,y) \tag{5.39}$$

where $\tilde{p}(x)$ is the empirical distribution of x in the training data. The empirical expectation of f_k (derived from the training data) is

$$\tilde{E}(f_k) = \sum_{x \in X, y \in Y} \tilde{p}(x,y) f_k(x,y) \tag{5.40}$$

The model is then trained such that expected values match empirical expected values

$$E(f_k) = \tilde{E}(f_k) \qquad \forall k \tag{5.41}$$

while simultaneously maximizing the entropy of the distribution $p(y|x)$. This is equivalent to maximizing the conditional log-likelihood of the training data.

The MaxEnt framework was first applied to language modeling by Rosenfeld [52]. In the context of language modeling, y represents the predicted word and x the history or, more generally, the conditioning variables used for prediction. Note that in this case, a much larger context than the $n-1$ directly preceding words may be included: feature functions may be defined over the entire sentence [53] or over an even larger domain. Most commonly, however, feature functions are simply defined over n-grams; for example, for a given word w_i and history h_i, a bigram feature function would be defined as follows:

$$f_{w_1,w_2}(h_i, w_i) = \begin{cases} 1 & \text{if } h_i \text{ ends in } w_1 \text{ and } w_i = w_2 \\ 0 & \text{otherwise} \end{cases} \tag{5.42}$$

The model can be trained using iterative methods, such as generalized iterative scaling [54] or improved iterative scaling [55], or by the faster quasi-Newton approaches (see [56]). Nevertheless, training of MaxEnt language models can be computationally demanding: in principle, the normalization factor in Equation 5.38 needs to be computed for all different values of x, and computing the feature expectations requires summation over all (x,y) pairs for which the feature is defined. Wu and Khudanpur [57] presented efficient training methods that result in considerable speedups during training. First the vocabulary is partitioned according to whether words are subject to marginal or conditional constraints, and

the summation required for the normalization factor is computed separately for both sets. Second, a hierarchical normalization computation procedure is proposed where partial sums (e.g., for histories ending in the same suffix) are reused. Together, these modifications led to speedups ranging between 15x and 30x.

Another potential problem of MaxEnt models is that they are prone to overfitting, especially when a large number of feature functions is used in relation to the number of samples. Possible solutions to this problem are feature selection [58], regularization [59], or incorporating priors on the feature functions. Using a Gaussian prior was proposed by Chen and Rosenfeld [59], for example. Rather than simply maximizing the conditional log-likelihood of the training data,

$$\underset{\Lambda}{\text{argmax}} \sum_{i=1}^{M} log P_\Lambda(y_i|x_i) \tag{5.43}$$

this approach maximizes the conditional log-likelihood modulo a penalty term composed of a product of zero-mean Gaussians for all feature functions:

$$\underset{\Lambda}{\text{argmax}} \sum_{i=1}^{M} \log P_\Lambda(y_i|x_i) \times \prod_{k=1}^{K} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(\frac{\lambda^2}{2\sigma_k^2}\right) \tag{5.44}$$

where σ_k^2 is the variance of the k^{th} Gaussian. Goodman [60] suggests exponential priors as an alternative, which can sometimes yield better performance.

Wu and Khudanpur [61] report on a MaxEnt model for speech recognition on the Switchboard task that incorporates topic constraints. The model achieved a 7% perplexity reduction and an absolute word-error rate reduction of 0.7% (38.5% to 37.8%). Integration of syntactic constraints independently yielded a 7% perplexity reduction and a 0.8% reduction in word-error rate. The combination of both types of constraints was shown to be additive, yielding a 12% relative perplexity reduction and an absolute word-error rate improvement of 1.3%.

5.6.6 Factored Language Models

The factored language model (FLM) approach [62, 63] builds on the observations that the prediction of words is dependent on the surface form of the preceding words and that better generalizations can be made when additional information, such as the word's POS or morphological class, is taken into account. In particular, word n-gram counts might be insufficient to robustly estimate the probability of word w_i given word w_{i-1}, but if we know that word w_{i-1} is of a particular class, say a determiner, we might be able to obtain a good probability estimate for $P(w_i|\text{determiner})$. This is reminiscent of the class-based models described previously; however, in an FLM, many such class-based estimates are combined and structured hierarchically through the use of a **generalized backoff** strategy. FLMs assume a factored word representation, where words are considered feature vectors rather than individual surface forms; that is, $W \equiv f_{1:K}$. An example is shown here:

WORD:	*Stock*	*prices*	*are*	*rising*
STEM:	Stock	price	be	rise
TAG:	Nsg	N3pl	V3pl	Vpart

The word surface form itself can be one of the features. A statistical model over this representation can be defined as follows (using a trigram approximation):

$$p\left(f_t^{1:K}, f_2^{1:K}, \ldots, f_t^{1:K}\right) \approx \prod_{i=3}^{t} p\left(f_t^{1:K} | f_{i-1}^{1:K}, \ldots, f_{i-2}^{1:K}\right) \quad (5.45)$$

Thus, each word is dependent not only on a single stream of temporally ordered word variables but also on additional parallel (i.e., simultaneously occurring) feature variables.

In the definition of standard backoff (Equation 5.8), the model backs off from the higher-order to the next lower-order distribution. In an FLM, however, it is not immediately obvious how backoff should proceed because conditioning variables are not only temporally ordered but also occur in parallel. Thus, a decision must be made as to which subset of features the model should back off to in which order. In principle, there are several different ways of choosing among different backoff "paths":

1. Choose a fixed, predetermined backoff path based on linguistic knowledge (e.g., always drop syntactic before morphological variables).
2. Choose the path at runtime based on statistical criteria.
3. Choose multiple paths and combine their probability estimates.

The last option, called **parallel backoff**, is implemented via a new, generalized backoff function (here shown for a trigram):

$$P_{GBO}(f|f_1, f_2) = \begin{cases} d_c P_{ML}(f|f_1, f_2) \text{ if } c > \tau \\ \alpha(f_1, f_2) g(f, f_1, f_2) \text{ otherwise} \end{cases} \quad (5.46)$$

where, similar to Equation 5.8, c is the count of (f, f_1, f_2), $P_{ML}(f|f_1, f_2)$ is the maximum likelihood estimate, τ is the count threshold, and $\alpha(f_1, f_2)$ is the normalization factor that ensures that the resulting scores form a probability distribution. The function $g(f, f_1, f_2)$ determines the backoff strategy. In a typical backoff procedure, $g(f, f_1, f_2)$ equals $P_{BO}(f|f_1)$. In generalized parallel backoff, however, g can be any nonnegative function of f, f_1, f_2 and can be instantiated to, for example, the mean, weighted mean, product, or maximum functions. For example, the mean function would take the average of the individual estimates:

$$g_{\text{mean}}(f, f_1, f2) = 0.5 P_{BO}(f|f_1) + 0.5 P_{BO}(f|f_2) \quad (5.47)$$

In addition to different choices for g, different discounting parameters can be chosen at different levels in the backoff graph.

It is not a priori obvious which backoff strategy works best; the optimal strategy is highly dependent on the particular language modeling task. Because the space of possible factored language model structures and backoff parameters is very large, it is advisable to use an automatic, data-driven procedure to find the best settings. An automatic FLM optimization procedure based on genetic algorithms was proposed by Duh and Kirchhoff [64].

FLMs have been implemented as an add-on to the widely used SRILM (Stanford Research Institute Language Modeling) toolkit [65] and have been used successfully for morpheme-based language modeling [62], multispeaker language modeling [66], dialog act tagging [67], and speech recognition [68, 63], especially in sparse data scenarios such as highly inflecting languages (see also §5.7.2).

5.6.7 Other Tree-Based Language Models

Several other language modeling approaches make use of tree structures; one, for example, is the hierarchical class-based backoff model proposed by Zitouni [69]. Here, backoff proceeds along a hierarchy of word classes arranged in a tree structure, with more general classes at the top and more specific classes at the bottom. Backoff proceeds along the class hierarchy from bottom to top; that is, more specific backoff classes are utilized before more general ones. The main differences from FLMs are that the backoff path is fixed and predefined in advance, whereas FLMs permit the combination of probability estimates from different paths in the backoff graph, as well as the dynamic choice of a path at runtime. Zitouni found that a hierarchical class-based language model yields gains primarily when the test set contains a large number of unseen events: on a speech recognition language modeling task with a 5,000-word vocabulary, the unseen word perplexity was reduced by 10%, whereas on a task with a 20,000-word vocabulary, it was reduced by 26% and the word-error rate decreased by 12%. Some extensions to this hierarchical class n-gram language model were proposed by Wang and Vergyri [70]. Specifically, POS information was integrated into the word clustering process, and separate hierarchical class trees were defined for different POS categories. On an Egyptian Colloquial Arabic speech recognition task (a test set of 18,000 words), the model achieved 8% relative improvement in perplexity over a standard n-gram model and a 3% relative improvement over the model described by Zitouni [69].

Random forest language models (RFLMs), proposed by Xu and Jelinek [71], model all word histories seen in the training data as a collection of randomly grown decision trees (a random forest). Nodes in the decision trees are associated with sets of histories, with the root node containing all histories. Trees are grown by dividing the set of histories into two subsets according to the word identity at a particular position in the history. Out of a number of possible splits, the split that maximizes the log-likelihood of the training data is chosen. Two steps are taken to introduce randomness into this process: first, the set of histories from a parent node are initially assigned randomly to the two subnodes; second, the set of splits chosen to undergo the log-likelihood test is selected randomly as well. After the growing process has stopped, each leaf node in a decision tree can be viewed as a cluster of similar word histories forming an equivalence class. Rather than defining equivalence classes deterministically by the most recent $n-1$ words (as done by conventional n-gram models), equivalence classes are now defined based on the set membership of the words in the history.

The decision tree–growing procedure is run multiple times, and the decision trees resulting from each run are added to the random forest. Supposing that M decision trees have been obtained, the RFLM probabilities are computed as averages of the individual decision tree probabilities:

$$P_{RF}(w_i|w_{i-n+1}, \ldots w_{i-1}) = \frac{1}{M} \sum_{j=1}^{M} P_{DT_j}(w_i|\phi_{DT_j}(w_{i-n+1}, \ldots, w_{i-1})) \quad (5.48)$$

Here, ϕ_{DT_j} is the $j'th$ is a function mapping the history $w_{i-1+n+1}, \ldots, w_{i-1}$ to a leaf node in the j'th decision tree. The number of decision trees M usually ranges in the dozens or hundreds. Perplexity and word-error rate tests on the Penn Treebank portion of the Wall Street Journal corpus showed that the perplexity of a random forest trigram was reduced by 10.6% relative to a trigram with interpolated Kneser-Ney smoothing. Interpolation of

the RFLM with a Kneser-Ney model did not improve perplexity any further. N-best list rescoring with RFLMs yielded a relative word-error rate improvement of 11% on the Wall Street Journal DARPA'93 HUB1 benchmark task. Since their inception, RFLMs have been applied to structured language modeling [71] and prosodic modeling [72]. In the context of multilingual language modeling, they have also been applied to morphologically rich languages (see §5.7.1).

5.6.8 Bayesian Topic-Based Language Models

A significant recent trend in statistical language modeling is Bayesian modeling of latent topic structure in documents. One of the first models of this type was the latent Dirichlet allocation model proposed by Blei, Ng, and Jordan [25]. The LDA model assumes that a document is composed of K topics, denoted as $z_1, \ldots, z_K$. Each topic generates words according to a topic-specific distribution over individual words (i.e., a topic is modeled as a bag of words; n-grams are not taken into account). The probability vector over words is denoted by $\boldsymbol{\phi_k}$ for each topic $k = 1, \ldots, K$. Each topic has a prior probability, denoted by θ_k. The topic priors $\theta_1, \theta_2, \ldots, \theta_K$ are themselves distributed according to the Dirichlet distribution with hyperparameters $\alpha_1, \ldots, \alpha_K$ (see §5.4.2 for an explanation of the Dirichlet distribution):

$$P(\theta_1, \ldots, \theta_k) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{5.49}$$

The generative model underlying this approach is that a set of priors $\theta_1, \ldots, \theta_K$ is sampled from the Dirichlet distribution. A given topic z_k is chosen with probability θ_k, then a word w is generated from the topic with probability $\phi_k(w)$. The probability of an entire document consisting of a sequence W of t words is modeled as

$$p(W|\alpha, \phi) = \int p(\theta|\alpha) \left(\prod_{i=1}^{t} \sum_{z_i} p(z_i|\theta) p(w_i|z_i, \phi) \right) d\theta \tag{5.50}$$

The challenging aspect of LDA is computing the posterior distribution of the latent variables θ and $\mathbf{z}$, $p(\theta, \mathbf{z}|W, \alpha, \phi)$, which is not amenable to exact inference. Sampling techniques such as Markov chain Monte Carlo (e.g., [73]) or variational inference [25] need to be used.

Because the LDA model is a unigram model, it also needs to be combined with an n-gram model for practical purposes. Wang et al. [74] combined LDA with a trigram and a probabilistic context-free grammar, yielding perplexity reductions of 9% to 23% on the Wall Street Journal corpus relative to a Kneser-Ney smoothed trigram model. Hsu and Glass [75] used LDA combined with a hidden Markov model for a spoken lecture recognition task. A combination of language models trained with the topic labels provided by this model yielded a 16.1% perplexity reduction and a 2.4% word-error rate reduction over an already adapted trigram model.

The LDA model has been extended in various ways: first, LDA can be generalized to utilize Dirichlet processes [76], a prior for nonparametric models that can handle an infinite number of topics. Thus, rather than assuming a fixed set of K topics, the number can be adjusted on the basis of training data properties. Second, the latent topic variables can

be structured hierarchically: each topic can have a number of subtopics, and individual topics can be shared between different data clusters. This is modeled by a hierarchical Dirichlet process (HDP) [77]. Huang and Renals exploited HDPs for integrating topic and participant role into language models for meeting-style conversational speech recognition. HDP-adapted language models yielded slight word-error-rate reductions (0.3% absolute) compared to standard adaptation methods, for baseline systems ranging around 39% word-error rate. Teh [78] reports on a Bayesian language model based on a Pitman-Yor process that by itself achieves perplexities comparable to those of a Kneser-Ney smoothed trigram model (without requiring interpolation with the baseline model).

5.6.9 Neural Network Language Models

With the exception of LSA-based language models, all of the language modeling approaches described previously estimate probabilities for events in a discrete space. Neural network language models (NNLMs) [79] take a different approach: discrete word sequences are first mapped into a continuous representation, and n-gram probabilities are then estimated within this continuous space. The assumption is that words having similar distributional properties will receive similar continuous representations, which will in turn result in smoother probability estimates.

The neural network is typically a multilayer perceptron with an input layer, a projection layer, a hidden layer, and an output layer of nodes. A graphical representation of the architecture of an NNLM is shown in Figure 5–1. Adjacent layers are fully interconnected by weights. For a vocabulary of V words, the input consists of a concatenation of $n-1$ V-dimensional binary feature vectors representing the history of $n-1$ words (e.g., the first two words in a trigram). The projection layer i of a given fixed dimensionality d encodes the shared continuous representation of the words, which is learned during training. The hidden layer h also has a fixed number of J nodes, each of which computes a thresholded, nonlinear combination of incoming activations, for example, using the tangent function:

$$h_j = \tanh\left(\sum_{k=1}^{d} w_{jk}^h i_k + b_j^h \right) \qquad \forall j, i = 1, \ldots, J \tag{5.51}$$

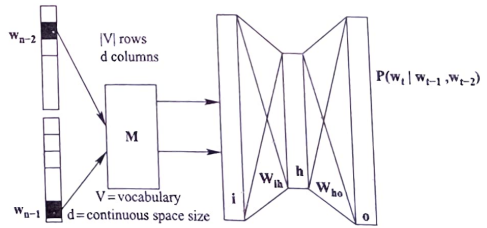

Figure 5–1: Neural network language model

where w^h denotes the weights connecting the projection, and the hidden layers and b^h are the biases on the hidden layer nodes. Finally, the output layer o computes a posterior probability distribution over V by

$$o_j = \frac{a_j}{\sum_{k=1}^{V} a_k} \qquad \forall j, j = 1, \ldots, V \tag{5.52}$$

and

$$a_j = \sum_{k=1}^{J} w_{jk}^o h_k + b_j^o \tag{5.53}$$

During training, binary target labels are associated with the output layer, 1 for the predicted word and 0 for all other words. The network is trained (e.g., using backpropagation) to maximize the log-likelihood of the training data, possibly enriched with a regularization term R to constrain the parameter values θ:

$$L = \frac{1}{T} \sum_{i=1}^{T} log(P(w_i|h_i); \theta) - R(\theta) \tag{5.54}$$

The regularization term can take various forms; a common approach is to use the sum of the squared weights: $\sum_w w^2$. This limits the complexity of the network and reduces the chance of overfitting by preventing the weights from becoming too large. Thus, a particular word in the history is first projected into a continuous representation shared among all words, which is then used as the basis for estimating the probability of the predicted word given its history.

NNLMs have been successfully used for speech recognition by Schwenk and Gauvain [80], Schwenk [81], and Schwenk, Déchelotte, and Gauvain [82], for example, and for machine translation by Alexandrescu and Kirchhoff [83]. Although they typically use a truncated vocabulary consisting of the m most frequent words only, they have yielded significant improvements when used in combination with standard n-gram models. Schwenk [81] reported an 8% relative reduction in perplexity and a 0.5% absolute improvement in word-error rate on a French broadcast news recognition task. Emami and Mangu [82] obtained a 0.8% absolute (3.8% relative) improvement in word-error rate on an Arabic speech recognition task.

Emami and Jelinek [84] used neural network–based probability estimation in combination with a structured language model; Alexandrescu and Kirchhoff [85] combined NNLMs with a factored word representation for Arabic to be able to exploit word similarities derived not only from distributional properties but from morphological and POS classes.

5.7 Language-Specific Modeling Problems

The majority of language modeling research has focused on the English language. However, speech and language processing technology has been ported to a range of other languages, some of which have highlighted problems with the standard n-gram modeling approach and

have necessitated modifications to the traditional language modeling framework. In this section, we look at three types of language-specific problems: morphological complexity, lack of word segmentation, and spoken versus written languages.

5.7.1 Language Modeling for Morphologically Rich Languages

A morphologically rich language is characterized by a large number of different unique word forms (types) in relation to the number of word tokens in a text that is due to productive morphological (word-formation) processes in the language. A morpheme is the smallest meaning-bearing unit in a language. Morphemes can be either free (i.e., they can occur on their own), or they are bound (i.e., they must be combined with some other morpheme). Morphological processes include compounding (forming a new word out of two independently existing free morphemes), derivation (combination of a free morpheme and a bound morpheme to form a new word), and inflection (combination of a free and a bound morpheme to signal a particular grammatical feature).

Germanic languages, for example, are notorious for their high degree of compounding, especially for nominals. Turkish, an agglutinative language, combines several morphemes into a single word; thus, the same material that would be expressed as a syntactic phrase in English can be found as a single whitespace-delimited unit in a Turkish sentence, such as: *görülmemeliydik* = 'we should not have been seen'.

As a result, Turkish has a huge number of possible words. Many languages have rich inflectional paradigms. In languages like Finnish and Arabic, a root (base form) may have thousands of different morphological realizations. Table 5–1 shows two Modern Standard Arabic (MSA) inflectional paradigms, one for present tense verbal inflections for the root *skn* (basic meaning: 'live'), one for pronominal possessive inflections for the root *ktb* (basic meaning 'book').

Morphological complexity causes problems for language modeling due to the high ratio of types to tokens, which results in a lack of training data: many n-grams in the test data are not seen at all in the training data or are not observed frequently enough, leading to unreliable probability estimates. Another effect is a high OOV rate. To some extent, this effect can be mitigated by simply collecting more training data; however, depending on the

Table 5–1: MSA inflectional paradigms for present tense verb forms and possessive pronouns (affixes are separated from the word stem by hyphens)

Word	Meaning	Word	Meaning
'a-skun(u)	I live	kitaab-iy	my book
ta-skun(u)	you (MASC) live	kitaabu-ka	your (MASC) book
ta-skun-iyna	you (FEM) live	kitaabu-ki	your (FEM) book
ya-skun(u)	he lives	kitaabu-hu	his book
ta-skun(u)	she lives	kitaabu-haa	her book
na-skun(u)	we live	kitaaabu-na	our book
ta-skun-uwna	you (MASC-PL) live	kitaabu-kum	your book
ya-skun-uwna	they live	kitaabu-hum	their book

Table 5–2: Number of word tokens, word types, and OOV rates for different languages in non-decomposed form

Language	Style	# Tokens	# Types	OOV Rate at (N) Words	Source
English	news text	19M	105k	1% (60k)	[86]
Arabic	news text	19M	690k	11% (60k)	[86]
Czech	news text	16M	415k	8% (60k)	[87]
Korean	news text	15.5M	1.5M	25% (100k)	[88]
Turkish	mixed text	9M	460k	12% (460k)	[89]
Finnish	news text, books	150M	4M	1.5% (4M)	[90]

morphological complexity of the language, the vocabulary growth does not slow down as sharply as for morphologically poor languages as more and more running text is taken into account. Table 5–2 shows examples of the relationship between word types and word tokens for different languages, as well as typical OOV rates for different languages on held-out test sets.

When processing morphologically rich languages, it must be determined whether the vocabulary needed for a given application can be expressed as a list of full word forms (e.g., the most frequent forms occurring in the training data) or whether smaller subword units need to be chosen as basic language modeling units. The choice depends on the constraints imposed by available computing resources, such as the efficiency of the decoder in a speech recognition application, memory, and desired speed, as well as on the amount of training data. The advantage of decomposing words into smaller units lies in the reduction of the vocabulary size, which in turn reduces the number of distinct n-grams. In addition to improving speed and reducing memory consumption, subword units occur in multiple words; therefore, training data can be shared across words, and the number of training tokens per unit increases, which leads to more robust probability estimation. Finally, modeling based on subword units might also allow the language model to assign probabilities to words that were not seen in the training data. On the other hand, if a word is decomposed linearly, a fixed n-gram context provides only information about the relationship between different parts of a word but not about interword dependencies. Thus, the predictive power of the language model is diminished. Moreover, when the language modeling vocabulary is identical to the vocabulary used in a speech recognizer, care must be taken to define units that do not become too short and hence acoustically confusable.

Arabic, generally recognized as a morphologically rich language, is an interesting example highlighting different situations where decomposition may or may not be required. Several studies [68, 63, 91] have shown that integrating morphological information into the language model is helpful for modeling dialectal Arabic. Although the dialects of Arabic exhibit morphological simplifications compared to MSA (the written standard), they have very sparse training data because they are essentially spoken languages and need to be transcribed manually to obtain language modeling data. Large amounts of data are available for MSA, and significant improvements through morphological decomposition in the language model have not been observed for this variety when large training sets were used [91]. Moreover, the vocabulary size needed for large-scale applications such as speech recognition of MSA

is around 600,000 to 800,000 words, which is within the range that current decoders can accommodate [92].

It is obvious, however, that for languages with particularly high type to token ratios (e.g., Finnish or Turkish), some form of decomposition is required: for large tasks, the size of the vocabulary needed to achieve sufficient coverage of the test data is likely to exceed current decoder capabilities, and the corresponding language model would not be able to be estimated reliably. In the following section, we discuss several recent approaches to the problem of word decomposition.

5.7.2 Selection of Subword Units

The identification of subword units can be performed in a data-driven, unsupervised way; it can be based on linguistic information (e.g., a morphological analyzer); or it can be a combination of both. Linguistically based methods typically involve a handcrafted morphological analysis tool, such as the Buckwalter Morphological Analyzer developed for Arabic [93], which analyzes each word form into its morphological components. In the case where several possible analyses are provided for each word form, statistical disambiguation needs to be performed in a subsequent step (e.g., [94]). Data-driven approaches may incorporate varying degrees of information about the specific language in question, and they may vary with respect to the optimization criterion. Some approaches are intended to discover units corresponding to linguistically defined morphemes, whereas others are aimed at selecting an inventory of units that is best suited to the task or application at hand.

Automatic algorithms for identifying linguistic morphemes are as old as Zellig Harris's 1955 approach of estimating the perplexity of different letters following each letter in a word [95]. If the perplexity at a certain transition is high (i.e., the following letters are not easily predictable) a morpheme boundary can be hypothesized at that point. Adda-Decker and Lamel [96] show how a modified version of this approach can be used to decompose German compounds, thus reducing the OOV rate by a relative amount of 23% to 50%, in a German corpus of 300 million words and a fixed vocabulary ranging from 65,000 to 100,000.

In general, simple frequency-based approaches are prone to overfitting to the training data and hypothesizing more morphemes than desired, because the fit to the data is not balanced against the total size of the inventory. This shortcoming is addressed by models that include explicit penalty terms for the size of the morpheme inventory. The more recently developed Morfessor package [97] is one such example. It attempts to derive a morpheme inventory M from a corpus C by maximizing its posterior probability:

$$M = \underset{M}{\operatorname{argmax}}\, P(M|C) = P(C|M)P(M) \tag{5.55}$$

which is equivalent to a minimum description length approach. The search over possible morphemes proceeds by way of a greedy algorithm that recursively splits each word into two subparts, trying out all possible positions. Those splits that improve the probability $P(M|C)$ (reduce the code length) are chosen. Later versions of this approach [98, 99] include a stochastic morphological category model and different probability estimation techniques. Morfessor models outperformed most other automatic segmentation algorithms in a benchmark evaluations task involving Finnish, Turkish, and English [100]. Language models using Morfessor to decompose words have been applied to Finnish, Estonian, Turkish, and Arabic

speech recognition and achieved good results on the first three languages, which are highly agglutinative [90].

An alternative to trying to match a predefined linguistic inventory is to derive a unit inventory that directly optimizes a criterion related to language model performance, such as perplexity or OOV rate. This may be preferable in cases where languages are not strictly agglutinative but contain some amount of fusion, such as nontransparent changes in the word form produced by the combination of two or more morphemes. This approach has been pursued by Whittaker and Woodland [101], for example, for Russian language modeling. Here, a particle-based model was developed, defined as

$$P(w_i|h) = \frac{1}{Z(h)} P\left(u^{w_i}_{L(w_i)}|u^{w_i}_{L(w_i)-1}\right) P\left(u^{w_i}_{L(w_i)-1}|u^{w_i}_{L(w_i)-2}\right), \ldots, P(u^{w_i}_1)\left|u^{w_{i-1}}_{L(w_{i-1})}\right.\Big) \tag{5.56}$$

where word w_i is decomposed by some decomposition function L into $L(w_i)$ particles $u_1, \ldots, u_{L(w_i)}$. The particle language model then computes the probability of a particle given its history consisting of all particles up to the last particle in the preceding word. Two data-driven methods for deriving particles were compared: a greedy search over possible units of a fixed length, retaining those particles that maximizing the likelihood of the data, and a particle-growing technique whereby particles are initialized to all single-character units and are successively extended by adding surrounding characters such that the resulting units minimize perplexity.

Kiecza, Schultz, and Waibel [88] proposed an approach to Korean language modeling whereby elementary syllable units are combined into units larger than syllables but smaller than Korean words (called *eojols*), which are of a complexity similar to Turkish words. Syllable combination was done by minimizing OOV rate. In both these approaches, significant improvements in perplexity/OOV rate, but only limited gains if any in the final system evaluation (speech recognition word-error rate), were reported.

More recently, the choice of subword units has been optimized with respect to the final system performance, usually by trying different segmentation schemes and evaluating each with respect to its effect on the performance metric. An example is presented by Arisoy, Sak, and Saraçlar [102], where words, statistically derived units, linguistically defined morphemes, and stems plus endings were compared for the purpose of Turkish language modeling for speech recognition, with the result that stems plus endings yielded the best word-error rate out of all four choices.

5.7.3 Modeling with Morphological Categories

Most of the work on language modeling with subword units has focused on linear decomposition of words, primarily for agglutinative languages. As a consequence, the resulting subword units are most frequently used in a standard n-gram model. As mentioned earlier, one problem is that the n-gram context needs to be increased in order to be able to model interword dependencies in addition to dependencies between subword units. This in turn places demands on the amount of training data needed.

However, several alternative approaches have been developed wherein the word remains the basic modeling unit but the probability assignment takes into account statistics over subword components or morphological classes. Arisoy et al. [46] proposed a discriminative language model (see §5.6.3) for Turkish that incorporates feature functions defined over

morphemes, such as root n-gram counts or inflectional group counts. The language model assigns probabilities to sequences of entire words (undecomposed n-best hypotheses), but does so by taking into account the constraints provided by morpheme-based feature functions. This model was shown to provide slightly better results (0.3% absolute word-error rate reduction on a Broadcast News recognition task) than a discriminative language model using only word-based features. The same approach was taken by Shafran and Hall [45] for Czech, with similar results.

Kirchhoff et al. [63] and Vergyri et al. [68] used both morphological classes (stem, root, etc.) and words as conditioning variables in a factored language model for Arabic. Although the model predicts probabilities for entire word forms, the probabilistic backoff procedure makes use of morphological constituents. On a dialectal Arabic speech recognition task with a limited amount of training data, FLMs yielded slight reductions in word error rate (0.5%–1.5% absolute). FLMs have also been used successfully for speech recognition of other highly inflected languages, such as Estonian [103], and for machine translation [104].

Morphological features were also used as additional input features (beyond words) to a neural language model (see §5.6.9) for both Arabic and Turkish [85], thus creating a factored neural language model. Perplexity was shown to improve substantially over a word-based neural language model (up to 10% for Arabic and 40% for Turkish), but no application results were reported.

Sarikaya and Deng [105] proposed a joint morphological-lexical language model for Arabic. Here, a sentence is annotated with a rich parse tree representing morphological, syntactic, semantic, and other attribute information. The language model predicts the probability of the word string and the associated tree jointly, using MaxEnt probability estimation (see §5.6.5). The model was evaluated on an English-to-Arabic translation task and improved the BLEU score by 0.3 points (absolute) over a word trigram model and 0.6 points over a morpheme trigram model.

RFLMs (see §5.6.7) were applied to morphological language modeling by Oparin et al. [106]. The difference from standard word-based RFLMs is that the decision trees used in the random forest model can ask questions not only about the set membership of words but also about morphological features (inflections or morphological tags), stems, lemmas, and parts of speech. Morphological RFLMs were evaluated on a Czech spoken lecture recognition task with a 240,000-word vocabulary. Whereas word-based RFLMs did not show any substantial gain over Kneser-Ney-style trigram language models, morphological RFLMs yielded a relative gain in perplexity of up to 10.4% and a relative improvement in word accuracy of up to 3.4%. Unlike previous studies' results, it was also found that interpolation of RFLMs and standard n-gram models yielded an improvement (of up to 15.6% in perplexity). In addition to producing different word history clusters (induced by morphological features rather than words), a morphological RFLM offers more possibilities for randomization because the set of potential splits at each decision tree node is greatly enlarged, which may contribute to the superior performance of morphological RFLMs in this case.

5.7.4 Languages without Word Segmentation

Although agglutination produces a large number of long and complex word forms in many languages, other languages do not possess any explicit segmentation of character strings into words at all. In languages such as Chinese and Japanese, sentences are written as sequences of characters delimited by punctuation signs but without any intervening whitespaces to

indicate word boundaries. Readers with knowledge of the language can immediately decompose character sequences into the word segmentation that corresponds to the most plausible interpretation of the sentence. Although statistical language models for such languages can be constructed over characters, it is preferable to first segment sentences into words automatically and then train the language model over the resulting words. Similar to the situation in which words are decomposed into subword units (§5.7.1), a language model using characters as the basic modeling units may fail to express important interword relationships. Moreover, the word segmentation may determine how characters are pronounced, which is important if the same modeling units are intended to be used in the language model and the acoustic model of a speech recognition system. Finally, it has been shown experimentally that a language model built on automatically segmented text outperforms a character-based language model in terms of perplexity for both Japanese [107] and Chinese [108].

Automatic segmentation algorithms typically use a combination of dictionary information, statistical search, and additional features such as nonnative letters, character co-occurrence counts, and character positions. Generating the most likely segmentation follows a statistical decoding framework that uses, for example, Viterbi search. Other modeling approaches that have been explored recently include conditional random fields [109, 110, 111], MaxEnt modeling [112, 113], and discriminative modeling using perceptrons [114]. Much of the work on Chinese word segmentation has been performed in the context of the SIGHAN Chinese word segmentation competitions that have been taking place since 2003, organized by the Association for Computational Linguistics. This competition serves as a benchmark comparing different word segmentation systems. Automatic word segmentations are compared against a linguistically segmented gold standard and are evaluated in terms of precision (P), that is, the percentage of correct cases out of all hypothesized boundaries, recall R (the percentage of identified boundaries out of all possible boundaries), and their combination in the form of F-measure, $F = 2PR/(P+R)$. These measures can be calculated separately on the OOV words versus in-vocabulary words. The best-performing systems in the most recent evaluation achieved an F-measure of 0.96; however, performance on OOV words was much lower with F-measures around 0.76 [115].

Rather than trying to match linguistically defined words, it is also possible to optimize segmentation directly for language model performance. Sproat et al. [108] showed that the dictionary used for Chinese word segmentation significantly influences the perplexity of a bigram language model trained on the segmented text and that it is possible to iteratively optimize the dictionary by merging frequent word co-occurrences, such that the perplexity is reduced at each iteration. Note that such approaches are similar to the data-driven algorithms for deriving morpheme-like subword units described earlier in Section 5.7.1. Another example of a data-driven approach, in this case for Japanese, uses chunks of characters for language modeling [107]. Chunks are derived from the training data by selecting the highest-frequency n-grams and additional patterns with close similarity to them. Thus, the basic modeling units are neither characters nor words but intermediate units.

5.7.5 Spoken versus Written Languages

Statistical language modeling crucially relies on large amounts of written text data, and a significant trend within language modeling research is the development of methods for

scaling current language modeling techniques to ever-larger databases. However, many of the world's 6,900 languages are spoken languages, that is, languages without a writing system. They are either indigenous languages without a literary tradition, or they are linguistic varieties such as regional dialects that are used in everyday spoken communication but are rarely put into writing. This is the case, for example, with the many dialects of Arabic, which are used in everyday conversation but are almost never found in written form. Other languages may be spoken as well as written but may not have a standardized orthography.

Both cases represent difficulties for language modeling. In the first case, the only way of obtaining language model training data is to manually transcribe the language or dialect. This is a costly and time-consuming process because it involves (i) the development of a writing standard, (ii) training native speakers to use the writing system consistently and accurately, and (iii) the actual transcription effort. In the second case, those text resources that can be obtained for the language in question (e.g., from the web) will need to be normalized, which can also be a laborious process. As a consequence, very little work has been carried out on language model development for such underresourced languages. Most studies have concentrated on how to rapidly collect corpora for underresourced languages using web resources. Some of the challenges inherent in this process are described by Le et al. [116] and Ghani, Jones, and Mladenic [117]. Possible methods for rapid language model bootstrapping for spoken languages and languages characterized by a lack of standardization might include grammar-based or class-based approaches in combination with a limited amount of transcribed material. For a constrained application, such as the development of a dialog system, the structure of possible utterances could be predefined by a task grammar or a class-based language model, whereas more fine-grained word sequence probabilities, or probabilities of words given their classes, are modeled by a language model trained from a small amount of data. An interesting research direction is the possible use of data from language that is closely related to the language in question or from a language that is unrelated but resource rich. The following section describes some of these approaches.

5.8 Multilingual and Crosslingual Language Modeling

5.8.1 Multilingual Language Modeling

Up to this point we have discussed problems that arise when tailoring statistical language models to a particular language or language type, such as agglutinative languages or languages without word segmentation. The tacit assumption has been that the resulting language model is used in an application where only the language of interest is encountered. However, in many situations, a system can be presented with multiple languages sequentially (e.g., different users speaking different languages, without advance indication of which language will be encountered next), or simultaneously, as happens in the case of **code switching**. Here, speakers may use several languages or dialects side by side, often within the same utterance. The phenomenon of code switching exists in a variety of bilingual and multilingual communities or where diglossia exists, such as where a formal standard language is used in addition to colloquial or dialectal varieties. The use of "Spanglish" (mixed Spanish/English)

in the United States is one example of code switching, as demonstrated by the following example from Franco and Solorio [118]:

I need to tell her que no voy a poder ir.
'I need to tell her that I won't be able to make it.'

To handle multilingual input where language can be switched dynamically between utterances, separate language models can be constructed from monolingual corpora, and a system using these models (e.g., a speech-based information kiosk or telephone-based dialog system) can access them dynamically based on the output from a first-step language identification module or based on which language model (possibly combined with an acoustic model in the case of speech recognition) yields the highest scores after the first processing steps.

Fügen et al. showed how several such monolingual models can be combined into a single multilingual language model by means of a context-free grammar whose nonterminal states can encode language information and whose terminal states correspond to monolingual n-gram models. Explicit grammar rules can be leveraged to expand current states with n-grams from the matching language only, thus preventing language switching at inappropriate times. Alternative ways of constructing a single multilingual language model are to combine monolingual corpora and train a single model on the pooled data or to interpolate several monolingual language models. The first technique has been shown to degrade performance, especially when the sizes of the corpora are unbalanced [120, 121]. The second technique may fare slightly better yet was shown to be inferior to the grammar-based combination method described earlier [119].

Building a language model for the second case (intrasentential language switching) is more difficult because little or no relevant training material is available. Weng et al. [122] built a four-lingual language model by introducing a common backoff node in the form of a pause unit; that is, language switches are allowed with some probability after a pause has occurred.

5.8.2 Crosslingual Language Modeling

Another question that might be asked is whether data in one language can be leveraged to improve a language model for a different language, provided that the style or domain are closely related. If insufficient data is available for the language of interest, inaccurate probability estimation might be improved if sufficient information can be extracted from a large amount of foreign-language text.

The most straightforward way of applying this idea is to automatically translate the foreign-language text into the desired language and to use the translated text (though errorful) as additional language training data. This approach was chosen, for example, by Khudanpur and Kim [123] and Jensson et al. [124].

In the former study, training data for a Mandarin news text language model intended for speech recognition was enriched with training data obtained by automatically translating English text from the same domain. A unigram extracted from the translated text was interpolated with a trigram baseline language model trained on the available Mandarin data. The selection of English text for translation, and the determination of the interpolation parameter λ were specific to each news story, thus providing at the same time an implicit

form of topic adaptation. The resulting model improved character perplexity by about 10% relative and the word-error rate of the speech recognizer by 0.5% absolute (for various different systems ranging around 26% baseline character-error rate). The authors also noted that the English text was more recent than the Chinese text, which may have contributed to the improved performance. This is an important consideration when investigating potential out-of-language data resources.

Jensson et al. [124] developed a language model for weather reports in Icelandic using a small amount of in-language data and a limited amount of parallel English–Icelandic data for training a machine translation system. A language model trained on a larger set of automatically translated data was then interpolated with the baseline language model, with positive effects on perplexity (9.20% improvement) and word-error rate (1.9–9.5% relative improvement) of an Icelandic speech recognition system.

The use of machine translation technology for processing out-of-language data is likely to fail when the desired language does not have sufficient data to train a machine translation system in the first place, although the abovementioned experiments on Icelandic show that a limited amount of parallel data may still be sufficient if the domain is very constrained. An alternative to using a fully fledged machine translation system is to rely on a high-quality word-based translation dictionary only. Kim and Khudanpur [125] showed that a good-quality translation dictionary can be extracted by simply computing mutual information statistics on word pairs from document-aligned parallel corpora, eliminating the need for sentence-aligned parallel text. In their experiments on Mandarin broadcast news recognition, they found that a unigram model built from the resulting dictionary-based translations and interpolated with a baseline language model achieved a performance similar to that of a crosslingual language model. Another possibility for constructing a translation dictionary from document-aligned data is to use crosslingual latent semantic analysis [126]. Under this approach, words in both languages are projected into a common semantic space, and a measure of similarity between words from different languages in this space is used to construct word translation probabilities.

A drawback of the previous approaches is that the quality of the resulting model is heavily dependent on the translation accuracy. Tam et al. [127] recently presented an alternative model in which bilingual latent semantic analysis (bLSA) is used for adaptation before translation. The approach uses one LSA model each for the source and the target languages; it thus requires a parallel training corpus. The LSA models incorporate Dirichlet-style prior distributions over topics (see §5.6.8). Topic mixture weights are determined from the source LSA model and are projected onto the target LSA model, where they can be used to compute target-language marginal distributions. Assuming that the source language is Chinese (Ch) and the target language is English (En), the marginal word probability distribution in English is

$$P_{\mathrm{En}}(w) = \sum_k \phi_k^{\mathrm{En}}(w)\theta_k^{\mathrm{Ch}} \tag{5.57}$$

where θ_k is the prior for the k^{th} topic and $\phi_k(w)$ is the probability assigned to word w by the k^{th} latent topic. As we can see, topic priors are determined by the source language, whereas the topic-dependent word probability distribution is determined by the target language. The

target-language marginals are then incorporated into the target-language model as follows:

$$P_{\text{target}}(w|h) \propto \left(\frac{P_{\text{bLSA}}(w)}{P_{\text{base}}(w)}\right)^{\beta} P_{\text{base}}(w|h) \tag{5.58}$$

where P_{bLSA} is the adapted probability and P_{base} is the baseline probability. This approach enforces a one-to-one topic correspondence across languages. Evaluation on a Chinese–English statistical machine translation task in the news domain showed that the bLSA-adapted language model reduced perplexity by 9% to 13% and improved the BLEU score by up to 0.3 points.

5.9 Summary

Statistical language modeling has undergone drastic developments in recent years. Although the classical n-gram model in combination with smoothed maximum-likelihood estimation is still the predominant approach, many novel models, ranging from neural network models to discriminative language models, are now being used side by side with standard n-gram models.

Many language modeling techniques, such as language model adaptation, have proven applicable to languages of vastly different types, and the core techniques can be said to be language independent. Crucial differences exist in those cases where languages have a rich morphology, in particular in highly agglutinative languages that can produce a very large number of different word forms per lexeme. In these cases, word decomposition prior to n-gram language modeling or the integration of statistics over subword components into discriminative, factored, or neural language models has been helpful.

A relatively recent trend is the use of very large-scale, distributed language models, which do not follow the traditional probability estimation scheme but use approximate scores or counts. Given that the amount of language modeling data is increasing daily, this trend will certainly become very influential in the near future. It also has significance for languages with large vocabularies (e.g., morphologically rich languages) because it facilitates the use of large language models in practical systems.

Comparatively little research has been carried out on languages that can be characterized as resource poor, that is, languages that do not have a large amount of text data. This includes most of the spoken languages and dialects in the world. At this point, the standard way of obtaining data for these linguistic varieties is to transcribe them manually, which will yield only a limited amount of data. The use of speech recognition technology in combination with a bootstrapping technique could be used to automatically transcribe more data in an incremental fashion. However, there is a chicken-and-egg problem in that the development of sufficiently accurate speech recognition systems requires a sufficient amount of both text and acoustic data to train initial models. The further development of crosslingual adaptation techniques for language models is an important future research direction, which would make a significant contribution toward applying human language technology to resource-poor languages.

Bibliography

[1] J. Ponte and B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the ACM Special Interest Group on Information Retrieval (SIGIR)*, pp. 275–281, 1998.

[2] F. Peng, D. Schuurmans, S. Wang, and V. Keselj, "Language independent authorship attribution using character level language models," in *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 267–274, 2003.

[3] F. Peng, D. Schuurmans, and S. Wang, "Language and task independent text categorization with simple language models," in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, pp. 110–117, 2003.

[4] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4(1), pp. 31–44, 1996.

[5] M. Nagata, "Japanese OCR error correction using character shape similarity and statistical language model," in *Proceedings of the Association for Computational Linguistics*, pp. 922–928, 1998.

[6] I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for English and Arabic," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 495–504, 1999.

[7] S. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Tech. Rep. TR-10-98, Harvard University, 1998.

[8] F. Jelinek and R. Mercer, "Interpolation estimation of Markov source parameters from sparse data," in *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[9] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39(1), pp. 1–38, 1977.

[10] M. Federico, "Bayesian estimation methods for n-gram language model adaptation," in *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP)*, pp. 240–243, 1996.

[11] A. Nadas, "Estimation of probabilities in the language model of the IBM speech recognition system," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 859–861, 1984.

[12] A. Emami, K. Papineni, and J. Sorensen, "Large-scale distributed language modeling," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 37–40, 2007.

[13] T. Brants, A. Popat, P. Xu, F. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pp. 858–867, 2007.

Chapter 11

Multilingual Information Retrieval

Philipp Sorg and Philipp Cimiano

Research on monolingual information retrieval (IR) dates back at least to the 1960s and represents by now an established and mature research field. The subfield of crosslingual information retrieval has observed a surge of interest in recent years. The reasons for this surge are many. First, it is certainly the case that improvements in related technologies, especially in machine translation (MT), have fostered the development of effective multilingual retrieval systems. Second, the number of non-English Internet users has been growing at a fast rate.[1] As a consequence, we find much more non-English content on the Web compared to several years ago. Further, the advent of Web 2.0 technologies has accelerated the need for crosslingual retrieval techniques. The reason is that, although professional websites are usually translated into all relevant languages, it is clearly not the case for the massive amount of user-generated content in Web 2.0 applications such as Flickr, Yahoo! Answers, Facebook, and Twitter. Finally, for multinational companies or globally acting organizations and communities, crosslingual retrieval is an essential need.

In this chapter we present current approaches to crosslingual information retrieval (CLIR) and multilingual information retrieval (MLIR). CLIR involves two languages—the query language and the collection language—and consists in answering documents in the query language on the basis of a language-homogeneous document collection. In contrast, MLIR involves arbitrarily many languages and consists in answering queries in any language supported by the system on the basis of a document collection containing documents in different languages.

Because the main approaches applied to CLIR and MLIR are based on fundamental IR techniques, we also introduce the IR foundations that are needed to understand these approaches. We present alternative design choices for the development of CLIR and MLIR systems along with best practices. The chapter should thus be of interest to both researchers looking for an overview of the main methods used in CLIR and MLIR and developers implementing multilingual information systems. We discuss alternative choices for document models, retrieval functions, and translation approaches depending on the languages that are supported by the retrieval systems. This includes a discussion of language-specific document preprocessing, statistical retrieval models, MT systems, and how the outcome of IR systems can be evaluated.

1. http://www.internetworldstats.com/stats7.htm

11.1 Introduction

Information retrieval deals with the representation, storage, organization of, and access to information items [1]. IR systems with suitable representation and organization of the information items allow users with a specific information need to access the information they are interested in.

Typically, users manifest their information need in the form of a query, usually a bag of keywords, to convey the need to the IR system. The IR system then retrieves items it believes are relevant to the user's information need. In contrast to question answering (QA) systems, which return direct answers to questions, the goal of an IR system is to deliver a list of documents ranked by relevance to the user's query. The target of an IR system is to rank the relevant documents as high as possible and the nonrelevant as low as possible.

In this chapter, we focus on multilingual aspects of information retrieval. In CLIR and MLIR, the information need and the corresponding query of the user may be formulated in a different language than the relevant information is expressed in. While relevance is in principle a language-independent notion, the user must also be able to understand the retrieved items. Multilingual search systems have to ensure that only items in languages supported by the user are returned or that the information items are translated to one of these languages.

11.2 Document Preprocessing

In this section we cover the preprocessing of documents. Preprocessing takes a set of raw documents as input and produces a set of **tokens** as output. Tokens are specific occurrences of terms (types) in a document that represent the smallest unit of meaning. This output defines the **vocabulary** that can be used to index a collection of documents, as described in Section 11.3.

As in most extant IR models and systems, we make the simplifying assumption that the order of tokens in a document is irrelevant. **Phrase indices** or **positional indices** are examples of approaches making use of the order of tokens (see Manning, Raghavan, and Schtze [2] for an introduction).

Depending on language, script, and other factors, the process for identifying terms can differ substantially. For Western European languages, terms used in IR systems are often defined by the words of these languages. However, terms can also be defined as a fixed number of characters in sequence, which is often the case in IR systems applied to Asian languages. Taking Chinese as an example, words are not separated by whitespaces, such that defining terms by character sequences bypasses the problem of having to recognize words.

In the following sections we introduce common techniques used for document preprocessing. In our discussion, we emphasize the differences in preprocessing for different languages and scripts. This includes, in particular, a discussion of the differences with respect to document syntax, encoding, tokenization, and normalization of tokens. The complete preprocessing pipeline is illustrated in Algorithm 11–1. This pipeline shows the dependencies of the different preprocessing steps described in the following.

Algorithm 11–1 The preprocessing pipeline of document d that results in a set of tokens T

```
d ← INPUT
T ← ∅
[c_1, c_2, ...] ← character-stream(d)
B ← tokenize([c_1, c_2, ...])
while B ≠ ∅ do
  t ← POLL(B)
  if is-compound(t) then
    B ← B ∪ compound-split(t)
  end if
  if not is-stop-word(t) then
    t = normalize(t)
    T ← T ∪ {t}
  end if
end while
return T
```

11.2.1 Document Syntax and Encoding

The first step in the preprocessing pipeline is to identify documents in the given data stream [2]. In many cases, this is a straightforward task because the researcher can assume that one file or web site corresponds to exactly one document. However, there are other scenarios with files containing several documents (e.g., XML retrieval) or documents spread over several files (e.g., web pages). The definition of what exactly constitutes a document is thus essentially a design choice of the developer and depends on the search task, that is, the kind of information items that should be retrieved.

The next step is to transform documents into character streams representing the content of documents. The goal of this step is to get a unified representation with respect to encoding, script and direction of script. Two documents with the same content in the same language thus should have identical character streams after this step. The following challenges have to be addressed:

Document Syntax The content of documents is usually encoded in a syntax specific for the given **file type**. Based on the file type specification, the textual content of a document has to be extracted before indexing, avoiding, in particular, that vocabulary elements containing formatting instructions or metadata information are indexed.

Examples for file types that require content extraction are PDF files or web pages. For both cases, many libraries are available that parse PDF or HTML files and extract the textual content.

In many scenarios only parts of the textual content contribute to the semantic content of an individual document. Other parts might be equal across documents (e.g., header or footer parts). In these cases, indexing the complete text also introduces noise. For specific document formats, extractors based on the structure of documents are usually applied to identify the relevant text parts. In the case of web pages for example, the extraction depends on the individual layout of the site. While elements from the header, such as title or keywords,

might describe content and should be extracted, the top bar or menus, which are identical on all web pages in the collection, should be ignored.

Encoding and Script Encoding is the representation of letters through a number of bytes in a computer system. Historically, the ASCII character encoding scheme was widely used to encode English documents. Because this scheme is mainly limited to the Latin alphabet, it cannot be used for scripts having different letters. As an encoding scheme supporting most common languages, Unicode [3] established itself as the de facto standard for internationalized applications. The unique number of every character across all languages ensures high portability across platforms and avoids errors introduced by conversion. Unicode also supports right-to-left scripts and can be used to encode, for example, Arabic or Hebrew. Because most operating systems and most current programming languages support Unicode, it is highly recommended to use this character encoding as default.[2]

For IR systems, it is essential that queries and documents are represented in the same script. At some level, retrieval boils down to matching characters, which is unsuccessful when query and document scripts are not compatible. Korean is an example of a language that has different common scripts, namely, Hangul and Hanja. The process of mapping text into another script is called **transliteration**. This must not be confused with translation, as the language is not changed. Transliteration attempts to mimic the sound of the word in the original language by using the spelling of a different language. It is therefore a phonetic transformation that is typically reversible.

For preprocessing of data sets containing documents in heterogeneous scripts, **romanization** is a common technique used to get a unified representation. Romanization is the transliteration of any script to the Latin (Roman) alphabet. For retrieval systems, this is especially useful when searching for common names. Applying romanization, common names used in documents of different languages and scripts are mapped to the same character sequence in most cases. As part of the United Nations Group of Experts on Geographical Names (UNGEGN), the Working Group on Romanization Systems [4] provides romanization resources for various languages. The motivation of this working group is to introduce unique representations of geographic names. However, the resources provided can be used for romanization of any text.

Direction of Script Because scripts are used to record spoken language, a natural order of words and characters is defined by their order in the speech stream [2]. Usually byte representations of text also reflect this natural order. The actual direction of script is handled by the visualization layer in applications, which is part of the user interface. The main problems are typically documents containing text in different languages with different directions of script. An example is Arabic texts with English common names. As we focus on the core functionality and models of MLIR in this chapter, we do not discuss these difficulties any further. We only operate on the data level of documents, which is usually independent of the direction of the script. When designing user interfaces, this is a more important issue.

2. Technically, Unicode is a mapping from characters to code points. The set of Unicode encodings includes UTF-8 and UTF-16.

11.2.2 Tokenization

Tokenization is the process of splitting a character stream into tokens. Tokens are instances of terms and correspond to the smallest indexation unit. The set of all terms is typically called the vocabulary. In the following we introduce three common types of vocabularies that require different tokenization approaches. The choice of vocabulary is an important design choice for any IR system. Guidelines for this decision can be found in Section 11.2.4.

To illustrate the different tokenization approaches, we use the following sentence as running example:

```
It is a sunny day in Karlsruhe.
```

Word Segmentation The most common approach to tokenization is splitting text at word borders. Thus, tokens refer to words of a language, and the vocabulary is equivalent to a lexicon (including morphemes).

For languages that use whitespaces to separate words, this is a successful approach used in most IR systems. Whitespaces and punctuations are thereby used as clues for splitting the text into tokens. Examples for such languages are Western European languages. The problem of this approach is that simply splitting text at all whitespaces and punctuations will also split parts of the text that should be represented by a single token. Examples of such error sources are hyphens *(co-education)*, white spaces in proper nouns *(New York)*, dates *(April 28, 2010)*, and phone numbers [2]. In many cases heuristics are used to decide whether or not to split. Classifiers can also be trained on this decision. For our running example, splitting using whitespaces results in the following tokens:

```
[It], [is], [a], [sunny], [day], [in], [Karlsruhe]
```

Tokenization at word borders is a much harder problem for scripts without whitespaces, such as Chinese. Approaches can be classified into two classes: lexical and linguistic. Lexical approaches match terms from a lexicon to the token stream to get a complete coverage. Usually this matching is not deterministic. To get the most accurate matching, heuristics can be applied, such as always preferring to match longer terms. A problem for such approaches is unknown terms that are not in the lexicon and will not be matched but should be detected as such. Linguistic approaches make use of background knowledge consisting of already tokenized text. Using statistical measures based on the frequency of tokens, the goal is to find the most probable segmentation of the current text. Hidden Markov models can be used to compute this in an efficient way [5]. Machine learning techniques like conditional random fields have also been successfully applied to this problem [6]. Because no approach ever achieves a perfect segmentation, wrong tokens will be used for indexing and search and will therefore influence the retrieval performance.

Phrase Indices Phrase indices are based on word segmentation. Tokens are thereby defined not as single words but as tuples of words. Phrase indices are also known as ***n*-gram models**, with n defining the number of words in each token. The character stream that has been already split into words is mapped to tokens by moving a window of n words iteratively over the text. These tokens preserve the context of words. However, this process comes at the cost of a very huge vocabulary. Another problem for search is the sparseness of terms; that is, many terms in queries will not be present in the collection at all. To circumvent this

problem, phrase indices can be used on top of a retrieval approach based on single-word segmentation. For our running example, a 3-gram model results in the following tokens:

```
[It is a], [is a sunny], [a sunny day], [sunny day in], ...
```

Character *n*-Gram Models Character n-gram models define terms as n subsequent characters. Character streams are tokenized by moving a window of n characters over the text. In this case terms do not correspond to words. The vocabulary is defined as the set of sequences having n characters, including whitespaces and punctuation. Term lengths of 4 or 5 have been shown to be reasonable. For our running example, a character 4-gram model results in the following tokens:

```
[_It_], [It_i], [t_is], [_is_], [is_a], [s_a_], [_a_s], ...
```

This approach can be applied to any character stream and does not depend on word border clues such as whitespaces. It can therefore be used to tokenize text of any script. Because no segmentation is needed, it also avoids errors introduced by word segmentation. In the literature, this approach has been proven superior to word-based segmentation in several scenarios [7]. It has also been applied to the problem of spelling correction [2]. In the context of multilingual retrieval, character n-gram tokenization can be used only if no mapping of terms into different languages is needed. Terms do not correspond to words and thus cannot be mapped or translated across languages. Another drawback using character n-grams is the more difficult visualization of search results. Because only n-grams are matched, it is not possible to highlight matching words for search results.

11.2.3 Normalization

The goal of normalization is to map different tokens describing the same concept to the same terms. An English example is the mapping of plural forms to their singular forms, like *cars* to *car*. Normalization can be defined as building equivalence classes of terms. In a search scenario, normalization can be used to increase the number of relevant documents retrieved and thus also increase the recall of the system. The same normalization methods have to be applied to the collection before indexing and to the query before search. This ensures that all tokens are mapped to equivalent terms, which is crucial for matching queries to documents.

The approach to normalization differs between languages. For languages having complex morphology, a common approach is to map (compound) terms to their lemma(s). Examples are Roman and Germanic languages. There are two main approaches to this problem. First, **lemmatizers** use lexical information to map terms to lemmas. For this approach, rich linguistic resources are required. Second, **stemmers** use a set of simple rules to map terms to stems. For the plural example, the rule of deleting a trailing *s* would map both terms to the same equivalence class. Stemmers do not require rich language resources. The drawback is that terms are not mapped to lemmas but to stems, which do not necessarily correspond to a word. In many cases terms describing different concepts might also be mapped to the same stem. For example, the terms *organize*, *organizing*, and *organization*, would be mapped to *organ*, which makes it impossible to distinguish these terms in the index. In contrast, a lemmatizer would correctly normalize *organize* and *organizing* to the lemma *organize* without changing the term *organization*.

Normalization might also be useful for scripts using diactritics. If the use of diactritics is not consistent, it is useful to delete them in the normalization step. For example, if users do not specify diactritics in queries, normalization should also delete them before indexing. Simple rule-based approaches are normally applied to remove diactritics.

For fusional languages (for example, German, Dutch, Italian) **compound splitting** is another form of normalization. In such languages, compound terms are typically split into the composing lemmas to increase recall. The problem of compound splitting is quite similar to the problem of word segmentation in Asian languages as described earlier. Lexical approaches use a lexicon to match terms in compounds. Linguistic approaches additionally use background knowledge. Many approaches compare the frequency of the compound itself and the frequency of its constituent terms to decide whether or not to split. When applying compound splitting, usually both compounds and split components are added to the token stream. In the search process, this still allows the matching of compounds.

Removal of **stop-words** is a normalization step that deletes frequent terms from the token stream. These terms occur in almost all documents and are therefore not useful to discriminate between relevant and nonrelevant documents. Stop-words are usually articles, prepositions, or conjunctions. For many languages, compiled lists of stop-words exist that can be used to match and filter tokens.

Coming back to our running example, stemming and stop-word removal would result in the following tokens:

```
[sunny], [day], [karlsruh]
```

11.2.4 Best Practices for Preprocessing

After introducing different steps and variants of preprocessing in the previous section, we now present some guidelines for preprocessing distinguishing different types of languages.

Languages Using Latin or Cyrillic Alphabet Because these languages use whitespaces to separate words, word segmentation is the preferable tokenization approach. This approach should be enhanced—depending on the search task—with special treatment for common names, dates, phone numbers, and so on. Stemming or lemmatization usually improves retrieval results for these languages, but this improvement is not always significant [1]. Stemming can be implemented at a modest cost, so it is typically worth doing it. If high precision in finding relevant documents is required, normalization can, however, decrease the retrieval quality. For fusional languages, compound splitting has shown to improve performance up to 25% [8].

Languages Using Arabic, Devanagari or Hebrew Script Word segmentation using whitespaces can be used for these languages and is therefore recommended. Because these languages are not morphologically rich, morphological analysis (stemming or lemmatizing) is not crucial, but diacritics in these languages have to be handled with care. In particular, preprocessing needs to ensure that query and document tokens are transformed into a canonical representation with respect to diacritics.

Languages Using Logographic or Syllabic Scripts Some of these languages (e.g., Korean, Japanese) use more than one writing system, so queries and/or documents might be written in various scripts. In this case, transliteration of queries/documents is needed to ensure

compatibility in the search process. In these scripts, words are usually not separated by whitespaces. If rich language resources are available for a language (e.g., Chinese), word segmentation based on heuristic rules or machine learning has been shown to produce good results [6]. However, without these resources, character n-gram tokenization can be applied. This approach is language-independent and avoids sophisticated approaches to word-border detection. It has been shown to be robust and to achieve comparable results to other systems based on word segmentation on European languages [7].

11.3 Monolingual Information Retrieval

Most approaches to MLIR are either directly based on monolingual IR techniques or make use of standard IR models. MLIR can be seen as the problem of aggregating the results of IR systems in different languages. Apart from aggregation, language-specific preprocessing of queries is needed, particularly translation (covered in §11.5). In general, MLIR is based on the same index structures and relies on similar document and retrieval models as known from monolingual information retrieval. In this chapter, we therefore give a short overview of monolingual information retrieval, including document representation, index structures, retrieval models, and document a priori models. We focus on those aspects of information retrieval that are also relevant for CLIR and MLIR. For more details concerning monolingual information retrieval, see Manning et al. [2] and Baeza-Yates and Ribeiro-Neto [1].

11.3.1 Document Representation

In Section 11.2, we described the preprocessing of documents. It results in **token stream** representations of documents. The tokens are instances of terms, which are defined by words, stems or lemmas of words, or character n-grams. The IR models presented in this chapter are independent of the used vocabulary and can be applied to any term model. For the sake of presentation, we make the simplifying assumption that terms correspond to words in spoken language throughout this chapter, as this yields the most intuitive vocabulary for humans.

Most current retrieval approaches use document models based on the **independence assumption** of terms. This means that occurrences of terms in documents are assumed to be independent of the occurrences of other terms in the same document. Although this assumption is certainly overly simplistic, retrieval models based on this assumption achieve reasonable results with current IR technology.

Given the independence assumption, documents can be represented using the **vector space model**. This vector space is spanned by the vocabulary in such a way that each dimension corresponds to a specific term. Documents are represented as vectors by a mapping function f, which maps token streams of documents d to term vectors $\vec{d}$. Different functions f are used in literature, the most prominent being

- *Boolean document model:* The value of a term dimension is set to 1 if the term occurs at least once in the document, otherwise to 0.

- *TF document model:* The value of each dimension depends on the number of occurrences of terms in the document token stream, that is, the **term frequency**. The term frequency can be directly used as value in the term vector. Variants are, for example, the normalization of the term frequency by document length.
- *TF.IDF document model:* These models additionally multiply term frequency values by the **inverse document frequency** of terms. The document frequency of a term is the number of documents in the collection containing this term. The inverse document frequency therefore puts more weight on infrequent terms and less weight on frequent terms that do not discriminate well between documents in the collection. In most cases, the logarithm of the inverse document frequency is used in TF.IDF models.

Given a collection of documents, the document term vectors can be aligned to form the **term-document matrix**. This matrix is spanned by terms as rows and documents as columns. We illustrate the different document representations using the following documents:

```
Doc1: It is a sunny day in Karlsruhe.
Doc2: It rains and rains and rains the whole day.
```

For the different document models discussed, this results in the following term-document matrices:

	Boolean		*TF*		*TF.IDF*	
Term	Doc1	Doc2	Doc1	Doc2	Doc1	Doc2
sunny	1	0	1	0	$1 \log 2/1 = 0.7$	0.0
day	1	1	1	1	$1 \log 2/2 = 0.0$	$1 \log 2/2 = 0.0$
Karlsruhe	1	0	1	0	$1 \log 2/1 = 0.7$	0.0
rains	0	1	0	3	0.0	$3 \log 2/1 = 2.1$

11.3.2 Index Structures

An important aspect of information retrieval is time performance. Users expect retrieval results in almost real time and delays of only 1 second might be perceived as a slow response. The simplistic approach to scan through all documents given a query obviously does not scale to large collections. The high time performance of current retrieval systems is achieved by using an **inverted index**. The idea is to store for each term the information in which documents it occurs. This relation from terms to documents is called **posting list**; a detailed example can be found in Manning et al. [2]. During retrieval, only posting lists of query terms have to be processed. Because queries usually consist of only a few terms, the scores can be computed with low average time complexity.

For the preceding example documents, we get the following posting lists:

```
sunny     -> doc1(1x)
day       -> doc1(1x), doc2(2x)
Karlsruhe -> doc1(1x)
rains     -> doc2(3x)
```

A remaining bottleneck using inverted indices is memory consumption. Loading of posting lists from storage to main memory is the slowest part and should be avoided. Heuristics are therefore needed to decide which posting lists should be kept in memory and which should be replaced. General approaches to reduce memory usage, for example by compression or by usage of suffix trees, are described in Baeza-Yates and Ribeiro-Neto [1]. For very large corpora, distributed indexing can be applied. Posting lists are distributed to several servers. Each server therefore indexes the posting lists of a subset of the vocabulary.

To reduce the time complexity of retrieval, **inexact retrieval models**—also known as **top-k models**—can be applied. These models determine documents that are most likely to be relevant without processing all matching documents. Using these methods, retrieval time can be reduced without getting significant losses in retrieval performance [9].

11.3.3 Retrieval Models

Retrieval models are used to estimate relevance of documents to queries. Different theoretical models have been used to derive these relevance functions. In the following we describe three main families of retrieval models: **boolean models**, **vector space models**, and **probabilistic models**. Depending on the retrieval model, queries are represented in different ways. Boolean queries used for boolean models are modeled as a binary term vector. As defined earlier, the order of query terms is lost in this representation, as only the presence or absence of terms is captured. For vector space and probabilistic models, queries are represented in real-valued vector space, and scores for each query term are accumulated [2].

Boolean Models Boolean models were the first retrieval models used in the beginning of information retrieval. In the case of the Boolean retrieval model, relevance is binary and is computed by matching binary vectors representing term occurrence in the query to binary document vectors representing term occurrence. Because current vector space or probabilistic models outperform boolean models, we focus on these models in this chapter. The interested reader is referred to Manning et al. [2] for details.

Vector Space Models Vector space models are based on vector space representations of documents. As described earlier, this vector space is spanned by the vocabulary, and entries in the term-document matrix are usually defined by term frequencies. There are different models to assess the relevance of documents to a given query:

1. *Accumulative model:* The retrieval function computes scores for each query term. The query term scores are summed up per document to get a final accumulated score for each document. Functions computing scores for a single query term t are based on the following measures:
   - $\mathrm{tf}_d(t)$. Term frequency in the document.
   - $|d|$. Length of the document.
   - $\mathrm{df}(t)$. Document frequency of the query term.
   - $\mathrm{tf}_D(t)$. Number of tokens of the query term in the whole collection.
   - $|D|$. Number of documents in the collection.

   For example, the accumulated score of a simple retrieval model based on term frequency and inverse document frequency is computed as follows:

$$\mathrm{score}(q,d) = \sum_{t \in q} \mathrm{tf}_d(t) \log \frac{|D|}{\mathrm{df}(t)}$$

2. *Geometric model:* The vector space representation of the query q can be interpreted as term vector $\vec{q}$. In this case, geometric similarity measures in the term vector space can be used as retrieval models [2]. For example, the **cosine similarity** has been applied successfully in retrieval scenarios:

$$\mathrm{score}(q,d) = \mathrm{cosine}(\vec{q},\vec{d}) = \frac{\langle \vec{q},\vec{d} \rangle}{||\vec{q}||||\vec{d}||}$$

Probabilistic Models In probabilistic retrieval models, the basic idea is to estimate the likelihood that documents are relevant to a given query. Relevance is thereby modeled as a random variable R taking values $\{1,0\}$. A document d is relevant for a given query q, if and only if $P(R \triangleq 1|d,q) > P(R = 0|d,q)$ [2, p. 203]. It has been shown that, given a binary loss function and the most accurate estimation of all probabilities based on all available information, these models achieve optimal performance [10]. However, in practice, it is not possible to get accurate estimations. Probabilistic models have also been used to justify design choices in heuristic functions used in vector space models: the use of the inverted document frequency is an example (see Manning et al. [2] for more details).

The BM25 model [11] is an example of a probabilistic retrieval model that has been proven to be very successful in practice. The scoring function is defined as follows:

$$\mathrm{score}(q,d) = \sum_{t \in q} \mathrm{idf}(t) \frac{\mathrm{tf}_d(t)}{k_1 \left((1-b) + b \frac{|d||D|}{\sum_{d'} |d'|} \right) + \mathrm{tf}_d(t)}$$

$$\mathrm{idf}(t) = \log \frac{|D| - \mathrm{df}(t) + 0.5}{\mathrm{df}(t) + 0.5}$$

Common values for the parameters of this model are $k_1 = 2$ and $b = 0.75$, but they should be adjusted to the search task and data set.

Language Models In recent years, language models have established themselves as powerful alternative retrieval models. Language models are a subclass of probabilistic models. Documents, queries, or whole collections are represented by generative models. These models are represented by probability distributions over terms, such as the probability that a document, query, or collection generate a certain term [12].

Maximum likelihood estimation is often used to define document models. The probability of a term t being generated by document d is then defined as:

$$P(t|d) = \frac{\mathrm{tf}_d(t)}{|d|}$$

In information retrieval, language models are used to estimate the probability $P(d|q)$, which is then interpreted as relevance score. Using **Bayes theorem**, this can be transformed to:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

Because $P(q)$ is constant for a query and $P(d)$ can be assumed to be uniform, ranking of documents is based on the value of $P(q|d)$. When modeling queries as sets of independent terms, this probability can be estimated using document language models:

$$P(q|d) = \prod_{t \in q} P(t|d)$$

This score will be zero for all documents not containing all query terms, so **smoothing** is often applied. Using background knowledge, a priori probabilities of terms $P(t)$ are estimated, and a mixture model is used for retrieval:

$$P(q|d) = \prod_{t \in q} (1-\alpha)P(t|d) + \alpha P(t)$$

Often the whole collection is used as background knowledge and the a priori probability is estimated by the language model of the collection:

$$P(t) = \frac{\sum_{d \in D} \mathrm{tf}_d(t)}{\sum_{d \in D} |d|}$$

11.3.4 Query Expansion

Query expansion is an established technique to improve retrieval performance, which is of special interest in the context of CLIR and MLIR. The query is expanded by additional terms that further characterize the information need. The goal is to match more relevant documents that contain relevant content but use other terms to describe it.

Expanded queries can be used in all retrieval models presented previously. Usually, expanded query terms are given less weight than the original query terms. The weight depends on the confidence of each expanded term and the overall weight put into query expansion. Using probabilistic retrieval models, query expansion can be used to improve the estimation of probabilities, such as the estimation of the query language model. We distinguish two different sources for expansion terms:

Background Knowledge Additional knowledge sources are exploited to find expansion terms for a given query. An example is the use of a thesaurus to expand the query with synonyms of the query terms. For CLIR or MLIR, a special case of query expansion is the translation of the query. In this case the query is expanded using the terms of its translation into different languages.

Relevance Feedback Using **relevance feedback** for query expansion is a two-step retrieval process. First, the original query is matched to the document collection. Then, relevance assessments are used to identify the relevant documents in the retrieval results.

Using an expansion model based on term frequency and document frequency in this set of **expansion documents**, promising terms are identified and used in a second retrieval step for query expansion.

The selection of relevant documents in the first step can be either manual or automatic. In the first case the user selects relevant documents manually out of the retrieval results of the first step. In the case of **pseudo-relevance feedback** (PRF), the top k documents of the first retrieval step are assumed to be relevant. This enables to implement automatic query expansion without user interaction. For this reason, PRF is often referred to as **blind relevance feedback**.

11.3.5 Document A Priori Models

In all retrieval models presented earlier, the a priori probability of documents is assumed to be uniform; that is, the probability of retrieving documents independently of a specific query is the same for all documents. However, in many scenarios this assumption does not hold. For example, documents have different perceived quality and popularity. Such factors could definitely influence the a priori probability of a document in the sense that very popular or high-quality documents should intuitively have a higher likelihood of being relevant.

For the different types of retrieval models, there are different approaches to integrate document priors. When using vector space models, these a priori probabilities can be multiplied with the IR score of each document [1]. Another option is the linear combination of scores and a priori probabilities. Weights in this linear combination have to be optimized for the special application scenario. For probabilistic and language models, the estimation of document priors $P(d)$ is required as part of the retrieval model. In the standard case without background knowledge, document priors are assumed to have equal distribution over all documents. However, if document a priori models are available, they can be integrated directly into the retrieval model by replacing the uniform model used before.

The way document priors are modeled clearly depends on the target application. In web search, for example, the web graph consisting of pages and hyperlinks can be exploited to compute authority measures, which can be used as document priors. Pagerank [13] and HITS [14] are established algorithms to compute authority in web graphs. Another example is search in community portals. Ratings of users, usage patterns, or other evidence can be used to compute document priors [15].

11.3.6 Best Practices for Model Selection

The main criteria for selecting retrieval models is retrieval performance. Both costs of indexing and costs of searching are similar across models. All are based on similar inverted indices and use arithmetic operations in the same complexity class to compute document scores.

When comparing the performance on reference data sets, vector space models, probabilistic models, and language models show no significant difference. The top results on the TEL data set published at Cross Language Evaluation Forum (CLEF) 2009 (see §11.6 for more details on evaluation campaigns) were achieved by retrieval systems based on different models. On English documents, for instance, a vector space model implemented by the University of Chemnitz outperformed a language model of the University of Dublin by only

0.4% in mean average precision.[3] On French documents, a probabilistic model implemented by the University of Karlsruhe outperformed the best vector space model by 1.4% [16]. As can be seen for the results on different languages, these differences are specific to certain data sets, such that it is difficult to decide in general which models perform best.

The choice of retrieval model also depends on the available training data—for example, in the form of relevance assessments of sample queries—and the richness of document models. Without training data, the use of established standard models with standard parameters is recommended, such as BM25 (defined in §11.3.3). This ensures a baseline performance on new data sets. When training data is available, parameters of these models can be optimized. When rich models of documents are available, language models offer the flexibility of integrating this new evidence of relevance. The estimation of probabilities can be adapted to a specific task to improve performance. On top of a well-performing search system, query expansion using PRF can be used to further improve retrieval results in most cases. Query expansion in particular has the goal of increasing recall. However, it should be applied with care if the precision of the system is the main evaluation criteria.

11.4 CLIR

CLIR is the task of retrieving documents relevant to a given query in some language (query language) from a collection of documents in some other language (collection language).

Definition 11–1. Crosslingual information retrieval Given a collection D containing documents in language l_D (collection language), CLIR is the task of retrieving a ranked list of relevant documents for a query in language l_q (query language). Hereby, D is a monolingual collection; that is, all documents in D have the same language.

Essentially, we can distinguish between two different paradigms of CLIR. On the one hand, we have translation-based approaches that translate queries and/or documents into the language supported by the retrieval system. Such approaches reduce the task of crosslanguage retrieval to a standard monolingual IR task to which standard retrieval techniques can be applied. On the other hand, there are also approaches that map both documents and queries into an interlingual (concept) space. The relevance functions are then defined on the basis of this interlingual space.

11.4.1 Translation-Based Approaches

Translation-based approaches translate the query and/or the document collection into some language supported by the retrieval system. Translation-based approaches differ in the choice of translation techniques as well as in the choice of whether only the query, the document collection, or both are translated. We describe several alternative choices for the latter. Further, translations can be obtained either by involving manual translators or through the application of MT techniques.

3. Mean average precision (MAP) is a standard evaluation measure used to evaluate the performance of IR systems. Higher values correspond to more relevant documents being retrieved at lower ranks. A precise definition of MAP is presented in §11.6.

Translating Queries The default strategy for CLIR is the translation of the query into the language of the document collection. This effectively reduces the problem of CLIR to monolingual information retrieval. In what follows, we list some of the advantages (PRO) and disadvantages (CON) of such an approach:

PRO

- Only the query has to be translated, which is usually a short text.
- The index can be used to evaluate queries in arbitrary languages provided they can be translated into the language of the collection/index.

CON

- An online query translation is needed. Because the response time of the retrieval system is the sum of the translation time and the retrieval time, an efficient MT system is needed to maintain system performance at reasonable levels.
- The accuracy of the retrieval system is partially dependent on the quality of the MT system used.

Translating Documents A further strategy is to translate the complete document collection into the query language and create an inverted index for the query language. This might be useful in search scenarios having a fixed query language, such as in portals that have only users of one language. In the following, we also summarize the advantages and disadvantages of such an approach:

PRO

- The translation is part of the preprocessing, as indices will be based on the translated documents. Thus, there is (almost) no temporal constraint on the translation step, such that researchers can resort to manual translation if needed for quality reasons.

CON

- The query language has to be known and fixed in advance. Because the index is specific for this language, queries in other languages are not supported.
- The whole collection has to be translated, which might be costly.

Pivot Language As a combination of the first two approaches, both queries and documents can be translated into a **pivot language**. The pivot language is either a natural or artificial language for which translation systems are available from many languages. English is most often used as such a pivot language because of the large amount of available translation systems. Because no direct translation from query language to document language is needed, the pivot language approach is useful if no language resources supporting this translation are available.

Using a pivot language reduces CLIR to the problem of standard monolingual information retrieval because an existing IR system in the pivot language can be applied to any pairs of query and document languages. However, the performance depends on an adequate

translation for both the query language and the collection language into the pivot language. Advantages and disadvantages here can be summarized as follows:

PRO

- Translation systems to a pivot language can be used for CLIR between languages for which direct translation is not available.
- Existing IR systems in the pivot language can be used for CLIR for any pair of query and document language.

CON

- Online translation of the query as well as offline translation of documents (as part of document preprocessing) is required.

Query Expansion Query expansion techniques can also be applied in CLIR settings in the following ways:

Pretranslation expansion expands the query before it is translated. The expanded query is then processed by the translation system. This has the advantage that more context is given as input to the translation process. In CLIR settings, this was shown to improve precision of the retrieval results [17].

Posttranslation expansion is equivalent to query expansion used in monolingual information retrieval. In a CLIR setting, it has been shown that posttranslation expansion can even alleviate translation errors because wrong translations can be spotted by using local analysis of the results of the query (e.g., using PRF) [17].

11.4.2 Machine Translation

As described earlier a translation step is necessary for translation-based CLIR. Either the query or documents need to be translated before queries can be evaluated using the (language-specific) inverted index. Manual translation, for example, by professional translators, typically incurs high costs. The manual translation of documents does not scale to large corpora, and it is not possible to have real-time translation of queries, a crucial requirement in retrieval systems that need response times in fractions of a second (e.g., in web search). This clearly motivates the use of machine translation for CLIR.

In this chapter we present the two main approaches to machine translation used in CLIR systems—**dictionary-based translation** and **statistical machine translation**.

Dictionary-Based Translation A straightforward approach to query translation is the use of bilingual dictionaries for term-by-term translation. There are different strategies to cope with alternative translations of terms, ranging from choosing the most common translation to taking into account all possible translations. Interestingly, Oard [18] showed that there are no significant differences between the different strategies in a CLIR setting. When using all alternative translations, query terms are usually weighted by their translation probability.

Ballesteros and Croft [17] argue that posttranslation expansion can be used to minimize translation errors in dictionary-based query translation. They showed that using PRF for query expansion removes extraneous terms introduced by the translation and therefore improves the retrieval performance.

Statistical Machine Translation In contrast to dictionary-based translation, statistical machine translation (SMT) aims at translating complete sentences. Thus, in principle, SMT can be applied to both query and document translation. (See Chapter 10 for a full discussion of approaches to SMT.)

Most current SMT systems are based on the IBM models introduced by Brown et al. [19]. These models are iteratively induced for language pairs on the basis of a training corpus in which sentences are aligned across languages. In two subsequent steps, the term alignment of translated sentences and the translation model of terms are optimized. The final model then is a product of the iterative optimization of these two steps. These models can be further improved by additionally translating phrases. In this case not only alignment and translation of single terms but also of phrases like *New York* are learned and applied. Using additional background knowledge can also improve translation, for example, by including language models derived from large monolingual training corpora.

The drawbacks of applying SMT systems to translate the query on the fly is the potentially longer execution time of the retrieval step and the requirement of a training corpus. The execution time bottleneck seems less problematic given the continuous advances in computer hardware and the fact that providers of online translation systems build on a large and distributed computer infrastructure. Indeed, recent systems can already be applied to real-time query translation. However, training corpora are still missing for many language pairs [20].

11.4.3 Interlingual Document Representations

An alternative to translation-based CLIR are interlingual document representations. The essential idea is that both query and documents are mapped to an interlingual **concept space**. In contrast to term-based representations of documents, concepts represent **units of thought** and are thus assumed to be language-independent. Language-specific mapping functions are needed, however, to map documents into the interlingual concept space. Such a mapping might for instance rely on a quantification of the degree of association for terms in different languages (and by aggregation also for a document) to the given set of interlingual concepts. By mapping queries to the same concept space as documents, information retrieval can be reduced to the comparison of query and document concept vectors. This enables the application of standard similarity measures such as the cosine of the angle enclosed by the two vectors representing the query and the document to compute a ranking. In the following, we present two approaches to interlingual concept spaces that have been applied to CLIR.

Latent Semantic Indexing In the monolingual case, latent semantic indexing (LSI) is used to identify latent topics in a text corpus. These topics, which correspond to concepts as described previously, are extracted by exploiting co-occurrences of terms in documents. This is achieved by singular value decomposition of the term-document matrix [21]. The latent topics then correspond to the eigenvectors having the largest singular values. This also results in a mapping function from term vectors to **topic vectors**. LSI was originally used for dimensionality reduction of text representation and for improved retrieval of synonyms or similar terms. By using parallel training corpora, LSI can also been applied to CLIR [22]. In this case the extracted topics span terms of different languages, and the mapping function maps documents of all languages to the latent topic space.

Explicit Semantic Analysis Recently, explicit semantic analysis (ESA) has been proposed as an alternative concept-based retrieval model [23]. Concepts are explicit and defined with respect to some external knowledge source. Textual descriptions of each concept are used to map documents into the concept space. Examples of such resources including concepts and their descriptions that have been used for ESA are Wikipedia and Wiktionary. ESA can be applied to CLIR if textual descriptions of concepts are available in all languages supported by the retrieval system. When using Wikipedia as a multilingual knowledge resource, crosslanguage links can be used to build multilingual concept definitions. Cimiano et al. [24] have shown that ESA can be extended to crosslanguage retrieval settings.

11.4.4 Best Practices

In most cases, query translation is the most flexible way to implement a CLIR system. While supporting arbitrary query languages, the same index can be used for retrieval in any language that can be translated into the language of the collection/index. However, its success depends on the availability of a real-time translation system from query to document language. Given limited resources, translating queries and documents into a pivot language having the best support in terms of translation resources might be the best approach.

Oard showed that SMT systems outperform dictionary-based approaches for query translation. It is therefore recommended to use existing SMT systems, whether commercial or open source. However, if the retrieval system is domain-specific, dictionary-based query translation based on a bilingual dictionary containing all relevant technical terms is likely to outperform generic SMT systems applied to CLIR on this particular domain [18]. If such resources are available for this specific scenario, dictionary-based query translation paired with postretrieval expansion is the best choice for the design of a CLIR system.

11.5 MLIR

In contrast to CLIR, multilingual information retrieval (MLIR) considers corpora containing documents written in different languages. It can be defined as follows:

> **Definition 11–2. Multilingual information retrieval** Given a collection D containing documents in languages $l_1, \ldots, l_n$, MLIR is the task of retrieving a ranked list of relevant documents for a query q in language l_q. These relevant documents may thereby be distributed over all languages $l_1, \ldots, l_n$.

MLIR finds application in all those settings where a data set consists of documents in different languages and users of the retrieval system have at least passive knowledge of some of the languages the documents are written in. In most cases, people have indeed basic reading and understanding skills in some language other than their mother tongue (the one the collection is queried with). Such settings can be found in particular on the web but also in large corporate multinationals. Further, still if the users do not understand the language of a returned document, MT techniques can be applied to produce a text in the native language of the user.

In general, MLIR systems are based on techniques similar to those used for CLIR systems, and essentially the same translation approaches can be applied. However, the multilingual scenario requires a different index organization and relevance computation strategies than are used in monolingual and crosslingual retrieval. In the following, we briefly describe different strategies ranging from unified indices to multiple language-specific indices. If the language of the documents is not known a priori, language identification is required as part of the preprocessing.

11.5.1 Language Identification

Language identification is the problem of labeling documents with the language in which the content of the document is expressed. In the following, we assume that documents are monolingual; that is, they contain text in one language. The more complex case of mixed documents is briefly touched upon at the end of this section.

The problem of language identification can be reduced to a standard classification problem with discrete classes. The target classes are given by a set of languages, and the task is to classify documents to one of these classes representing each of the relevant languages. Given monolingual training corpora for each language, supervised machine learning approaches can be applied to this task. The most successful reported classification method is based on character n-gram representations of documents. Cavnar and Trenkle [25] present language identification results with 99% precision using a set of 14 languages. They build term vectors for each document by extracting character n-grams for $n = 1 \ldots 5$. An important aspect here is that the classifiers can be trained on monolingual input for each of the languages so that an aligned data set is not necessary. Thus, the approach is applicable in principle to any set of languages. Further, the method requires no preprocessing because character n-grams are based on the character streams without word splitting. It has been demonstrated that the accuracy of this language identification method is dependent on document length because longer documents provide more evidence for the language they are written in. The results of Cavnar and Trenkle show that precision of 99% or more can be expected for documents having more than 300 characters.

Applying the proposed classifier on mixed documents having content in multiple languages results in unpredictable classification. The language-specific distribution of terms or n-grams (that is exploited by the classifier) is lost as the characteristics of the individual languages overlay each other. These documents have to be split into their monolingual components beforehand. As the splitting is done on sections, paragraphs, or sentences, this produces shorter documents that are more difficult to classify, thus degrading results.

11.5.2 Index Construction for MLIR

There are two main approaches to index construction for MLIR, differing in whether a single or multiple indices are used. Single-index approaches build one index for the documents in the different languages. We distinguish three techniques for constructing such an index:

Document translation: By translating all documents into a pivot language, the problem of MLIR can be reduced to CLIR. The single index then contains all the translated documents.

Language token prefixes: Nie [26] proposes the creation of a unified index by adding language prefixes to all tokens. This ensures that terms having the same character representation in different languages can be distinguished. The lexicon of the unified index consists of terms in all languages. Nie argues that this unified index preserves term distribution measures like term frequency and document length.

Concept index: As discussed earlier, language-independent concept indices can also be applied to MLIR. As documents of different languages are mapped to the same interlingual concept space, only a single concept index is needed for the multilingual corpus.

Approaches based on multiple indices build different indices for each language of the corpus. There are two different techniques:

Language-specific indices: Each document in a multilingual collection is added to the index for the corresponding language, whereby the language needs to be identified such that the language-specific preprocessing can be applied. For monolingual documents, the set of documents contained in each index are thus disjointed. For mixed documents having content in different languages, only document parts in the language of the index are added. In this case a document can appear in many of the language-specific indices.

Specific preprocessing: For each language, an index is constructed containing all documents of the corpus. However, preprocessing of these documents is specific to the language associated to each index. For each index, documents are therefore assumed to be in the index language. As a consequence, each document is contained in all indices.

11.5.3 Query Translation

The different approaches to index construction require different query translation strategies. For single indices based on document translation or concept indices, refer to Section 11.4 because the query translation is analogous to query translation used in CLIR.

For all other approaches, queries need to be translated into all document languages. Depending on the index used, these translations are applied in different ways:

Language token prefixes: A query for the unified index with language prefixes for each term is built by concatenation of all query translations into a single query and by adding language prefixes to all query tokens. Standard IR models can then be used to query the unified index.

Multiple indices: When using multiple indices for the different languages, the translation of the query into each language is used to query the index of the corresponding language. This produces different language-specific rankings for each language, and the rankings must be combined into an aggregated score determining an aggregated ranking. We discuss some of the most important aggregation models next.

11.5.4 Aggregation Models

Retrieval based on multiple indices requires score aggregation models because the rankings based on evidence in each language need to be combined to yield a final ranking. Given a set of languages $L = \{l_1, \ldots, l_n\}$, a query q, and language-specific scores for each document—$\text{score}_l(d, q)$—a straightforward approach is to sum up the scores for all the languages:

$$\text{score}(q, d) = \sum_{l \in L} \text{score}_l(q, d)$$

The aggregated score can then be used to sort all documents and produce an overall ranking of the documents.

The main problem of this aggregation strategy is the potential incompatibility of the scores. In fact, by simply adding scores, it is assumed that the absolute score values express the same relevance level in each ranking. However, for most retrieval models, this is not the case. The absolute values of scores depend on collection statistics and term weights, such as the number of documents, number of tokens, average document length, or document frequency. For each index, these values differ, and therefore the absolute scores are not necessarily comparable. To overcome this problem, normalization typically is applied to each ranking before aggregation. A standard approach for MLIR is the Z-score normalization [27]. Each ranking is normalized by using statistical measures on its scores: the minimal score, the mean score, and the standard deviation. Given training data in the form of queries and relevance judgments for documents, machine learning techniques can be used to compute optimal weights by which scores can be combined (see Croft [28]).

A complete aggregation step using Z-score normalization is presented in Algorithm 11–2. Given a set of rankings $R = \{r_1, \ldots, r_n\}$, the algorithm computes the combined ranking r_c. In the first step, each ranking r_i is normalized using the minimum value, the mean value, and the standard deviation of its values. In the second step, combined scores are computed by summing the scores of each document across all rankings. Finally, the combined ranking is built by reordering documents according to descending values of the aggregated scores.

11.5.5 Best Practices

By analogy to CLIR, indexing documents in their original language and using translated queries for retrieval is the most flexible approach to MLIR because it directly supports new query languages and can usually be adopted to new retrieval and aggregation models. An interesting feature is that if the translation system is changed or updated, there is no need to rebuild the indices. An analysis of the results of recent evaluation campaigns for MLIR—the results of the CLEF workshop 2009 [16]—reveals that the most successful systems are based on multiple language-specific indices. SMT systems are thereby used for query translation, and score aggregation is based on Z-score normalization [29].

Using a unified index with language prefixes is a good option for MLIR systems that are built on top of existing IR systems. Because the indexing and retrieval steps remain unaffected, only the preprocessing of documents (adding language prefixes on tokens) and queries (translation and language prefixes) needs to be adjusted.

Algorithm 11–2 Aggregation of multiple rankings $r_1, \ldots, r_n$ based on Z-score normalization. For ranking r, $r[i]$ defines the score at rank position i, $\text{score}_r(d)$ defines the score of document d. MIN, MEAN, and STD-DEVIATION are defined on the set of score values of ranking r

```
R ← {r_1, ..., r_n}
for all r ∈ R do                          // Normalization
  μ ← MEAN(r)
  σ ← STD-DEVIATION(r)
  δ ← (μ − MIN(r)) / σ
  for i = 1..|r| do
    r(i) ← (r(i) − μ) / σ + δ
  end for
end for

r_c ← {}
for all d ∈ D do                          // Aggregation
  s ← 0
  for all r ∈ R do
    s ← s + score_r(d)
  end for
  score_{r_c}(d) ← s
end for
r_c ← DESCENDING-SORT(r_c)
return r_c
```

11.6 Evaluation in Information Retrieval

Ultimately, the goal of any IR system is to satisfy the information needs of its users. Needless to say, user satisfaction is very hard to quantify. Thus, IR systems are typically evaluated building on the notion of **relevance**, where relevance is assessed by a team performing the evaluation of the system rather than by the final. We can adopt a binary notion of relevance, where a document is either relevant to a query or it isn't, or a real-valued notion of relevance, where we consider the degree of relevance of a document to a query. The former case is the most frequent one in IR evaluation. Given a specification of which documents are relevant to a certain query and which ones are not, the goal of any IR system is to maximize the number of relevant documents returned while minimizing the amount of nonrelevant documents returned. If the IR system produces a ranking of documents, then the goal is clearly to place relevant documents on top and nonrelevant ones on the bottom of the ranked list. Several evaluation measures have been proposed to capture these intuitions. In addition, several reference collections with manual relevance judgments have been developed over the years. Because results on such data sets are thus reproducible, they allow different system

developers to compete with each other and support the process of finding out which retrieval models, preprocessing, indexing strategies and so on perform best on certain tasks.

In this section we first describe the experimental setup that was introduced as the **Cranfield paradigm**. Then we introduce and motivate different evaluation measures. These measures are based on relevance assessments. We describe manual and automatic approaches to create relevance assessments. Finally, we provide an overview of established datasets that can be used to evaluate CLIR and MLIR systems.

11.6.1 Experimental Setup

The experimental setup used to evaluate IR systems has to ensure that an experiment is reproducible, which is the primary motivation for the development of the Cranfield evaluation paradigm [30]. According to this paradigm, we have to fix a certain corpus as well as a minimum number of **topics** consisting of a textual description of the information need as well as a query to be used as input for the IR system. The systems under evaluation are expected to index the collection and return (ranked) results for each topic (query). To reduce the bias toward outliers, a reasonable number of topics must be used in order to yield statistically stable results. A number of at least 50 topics is typically recommended.

For each topic, a gold standard defines the set of relevant documents in the collection [2]. The notion of relevance is hereby typically binary—a document is relevant or not to a given query. Using this gold standard, the IR system can then be evaluated by examining whether the returned documents are relevant to the topic and whether all relevant documents are retrieved. These notions can be quantified by certain evaluation measures, which are supposed to be maximized (see §11.6.3). Because user satisfaction is typically difficult to quantify and such experiments are difficult to reproduce, an evaluation using a gold standard—with defined topics and given relevance assessments—is an interesting and often adopted strategy.

11.6.2 Relevance Assessments

Experimentation in information retrieval usually requires **relevance assessments** that are used to create the gold standard. Whereas for smaller collections, such as the original Cranfield corpus, the manual examination of all the documents for each topic by assessors is possible, it is unfeasible for larger document collections [2]. Thus, a technique known as **result pooling** is used to avoid the need for assessors to scan the whole document collection for each topic. The idea is that the top-ranked documents are pooled from a number of IR systems to be evaluated. For each topic, the top k documents retrieved by different systems, where k is usually 100 or 1,000, are typically considered. As relevance assessments vary between assessors, each document/topic pair is usually judged by several assessors. The final relevance decision as contained in the gold standard is an aggregated value, for example, based on majority votes. The measurement of the interannotator agreement, for example, through the kappa statistic [2], is an indicator for the validity of an experiment. A low agreement might, for instance, result from the ambiguous definition of information needs.

By involving several systems in the pooling, the researcher tries to reduce the bias of the relevance judgments toward any single system. Moreover, the test collection should be

sufficiently complete so that the relevance assessments can be reused to test IR techniques or systems that were not present in the initial pool.

For CLIR or MLIR systems, an alternative evaluation method is provided by the **mate retrieval** setup. This setup avoids the need to provide relevance judgments by using a parallel or aligned data set consisting of documents and their translation into all relevant languages. The topics used for evaluation correspond to a set of documents in the corpus. The **mates** of this topic—the equivalents of the document in different languages—are regarded as the only relevant documents, such that the goal of any system is to retrieve exactly these mates. The gold standard can therefore be constructed automatically. The values of evaluation measures are clearly underestimated using this gold standard, as other documents might be relevant as well.

11.6.3 Evaluation Measures

To quantify the performance of an IR system on a certain data set with respect to a given gold standard consisting of relevance judgments, we require the definition of certain evaluation metrics. The most commonly used evaluation measures in information retrieval are **precision** and **recall**. Precision measures the percentage of the retrieved documents that are actually relevant, and recall measures the percentage of the relevant documents that are actually retrieved.

Computation of these measures can explained on the basis of the contingency table of retrieval results for a single query, as presented in Table 11–1. Precision P and recall R are then defined as:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

To choose an appropriate evaluation metric, it is crucial to understand how the retrieval system will be used. In some scenarios, users are likely to read all documents, for example, to compile a report, while in other scenarios, such as ad hoc search in the web, users are likely to only examine the top-ranked documents. It should be clear from these very extreme examples of usage that the choice of an evaluation metric is not independent of the way the IR system is supposed to be used.

For retrieval systems in which both precision and coverage (i.e., returning all relevant documents) is important—which is not necessarily the case for web search—a reasonable choice is average precision (AP), which averages the precision at certain positions in the ranking. In particular, these are the positions at which relevant documents are found.

For ranking r of n documents, the set of relevant documents REL, the binary function $rel : D \rightarrow \{0, 1\}$ mapping relevant documents to 1 and nonrelevant to 0 and P_k as precision

Table 11–1: Contingency table of retrieval results for a single query

	relevant	*nonrelevant*
retrieved	TP	FP
nonretrieved	FN	TN

at cutoff level k, AP is computed as follows:

$$AP(r) = \frac{\sum_{i=1}^{n} P_i \times rel(r_i)}{|REL|}$$

Mean average precision (MAP) averages AP over all topics and can be used to evaluate the overall performance of an IR system.

A common feature of measures such as MAP—others are bpref [31] and infAP [32]—is that they are primarily focused on measuring retrieval performance over the entire set of retrieved documents for each query, up to a predetermined maximum (usually 1,000). As already mentioned, such evaluation measures are a reasonable choice for scenarios in which users require as many relevant documents as possible. However, it is likely that users will not read all 1,000 retrieved documents provided by a given IR system. For this reason, other measures have been proposed to assess the correctness of the retrieval system given that users typically examine only a limited set of (top-ranked) documents. For example, precision can be calculated at a given rank (denoted P@r). Precision @ rank 10 (P@10) is commonly used to measure the accuracy of the top-retrieved documents. Where there is an importance to get the top-ranked document correct, mean reciprocal rank of the first relevant document is often used.

In some cases, relevance assessments contain multiple levels of relevance in combination with a measure such as normalized discounting cumulative gain (NDCG) [33], which takes into account the preference to have highly relevant documents ranked above less relevant ones.

11.6.4 Established Data Sets

IR experiments become reproducible and results comparable by reusing shared data sets consisting of a common corpus of documents, topics/queries and relevance assessments. In the field of information retrieval, different evaluation initiatives defining various retrieval tasks and providing appropriate data sets have emerged.

Apart from data sets published by evaluation campaigns, parallel corpora are also of high interest to CLIR and MLIR. They are used as language resources, for example, to train SMT systems or to identify crosslanguage latent concepts as in LSI. Additionally, they are used as test collection, for example, in mate retrieval scenarios.

Evaluation Campaigns

Text REtrieval Conference (TREC) is organized yearly with the goal of providing a forum where IR systems can be systematically evaluated and compared. TREC is organized around different tracks (representing different IR tasks such as ad hoc search, entity search, or search in special domains). For each track, data sets and topics/queries (and relevance judgments) are typically provided that participants can use to develop and tune their systems. Since its inception in 1992, TREC has been applying the pooling technique that allows for a cross-comparison of IR systems using incomplete

assessments for test collections. TREC and similar conferences are organized in a competitive spirit in the sense that different groups can compete with their systems on a shared task and data set, thus making results comparable. Such shared evaluations have indeed contributed substantially to scientific progress in terms of understanding which retrieval models, weighting methods, and so on, work better compared to others on a certain task. However, the main goal of TREC is not only to foster competition but also to provide shared data sets to the community as a basis for systematic, comparable, and reproducible results. The main focus of TREC is monolingual retrieval of English documents such that the published data sets consist only of English topics and documents.

Crosslingual Evaluation Forum (CLEF) was established as the European counterpart of TREC with a strong focus on multilingual retrieval. In the ad hoc retrieval track, different data sets have been used between 2000 and 2009, such as a large collection of European newspapers and news agency documents with documents in 14 languages, the TEL data set containing bibliographic entries of the European Library in English, and French, German, and Persian newspaper corpora. For all data sets, topics in different languages are available, which makes these data sets suitable for CLIR and MLIR. The TEL data set also contains mixed documents with fields in different languages.

NII Test Collection for IR Systems (NTCIR) defines a series of evaluation workshops that organize retrieval campaigns for Asian languages, including Japanese, Chinese, and Korean. A data set of scientific abstracts in Japanese and English as well as news articles in Chinese, Korean, Japanese, and English with topics in different languages has been released. In addition, a data set for Japanese-English patent retrieval has been published.

Forum for Information Retrieval Evaluation (FIRE) is dedicated to Indian languages. It has released corpora built from web discussion forums and mailing lists in Bengali, English, Hindi, and Marathi. Topics are provided in Bengali, English, Hindi, Marathi, Tamil, Telugu, and Gujarati.

Parallel Corpora

JRC-Acquis is a document collection extracted from the Acquis Communautaire, the total body of European Union law that is applicable in all EU member states. It consists of parallel texts in the following 22 languages: Bulgarian, Czech, Danish, German, Greek, English, Spanish, Estonian, Finnish, French, Hungarian, Italian, Lithuanian, Latvian, Maltese, Dutch, Polish, Portuguese, Romanian, Slovak, Slovene, and Swedish.
http://langtech.jrc.it/JRC-Acquis.html

Multext Dataset is a document collection derived from the Official Journal of European Community in the following five languages: English, German, Italian, Spanish, and French.
http://aune.lpl.univ-aix.fr/projects/multext/

Canadian Hansards consists of pairs of aligned text chunks (sentences or smaller fragments) from the official records (Hansards) of the 36th Canadian Parliament in English and French.
http://www.isi.edu/natural-language/download/hansard/

Europarl is a parallel corpus containing the proceedings of the European Parliament from 1996 to 2009 in the following languages: Danish, German, Greek, English, Spanish, Finnish, French, Italian, Dutch, Portuguese, and Swedish.
http://www.statmt.org/europarl/

11.6.5 Best Practices

We have clearly argued that the ideal evaluation method and metric for CLIR and MLIR systems are dependent on the way the system will be used.

If the system is designed for research purposes and the goal is to advance the state-of-the-art in information retrieval by analyzing a specific research question, using established data sets and standard evaluation measures is highly advised to ensure that results are comparable to existing systems. In many cases existing gold standards can also be used to compute evaluation measures.

If the retrieval system is part of an application involving real users, the data set is usually defined by the task. To evaluate the system, topics covering the expected information needs of users have to be defined, and relevance assessments are required in order to construct an appropriate gold standard. As described previously, pooling techniques help to reduce the effort to come up with a gold standard of relevance judgments.

In all cases, standard evaluation measures like MAP or mean reciprocal rank are the preferred measures to assess retrieval performance. For specific scenarios, different measures can be used to focus on specific aspects of the desired outcome, for example, if high precision at low ranks is required.

11.7 Tools, Software, and Resources

The development of a complete IR system includes many different aspects, such as the implementation of preprocessing steps, file structures for inverted indices, and efficient retrieval algorithms. Building a system from scratch therefore constitutes an enormous effort. It is essential to build on existing tools to reduce the costs related to implementation.

In a specific project, it might be the case that only the retrieval model or ranking function needs to be adapted, while the other components of the system can be used off-the-shelf. Fortunately, several libraries provide standard IR components or even complete frameworks where certain components can be replaced.

Following are selected tools and software libraries supporting the development of IR systems. We focus on established tools that are widely used and also have community support. The most popular IR framework is Lucene, which also contains wrappers for many other tools we present.

Preprocessing

Content Analysis Toolkit (Tika) is a toolkit to extract text from documents of various file types, such as PDF or DOC, implemented in Java. The detection of file types is also supported. Tika evolved from the Lucene project.
http://lucene.apache.org/tika/

Snowball Stemmer is a stemmer for several European languages. The implementation is very fast and also supports stop-word removal. Lists of stop-words for the supported languages are provided on the project website.
http://snowball.tartarus.org

HTML Parser is a tool for parsing HTML documents. It can be used to extract textual content from websites, ignoring tags and parts not related to the semantic content.
http://htmlparser.sourceforge.net/

BananaSplit is a compound splitter for German based on dictionary resources.
http://www.drni.de/niels/s9y/pages/bananasplit.html

Translation The web portal http://www.statmt.org is an excellent entry point to get information about statistical machine translation systems. It provides software and data sets to train translation models.

As an example of a commercial SMT system, the Google Translate Service[4] provides an API for translation into various languages. However, because translation is part of preprocessing and is usually not deeply integrated into the retrieval framework, any commercial translation system might be plugged into a CLIR or MLIR system.

IR Frameworks

Lucene is a widely used IR framework implemented in Java. It is available as open source software under the Apache License and can therefore be used in both commercial and open source programs. It has reached a mature development status and is used in various applications. The main features of Lucene are scalability and reliability, which come at the price of a decreased flexibility, making it more difficult to exchange components. For instance, in Lucene the index construction is dependent on the retrieval model selected, so the retrieval model cannot be exchanged without rebuilding the index.
http://lucene.apache.org

Terrier and Lemur are tools used for research purposes. Terrier (implemented in Java) and Lemur (implemented in C++) are both flexible IR frameworks that can be easily extended and modified. Because of their different focus, they do not match the stability and performance of Lucene.
http://terrier.org
http://www.lemurproject.org

4. http://translate.google.com/

Evaluation

trec_eval (evolved from TREC) is a tool that can be used to compute various evaluation measures for a given document ranking with respect to a gold standard. The input is expected as plain text files with simple syntax. Creating output in the TREC format enables to use trec_eval for any IR system. The IR frameworks presented earlier also support output in the TREC format.
http://trec.nist.gov/trec_eval/

11.8 Summary

In this chapter we provided an overview of methods that can be used to implement IR systems to access documents in different languages. We distinguished two flavors of this problem: crosslingual and multilingual information retrieval. CLIR retrieves documents in one language given topics in another language, and MLIR can be applied to multilingual document collections with topics in different languages.

We discussed what types of preprocessing (tokenization, stemming, etc.) are required depending on the languages supported. We further discussed foundational approaches to information retrieval that can be reused as building blocks when developing CLIR and MLIR systems. In particular, we showed that most of the standard document models used in information retrieval can be reused in CLIR and MLIR systems. We discussed two main approaches to CLIR and MLIR: translation-based approaches and interlingual representation-based approaches. In this context, we also discussed different machine translation techniques and how they can be applied in CLIR and MLIR settings. Advances in statistical machine translation have made it possible to reduce CLIR and MLIR to a standard monolingual retrieval task by supporting the translation of queries into several languages in real time. We also discussed approaches to identify the language of a document, a crucial step if several language-specific indices are constructed and maintained. We briefly touched upon the problem of aggregating different scores obtained from different language-specific indices as a way to yield an overall score and ranking. This is a nontrivial problem in multilingual retrieval.

Because evaluation is crucial for the field of information retrieval, we examined how IR systems are typically evaluated. In particular, we presented methodologies to obtain relevance judgments, either manually or automatically, and introduced standard IR evaluation measures. Finally, we provided an overview of standard data sets, evaluation campaigns, as well as software libraries and general resources.

Acknowledgments

This work was funded by the German Research Foundation (DFG) under the Multipla project (grant 38457858) and by the European Commission under the Monnet Project (grant FP7-ICT-4-248458).

[29] J. Krsten, "Chemnitz at CLEF 2009 Ad-Hoc TEL task: Combining different retrieval models and addressing the multilinguality," in *Working Notes of the Annual CLEF Meeting*, 2009.

[30] C. Cleverdon, "The Cranfield tests on index language devices," *Aslib Proceedings*, vol. 19, no. 6, pp. 173–194, 1967.

[31] C. Buckley and E. M. Voorhees, "Retrieval evaluation with incomplete information," in *Proceedings of the 27th International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 25–32, 2004.

[32] E. Yilmaz and J. A. Aslam, "Estimating average precision with incomplete and imperfect judgments," in *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, p. 111, 2006.

[33] K. Jrvelin and J. Keklinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 41–48, 2000.

Chapter 12

Multilingual Automatic Summarization

Frank Schilder and Liang Zhou

12.1 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 [1].

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]

In 2002, the EU-sponsored project MLIS-MUSI (Multilingual Summarization for the Internet) offered multilingual summarization for English and Italian scientific articles [8].

A couple of years later, the NewsBlaster summarization system was developed by Columbia University enabling users to browse news written in multiple languages from multiple sites on the Internet [9].[3]

In 2005, the Multilingual Summarization Evaluation (MSE) project at the Linguistics Data Consortium (LDC) advanced the research yet further.[4] The 25 news topics used for this evaluation were derived from the output of Columbia's NewsBlaster topic clustering system. The summarization task captured English and Arabic news messages. However, annotators generated their 100-word summaries from the English news documents, but only from the English translations of the Arabic articles and not directly from the Arabic source documents.

Another milestone in multilingual research was reached when the GATE-based summarizer SUMMA by Horacio Saggion was released in 2006 [10, 11]. Given the open architecture of the GATE system [12], it was straightforward to integrate language-specific tools (e.g., tokenizers, sentence splitters) into the SUMMA system supporting also summaries in Latvian, Swedish, and Finnish.[5]

Recent papers that address specific problems of multilingual summarization or offer approaches to trans- or crosslingual summarization systems include work by Mani, Yeh, and Condon, who describe a system that finds names across different languages and show that it can match names from English to Chinese with an F-measure of up to 97.0 [14]. Another summarization problem that requires processing multiple languages is described by Mille and Wanner [15], who propose a system that processes patents in different languages.

Systems that allow users to summarize text in a language other than English have been proposed in recent years. Leuski et al. [16] describe a system that translates English headlines into Hindi. Orăsan and Chiorean [17] utilize maximal marginal relevance (MMR) [18] for summarizing Romanian news messages.

12.2 Approaches to Summarization

12.2.1 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

2. http://www.summarization.com/summbank/
3. The website http://newsblaster.cs.columbia.edu summarizes news (and images) from (English) news websites.
4. http://projects.ldc.upenn.edu/MSE/
5. These language resources were developed in the context of the Clarity project [13].

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 multi-document 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.

12.2.2 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]]

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

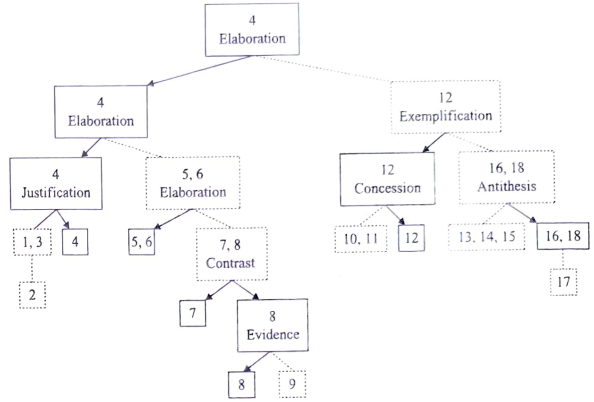

Figure 12–1: The RST structure of the Mars example text (Reproduced from Marcu [31])

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]

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

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].

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]

$$S(V_i) = (1 - d) + d * \sum_{j \in in(V_i)} \frac{1}{|out(V_j)|} S(V_j) \tag{12.1}$$

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.

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \wedge w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)} \tag{12.2}$$

The weighted PageRank score is defined as follows:

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in in(V_i)} \frac{1}{\sum_{V_k \in out(V_j)} w_{jk}} WS(V_j) \tag{12.3}$$

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.

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

[7] The parameter d is normally set to 0.85.

Table 12–1: A sample newspaper article used as input for TextRank. The output graph is shown in Figure 12–2

4:	BC–Hurricane Gilbert, 0348
3:	BC–Hurricane Gilbert, 0–11 339
5:	Hurricane Gilbert heads toward Dominican Coast
6:	By Ruddy Gonzalez
7:	Associated Press Writer
8:	Santo Domingo, Dominican Republic (AP)
9:	Hurricane Gilbert Swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
10:	The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
11:	"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
12:	Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
13:	An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
14:	Tropical storm Gilbert formed in the eastern Carribean and strenghtened into a hurricane Saturday night.
15:	The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
16:	The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
17:	The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
18:	Strong winds associated with Gilbert brought coastal flooding, strong southeast winds, and waves up to 12 feet to Puerto Rico's south coast.
19:	There were no reports on casualties.
20:	San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
21:	On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
22:	Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
23:	Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
24:	The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

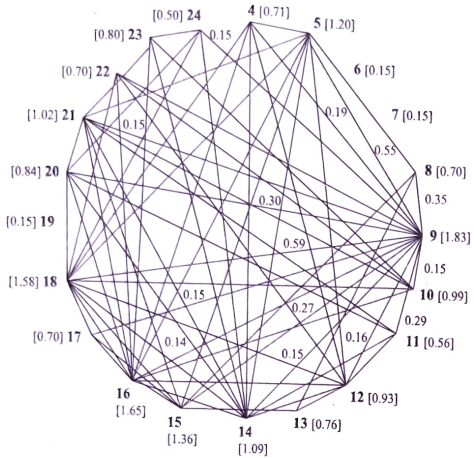

Figure 12–2: A sample graph generated from a text (Reproduced from Mihalcea and Tarau [19])

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.

12.2.3 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 $d \in D$.

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

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 $\sum_{i=1}^{n} \beta_i (s_i - s'_i)$:

$$L_{exp}(D, B) = \frac{1}{\mid D \mid} \sum_{d \in D} \frac{1}{\mid S_d^{pos} \mid\mid S_d^{neg} \mid} \sum_{(s,s') \in S_d^{pos} \times S_d^{neg}} e^{\sum_{i=1}^{n} \beta_i (s'_i - s_i)} \tag{12.5}$$

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:

$$L_{exp}(D, B) = \frac{1}{\mid D \mid} \sum_{d \in D} \frac{1}{\mid S_d^{pos} \mid\mid S_d^{neg} \mid} \sum_{s' \in S_d^{neg}} e^{\sum_{i=1}^{n} \beta_i s'_i} \sum_{s \in S_d^{pos}} e^{\sum_{i=1}^{n} \beta_i s_i} \tag{12.6}$$

Algorithm 12–1 Pseudocode for ranking-based trainable extractive summarizer: LinearRank

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 $F = (\beta_1 ... \beta_n)$ 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):

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

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 $\mathcal{T}$ to the total number of words $t_{1..|s|}$ in the sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{T}}(t_i)}{|s|}$, where

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

Topic description frequency: ratio of number of words t_i in the sentence s that also appear in the topic description $\mathcal{D}$ to the total number of words $t_{1..|s|}$ in the sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{D}}(t_i)}{|s|}$, where $f_{\mathcal{D}} = \begin{cases} 1 & : \ t_i \in \mathcal{D} \\ 0 & : \ otherwise \end{cases}$

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.

12.2.4 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.

8. http://www.summarization.com/mead/

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:

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

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.

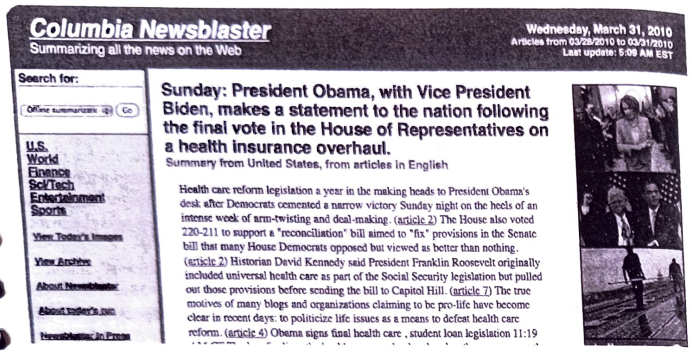

Figure 12–3: A sample page generated by NewsBlaster (Courtesy Columbia University)

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

12.3 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 gold-standard 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.

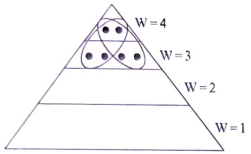

Figure 12–4: A pyramid of four levels (Reproduced from Nenkova and Passonneau [49])

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 York-bound 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.

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:

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

- *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:

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

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

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

$\text{LCS}(X,Y)$ 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. Skip-bigram-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).

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.

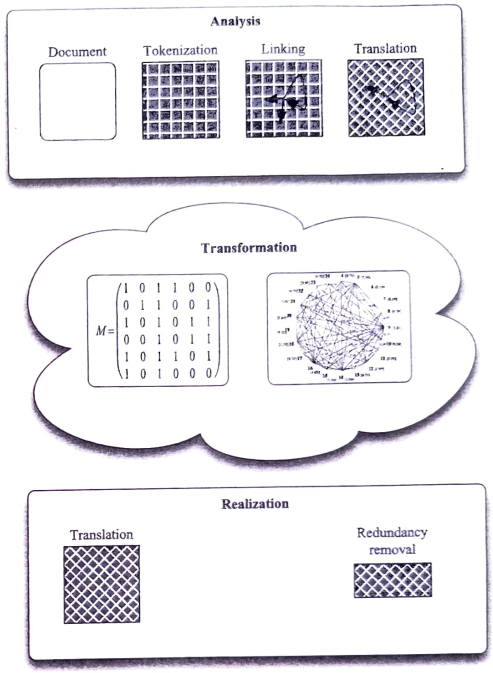

Figure 12–5: A blueprint for a multilingual summarization system

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.

10. http://www.nist.gov/tac/data/index.html
11. http://www.ldc.upenn.edu/
12. http://sites.google.com/site/danlp2010/home
13. http://uima.apache.org/
14. http://gate.ac.uk
15. http://www.nltk.org/
16. http://www.r-project.org/
17. http://cran.r-project.org/web/views/MachineLearning.html
18. http://igraph.sourceforge.net/

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:

Giza++ http://fjoch.com/GIZA++.html

Thot http://sourceforge.net/projects/thot/

Moses http://www.statmt.org/moses/

Joshua http://www.cs.jhu.edu/~ccb/joshua/index.html

Another way of integrating machine learning is via the Google translation API: http://code.google.com/p/google-api-translate-java/

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].

12.4.3 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).

19. http://code.google.com/p/cleartk/

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:
  http://clair.si.umich.edu/clair/CSTBank/
  http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T16
- Document Understanding Conference (English, Arabic [DUC 2003])[20]
- Text Analysis Conference (English)[21]
- 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
  (http://clair.si.umich.edu/clair/CSTBank/)
- 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.
  http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19
- 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.
  http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2009T10
- 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

20. http://www-nlpir.nist.gov/projects/duc/data.html
21. http://www.nist.gov/tac/data/index.html

Mandarin Chinese) over a period of six months (January through June 1998).
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T57
TDT 3 contains data from the same nine 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).
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T58
Finally, TDT4 contains the complete set of English, Arabic, and Chinese news text (broadcast news transcripts, and newswire data) used in the 2002 and 2003 Topic Detection and Tracking technology evaluations.
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T16

12.6 Summary

In this chapter we presented the main approaches to summarization and showed how they need to be extended in order to work in a multilingual environment. Multilingual summarization presents an additional complexity to automatic summarization systems designed for one source/target language.

After providing a brief history to multilingual summarization, we gave an overview to the major approaches to summarization. Most monolingual summarization systems are divided into three stages: analysis, transformation, and realization. The same is true for multilingual systems.

1. For the analysis stage, summarization systems may represent the text in the form of a graph. This may be a linguistically motivated discourse tree or a matrix representation based on sentence-to-sentence similarity.
2. The transformation process can be carried out via graph-based algorithms such as PageRank or by machine learning–based classifiers that learn to classify sentences according to their relevancy.
3. Multilingual approaches have to face many language-dependent challenges such as tokenization, anaphoric expressions, and discourse structure for the realization of the summary.

Many natural language processing research areas can be encompassed into the summarization process, such as language modeling and understanding, coreference resolution, anaphora resolution, and surface realization. Each of these tasks elicits rich problems and solution representations, adding complexity and variability to summarization problem solutions.

This complexity in turn complicates the evaluation process in terms of task definition and solution comparison. As part of the continuing research effort, many manual and automated evaluation methodologies have been created to suit various task needs, such as query-based and single- and multidocument summarizations. We discussed all major approaches to evaluation. The approaches range from manual annotation to automatically generated metrics

that are sometimes language-independent but also require language-specific resources such as parsers.

We then discussed the different concrete ingredients that can be used for building a summarization system. Many natural language processing tools already are available, although perhaps not for all languages. Machine translation systems may be able to bridge some of the gaps, but often new resources need to be developed in order to build a summarization system in not-well-researched languages.

Finally, we compiled a list of data sets that can be used to develop and train your own summarization system in languages other than English.

Bibliography

[1] S. Wan and C. Paris, "In-browser summarisation: Generating elaborative summaries biased towards the reading context," in *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pp. 129–132, 2008.

[2] U. Hahn and I. Mani, "The challenges of automatic summarization," *Computer*, vol. 33, no. 11, pp. 29–36, 2000.

[3] K. Knight and D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression," *Artificial Intelligence*, vol. 139, no. 1, pp. 91–107, 2002.

[4] R. Mitkov, "Robust pronoun resolution with limited knowledge," in *Proceedings of the 18th International Conference on Computational Linguistics (COLING'98)/ACL'98 Conference*, pp. 869–875, 1998.

[5] G. Erkan and D. R. Radev, "LexPageRank: Prestige in Multi-Document Text Summarization," in *Proceeding of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2004.

[6] E. Hovy and C.-Y. Lin, "Automated text summarization and the SUMMARIST system," in *Advances in Automated Text Summarization* (I. Mani and M. Maybury, eds.), Cambridge, MA: MIT Press, 1998.

[7] D. R. Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, H. Qi, A. Çelebi, D. Liu, and E. Drabek, "Evaluation challenges in large-scale multi-document summarization: the mead project," in *Proceedings of the Association for Computational Linguistics 2003*, 2003.

[8] A. Lenci, R. Bartolini, N. Calzolari, A. Agua, S. Busemann, E. Cartier, K. Chevreau, and J. Coch, "Multilingual summarization by integrating linguistic resources in the MLIS-MUSI project," in *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'02)*, 2002.