2. Smart vs. Solid Solutions

For computational applications, two approaches may be generally distinguished: smart solutions and solid solutions. Smart solutions use special properties of the particular application to obtain a sufficient result at a minimum of cost. Solid solutions, in contrast, analyze the application at a more general level such that they can be used for a whole class of applications. By maintaining and extending a solid solution, it may be re-used again and again, thus saving time and money in the long run.

For computational linguistics, the most general solid solution is a general theory of how natural language communication works. To give an idea of the kinds of applications such a solid solution may be used for, Sect. 2.1 explains the structures underlying the use of textual databases. Section 2.2 shows how linguistic methods can improve the retrieval from textual databases. Section 2.3 shows how different applications require linguistic knowledge to different degrees in order to be practically useful. Section 2.4 explains the notion of language pairs in machine translation and describes the *direct* and the *transfer* approaches. A third approach to machine translation, the *interlingua* approach, as well as computer-based systems for aiding translation, are described in Sect. 2.5.

2.1 Indexing and Retrieval in Textual Databases

A textual database is an arbitrary collection of electronically stored texts. In contrast to a classic, record-based database like the one in 1.1.1, no structural restrictions apply to a textual database. Thus, the individual texts may be arranged, for example, in the temporal order of their arrival or according to their subject matter, the name of their author(s), their length, or no principle at all.

The search for a certain text or text passage is based on the standard, letter-based indexing of the textual database.

2.1.1 Standard Indexing of a Textual Database (Inverted File)

The indexing of a textual database is based on a table, called inverted file, which specifies for each letter all the positions (place numbers) where it occurs in the storage medium of the database.

R. Hausser, *Foundations of Computational Linguistics*,
DOI 10.1007/978-3-642-41431-2_2, © Springer-Verlag Berlin Heidelberg 2014

The electronic index (inverted file) of a textual database functions in many ways like a traditional library catalog of alphabetically ordered file cards.

Each traditional file card contains a *keyword*, e.g., the name of the author, and the associated *address*, e.g., the shelf where the book of the author may be found. While the file cards are ordered alphabetically according to their respective keywords, the choice of the addresses is initially free. Once a given book has been assigned a certain address and this address has been noted in the catalog, however, it is bound to this address.

In an unordered library without a catalog, the search for a certain book requires looking through the shelves (linear search). In the worst case, the book in question happens to be on the last of them. A library catalog speeds up such searching because it replaces a linear search by specifying the exact address(es) of the physical location. Thus, a book may be found using the alphabetical order of the file cards, irrespective of how the actual locations of the books are arranged.

The electronic index of a textual database uses the letters of the alphabet like the keywords of a library catalog, specifying for each letter all its positions (addresses) in the storage medium. The occurrences of a certain word form, e.g., sale, is then computed from the intersection of the position sets of s, a, l, and e. The electronic index is built up automatically when the texts are read into the database, whereby the size of the index is roughly equal to that of the textual database itself.

The search for relevant texts or passages in the database is guided by the user on the basis of words he considers characteristic of the subject matter at hand. Consider for example a lawyer interested in legal decisions dealing with warranties in used car sales. After accessing an electronic database in which all federal court decisions since 1960 are stored, he specifies the words warranty, sale, and used car. After a few seconds the database returns a list of all the texts in which these words occur. When the user clicks on a title in the list the corresponding text appears on the screen.

The user might well find that not all texts in the query result are actually relevant for the purpose at hand. It is much easier, however, to look through the texts of the query result than to look through the entire database.

Also, the database might still contain texts which happen to be relevant to the subject matter, yet are not included in the query result. Such texts, however, would have to deal with the subject matter without mentioning the query words, which is unlikely.

The use of an electronic index has the following advantages over a card index:

2.1.2 Advantages of an Electronic Index

(a) Power of search
Because the electronic index of a textual database uses the letters of the alphabet as its keys, the database may be searched for any sequence of letters, whereas the keys of a conventional catalogue are limited to certain kinds of words, such as the name of the author.

(b) Flexibility
  (i) General specification of patterns
  An electronic index makes it possible to search for patterns. For example, the pattern[1] in.*i..tion matches all word forms of which the first two letters are in, the seventh letter from the end is i and the last four letters are tion, as in inhibition and inclination.
  (ii) Combination of patterns
  The electronic index allows to search for the combination of several word forms, whereby a maximal distance for their co-occurrence may be specified.

Though it is theoretically possible to create a conventional card index for the positions of each letter of the books in a library, this would not be practical. Therefore searching with patterns or the combination of keywords and/or patterns is not technically feasible with a conventional card index.

(c) Automatic creation of the index structure
The electronic index of a textual database is generated automatically during the reading-in of texts into the database. In a conventional card index, on the other hand, each new keyword requires making a new card by hand.

(d) Ease, speed, and reliability
While an electronic search is done automatically in milliseconds, is error free, and is complete, a conventional search using a card index requires human labor, is susceptible to errors, and may take anywhere from minutes to hours or days. The advantages of electronic search apply to both the *query* (input of the search words) and the *retrieval* (output of the corresponding texts or passages).
  (i) Query
  An electronic database is queried by typing the search patterns on the computer, while the use of a card index requires picking out the relevant cards by hand.
  (ii) Retrieval
  In an electronic database, the retrieved texts or passages are displayed on the screen automatically, while use of a conventional card index requires going to the library shelves to get the books.

The quality of a query result is measured in terms of recall and precision.[2]

2.1.3 Definition of Recall and Precision

Recall measures the percentage of relevant texts retrieved as compared to the total of relevant texts contained in the database.

[1] A widely used notation for specifying patterns of letter sequences are *regular expressions* (RegEx) as implemented in Unix.

[2] See Salton (1989), p. 248.

For example: a database of several million pieces of text happens to contain 100 texts which are relevant to a given question. If the query returns 75 texts, 50 of which are relevant to the user and 25 irrelevant, then the recall is $50 : 100 = 50\ \%$.

Precision measures the percentage of relevant texts contained in the result of a query.
For example: a query results in 75 texts of which 50 turn out to be relevant to the user. Then the precision is $50 : 75 = 66.6\ \%$.

Experience has shown that recall and precision are not independent of each other, but inversely proportional: a highly specific query will result in low recall with high precision, while a loosely formulated query will result in high recall with low precision.

High recall has the advantage of retrieving a large percentage of the relevant texts from the database. Because of the concomitant low precision, however, the user has to work through a huge amount of material most of which turns out to be irrelevant.

High precision, on the other hand, produces a return most of which is relevant for the user. Because of the concomitant low recall, however, the user has to accept the likelihood that a large percentage of relevant texts remain undiscovered.

Measuring recall is difficult in large databases. It presupposes exact knowledge of all the texts or passages which happen to be relevant for any given query. To measure the recall for a given query, one would have to search the entire database manually in order to objectively determine the complete set of documents relevant to the user's question and to compare it with the automatic query result.

Measuring precision, on the other hand, is easy, because the number of documents returned by the system in response to a query is small compared to the overall database. The user need only look through the documents in the query result in order to find out which of them are relevant.

In a famous and controversial study, Blair and Maron (1985) attempted to measure the average recall of a commercial database system called STAIRS.[3] For this purpose they cooperated with a large law firm whose electronic data comprised 40 000 documents, amounting to a total of 350 000 pages. Because of this substantial, but at the same time manageable, size of the data it was possible to roughly determine the real number of relevant texts for 51 queries with the assistance of the employees.

Prior to the study, the employees subjectively estimated an electronic recall of 75 %. The nonelectronic verification, however, determined an average recall of only 20 %, with a standard deviation of 15.9 %, and an average precision of 79.0 %, with a standard deviation of 22.2 %.

[3] STAIRS is an acronym for *Storage and Information Retrieval System*, a software product developed by IBM and distributed from 1973 to 1994.

2.2 Using Grammatical Knowledge

The reason for the surprisingly low recall of only 20 % on average is that STAIRS uses only technological, i.e., letter-based, methods. Using grammatical knowledge in addition, recall could be improved considerably. Textual phenomena which resist a technological treatment, but are suitable for a linguistic solution, are listed below under the heading of the associated grammatical component.

2.2.1 Phenomena Requiring Linguistic Solutions

(a) *Morphology*
A letter-based search does not recognize words. For example, the search for sell will overlook relevant forms like sold.
A possible remedy would be a program for word form recognition which automatically assigns to each word form the corresponding base form (lemmatization). By systematically associating each word form with its base form, all variants of a search word in the database can be found. A program of automatic word form recognition would be superior to the customary method of truncation – especially in languages with a morphology richer than that of English.

(b) *Lexicon*
A letter-based search does not take semantic relations between words into account. For example, the search for car would ignore relevant occurrences such as convertible, pickup truck, station wagon, etc.
A lexical structure which automatically specifies for each word the set of equivalent terms (synonyms), of the superclass (hypernyms), and of the instantiations (hyponyms) can help to overcome this weakness, especially when the domain is taken into account.[4]

(c) *Syntax*
A letter-based search does not take syntactic structures into account. Thus, the system does not distinguish between, for example, teenagers sold used cars and teenagers were sold used cars.
A possible remedy would be a syntactic parser which recognizes different grammatical relations between, for example, the subject and the object. Such a parser, which presupposes automatic word form recognition, would be superior to the currently used search for words within specified maximal distances.

[4] Some database systems already use *thesauri*, though with mixed results. Commercially available lexica are in fact likely to lower precision without improving recall. For example, in *Webster's New Collegiate Dictionary*, the word car is related to vehicle, carriage, cart, chariot, railroad car, streetcar, automobile, cage of an elevator, and part of an airship or balloon. With the exception of automobile and perhaps vehicle, all of these would only lower precision without improving recall.

(d) *Semantics*
A letter-based search does not recognize semantic relations such as negation. For example, the system would not be able to distinguish between selling cars and selling no cars. Also, equivalent descriptions of the same facts, such as A sold x to B and B bought x from A, could not be recognized.
Based on a syntactic parser and a suitable lexicon, the semantic interpretation of a textual database could analyze these distinctions and relations, helping to improve recall and precision.

(e) *Pragmatics*
According to Blair and Maron (1985), a major reason for poor recall was the frequent use of context-dependent formulations such as concerning our last letter, following our recent discussion, as well as nonspecific words such as problem, situation, or occurrence.
The treatment of these frequent phenomena requires a complete theoretical understanding of natural language pragmatics. For example, the system will have to be able to infer that, for example, seventeen-year-old bought battered convertible is relevant to the query used car sales to teenagers.

In order to improve recall and precision, linguistic knowledge may be applied in various different places in the database structure. The main alternatives are whether improvements in the search should be based on preprocessing the query, refining the index, and/or postprocessing the result. Further alternatives are an automatic or an interactive refinement of the query and/or the result, as described below.

2.2.2 Linguistic Methods of Optimization

A Preprocessing the query
1. Automatic query expansion
   (i) The search words in the query are automatically 'exploded' into their full inflectional paradigm and the inflectional forms are added to the query.
   (ii) Via a thesaurus the search words are related to all synonyms, hypernyms, and hyponyms. These are included in the query – possibly with all their inflectional variants.
   (iii) The syntactic structure of the query, e.g., A sold x to B, is transformed automatically into equivalent versions, e.g., B was sold x by A, x was sold to B by A, etc., to be used in the query.
2. Interactive query improvement
   The automatic expansion of the query may result in an uneconomic widening of the search and considerably lower precision. Therefore, prior to the search, the result of a query expansion is presented to the user to eliminate useless aspects of the automatic expansion and to improve the formulation of the query.

B Improving the indexing
   1. Letter-based indexing
      This is the basic technology of search, allowing one to retrieve the positions of each letter and each letter sequence in the database.
   2. Morphologically based indexing
      A morphological analyzer is applied during the reading in of texts, relating each word form to its base form. This information is coded into an index which for any given word (base form) allows one to find the associated word forms in the text.
   3. Syntactically based indexing
      A syntactic parser is applied during the reading in of texts, eliminating morphological ambiguities and categorizing phrases. This information is coded into an index on the basis of which all occurrences of a given syntactic construction may be found.
   4. Concept-based indexing
      The texts are analyzed semantically and pragmatically, whereby the software eliminates syntactic and semantic ambiguities and infers special uses characteristic of the domain. This information is coded into an index on the basis of which all occurrences of a given concept may be found.

C Postquery processing
   1. The low precision resulting from a nonspecific formulation of the query may be countered by an automatic processing of the data retrieved. Because there is only a small amount of raw data retrieved, as compared to the database as a whole, it may be parsed after the query and checked for their content. Then only those texts are displayed which are relevant according to this postquery analysis.

The ultimate goal of indexing textual databases is a concept-based retrieval founded on a complete morphological, syntactic, semantic, and pragmatic analysis of the texts.

2.3 Smart vs. Solid Solutions in Computational Linguistics

Which of the alternatives mentioned above is actually chosen in the design of a textual database depends on the amount of data to be handled, the available memory and speed of the hardware, the users' requirements regarding recall, precision, and speed of the search, and the designer's preferences and abilities. At the same time, the alternatives of 2.2.2 are not independent from each other.

For example, if an improvement of recall and precision is to be achieved via an automatic processing of the query, one can use a simple indexing. More specifically, if the processing of the query explodes the search words into their full inflectional paradigm for use in the search, a morphological index of the database would be su-

perfluous. Conversely, if there is a morphological index, there would be no need for exploding the search words.

Similarly, the automatic expansion of queries may be relatively carefree if it is to be scrutinized by the user prior to search. If no interactive fine-tuning of queries is provided, on the other hand, the automatic expansion should be handled restrictively in order to avoid a drastic lowering of precision.

Finally, the indexing of texts may be comparatively simple if the results of each query are automatically analyzed and reduced to the most relevant cases before being output to the user. Conversely, a very powerful index method, such as concept-based indexing, would produce results with such high precision that there would be no need for an automatic postprocessing of results.

The different degrees of using linguistic theory for handling the retrieval from textual databases illustrate the choice between smart versus solid solutions. Further examples are the following:

2.3.1 Smart Solutions in Computational Linguistics

Smart solutions avoid difficult, costly, or theoretically unsolved aspects of natural communication, as in

(a) Weizenbaum's Eliza program, which appears to understand natural language, but doesn't (cf. the Introduction in the Frontmatter)
(b) direct and transfer approaches in machine translation, which avoid understanding the source text (Sects. 2.4 and 2.5), and
(c) finite state technology (Sect. 13.5) and statistics (Sect. 15.5) for tagging and probabilistic parsing.[5]

Initially, smart solutions seem cheaper and quicker, but they are costly to maintain and their accuracy cannot be substantially improved. The alternative is solid solutions:

2.3.2 Solid Solutions in Computational Linguistics

Solid solutions aim at a complete theoretical and practical understanding of natural language communication. Applications are based on ready-made off-the-shelf components such as

(a) online lexica,
(b) rule-based grammars for the syntactic-semantic analysis of word forms and sentences,

[5] These methods may seem impressive because of the vast number of toys and tools assembled in the course of many decades (Jurafsky and Martin 2000). But they do not provide an answer to the question of how natural language communication works. Imagine that the Martians came to Earth and modeled cars statistically: they would never run. What is needed instead is a functional reconstruction of the engine, the transmission, the steering mechanism, etc., i.e., a solid solution.

(c) parsers and generators for running the grammars in the analysis and production of free text, and
(d) reference and monitor corpora for different domains, which provide a systematic, standardized account of the current state of the language.

Solid solution components are an application-independent long-term investment. Due to their systematic theoretical structure they are easy to maintain, can be improved continuously, and may be used again and again in different applications.

Whether a given task is suitable for a smart or a solid solution depends to a great extent on whether the application requires a perfect result or whether a partial answer is sufficient. For example, a user working with a giant textual database will be greatly helped by a recall of 70 %, while a machine translation system with 70 % accuracy will be of little practical use.

This is because a 70 % recall in a giant database is much more than a user could ever hope to achieve with human effort alone. Also, the user never knows which texts the system did not retrieve.

In translation, in contrast, the deficits of an automatic system with 70 % accuracy are painfully obvious to the user. Furthermore there is an alternative available, namely professional human translators. Because of the costly and time-consuming human correction required by today's machine translation, the user is faced daily with the question of whether or not the machine translation system should be thrown out altogether in order to rely on human work completely.

Another, more practical factor in the choice between a smart and a solid solution in computational linguistics is the off-the-shelf availability of grammatical components for the natural language in question. Such components of grammar, e.g., automatic word form recognition, syntactic parsing, etc., must be developed independently of any specific applications as part of basic research – solely in accordance with the general criteria of (i) their functional role as components in the mechanism of natural communication, (ii) completeness of data coverage, and (iii) efficiency.

Modular subsystems fulfilling these criteria can be used in practical applications without any need for modification, using their standard interfaces. The more such modules become available as ready-made, well-documented, portable, off-the-shelf products for different languages, the less costly will be the strategy of solid solutions in practical applications.

The main reason for the long-term superiority of solid solutions, however, is quality. This is because a 70 % smart solution is typically very difficult or even impossible to improve to 71 %.

2.4 Beginnings of Machine Translation

The choice between a smart and a solid solution is exemplified by machine translation. Translation in general requires understanding a text or utterance in a certain language (interpretation) and reconstructing it in another language (production).

On the one hand, translation goes beyond the general repertoire of natural communication. Superficially, it may seem related to bilingual communication. Bilingualism, however, is merely the ability to switch between languages, whereby only *one* language is used at any given time – in contradistinction to translation.

On the other hand, translation offers the facilitating circumstance that a coherent source text is given in advance – in contrast to automatic language production, which has to grapple with the problems of 'what to say' and 'how to say it'. A given source text can be utilized to avoid the really difficult problems of interpretation (e.g., a language-independent modeling of understanding) and production (e.g., the selection of content, the serialization, the lexical selection) in order to automatically translate large amounts of nonliterary text, usually into several different languages at once.

The administration of the European Union,[6] for example, must publish every report, protocol, decree, law, etc., in the 24 different languages of the member states (as of 2013). For example, a decree formulated in French under a French EU presidency would have to be translated into the following 22 languages.

French → Bulgarian	French → Irish
French → Czech	French → Latvian
French → Danish	French → Lithuanian
French → Dutch	French → Maltese
French → English	French → Polish
French → Estonian	French → Portuguese
French → Finnish	French → Slovene
French → German	French → Slovak
French → Greek	French → Spanish
French → Hungarian	French → Swedish

Under a Danish EU presidency, on the other hand, a document might first be formulated in Danish. Then it would have to be translated into the remaining EU languages, resulting in another set of 22 language pairs.

The total number of language pairs for a set of different languages is determined by the following formula:

2.4.1 Formula to Compute the Number of Language Pairs

$$n \cdot (n-1), \quad \textit{where } n = \textit{number of different languages}$$

For example, an EU with 23 different languages has to deal with a total of $23 \cdot 22 = 506$ language pairs.

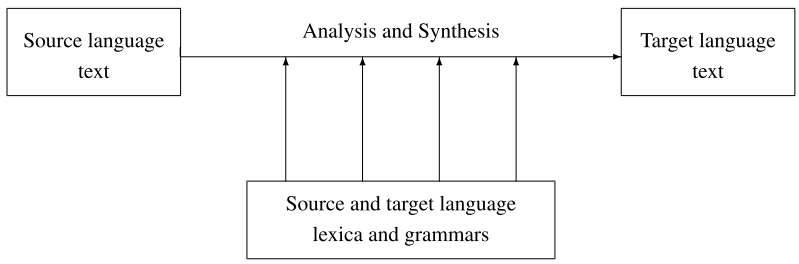
In a language pair, the source language (SL) and the target language (TL) are distinguished. For example, 'French→Danish' and 'Danish→French' are different language pairs. The source language poses the task of correctly *understanding* the meaning, taking into account the domain and the context of utterance, whereas the target language poses the task of *formulating* the meaning in a rhetorically correct way.

[6] Another example is the United Nations, which generates a volume of similar magnitude.

The first attempts at machine translation tried to get as far as possible with the new computer technology, avoiding linguistic theory as much as possible. This resulted in the smart solution of 'direct translation', which was dominant in the 1950s and 1960s.

Direct translation systems assign to each word form in the source language a corresponding form of the target language. In this way the designers of these systems hoped to avoid a meaning analysis of the source text, while arriving at translations which are syntactically acceptable and express the meaning correctly.

2.4.2 Schema of Direct Translation



Each language pair requires the programming of its own direct translation system.

Direct translation is based mainly on a differentiated dictionary, distinguishing many special cases for a correct assignment of word forms in the target language. In the source language, grammatical analysis is limited to resolving ambiguities as much as possible; in the target language, it is limited to adjusting the word order and handling agreement.

The methodological weakness of direct translation systems is that they do not systematically separate source language analysis and target language synthesis. Consequently one is forced with each new text to add new special cases and exceptions. In this way the little systematic structure which was present initially is quickly swept away by a tidal wave of exceptions and special cases.

Even though representatives of the direct approach asserted repeatedly in the 1950s that the goal of machine translation, namely

FULLY AUTOMATIC HIGH QUALITY TRANSLATION (FAHQT)

was just around the corner, their hopes were not fulfilled. Hutchins (1986) provides the following examples to illustrate the striking shortcomings of early translation systems:

2.4.3 Examples of Automatic Mis-Translations

Out of sight, out of mind. ⇒ *Invisible idiot.*

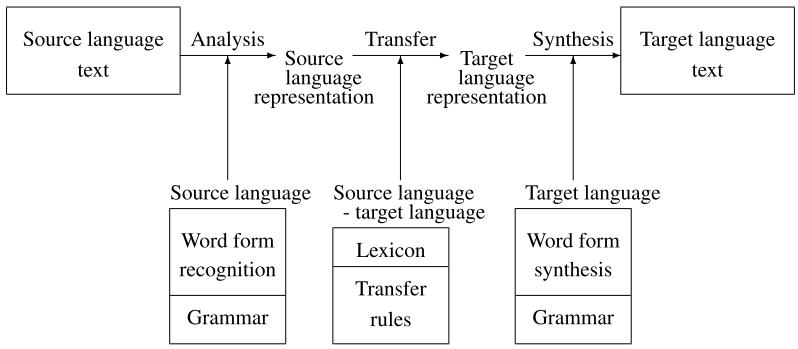
The spirit is willing, but the flesh is weak. ⇒ *The whiskey is all right, but the meat is rotten.*

La Cour de Justice considère la création d'un sixième poste d'avocat général. ⇒ *The Court of Justice is considering the creation of a sixth avocado station.*

The first two examples are apocryphal, described as the result of an automatic translation from English into Russian and back into English. The third example is documented in Lawson (1983) as output of the SYSTRAN system.

An attempt to avoid the weaknesses of direct translation is the transfer approach:

2.4.4 Schema of the Transfer Approach



The transfer approach is characterized by a modular separation of

1. source language analysis and target language synthesis, of
2. linguistic data and processing procedures, and of the
3. lexica for source language analysis, target language transfer and target language synthesis.

The result is a clearer structure as compared to the direct approach, facilitating debugging and upscaling. Implementing the different modules independently of each other and separating the computational algorithm from the language-specific data also makes it possible to re-use parts of the software when adding another language pair.

For example, given a transfer system for the language pair A-B, adding the new language pair A-C will require writing new transfer and synthesis modules for language C, but the analysis module of the source language A may be re-used. Furthermore, if the language-specific aspects of the new transfer and synthesis modules are written within a prespecified software framework suitable for different languages, the new language pair A-C should be operational from the beginning.

The three phases of the transfer approach are illustrated below with a word form.

2.4.5 Three Phrases of a Word Form Transfer *English-German*

1. Source language analysis:

Unanalyzed surface:	`knew`
	⏐
Morphological and lexical analysis:	`(knew (N A V) know)`

The source language analysis produces the syntactic category (N A V) of the inflectional form (categorization) and the base form know (lemmatization).

2. Source-target language transfer:
   Using the base form resulting from the source language analysis, a source-target language dictionary provides the corresponding base forms in the target language.

   ```
   know  ⟹  wissen
             kennen
   ```

3. Target language synthesis
   Using the source language category (resulting from analysis) and the target language base forms (resulting from transfer), the desired target language word forms are generated based on target language morphology.

   ```
   wußte        kannte
   wußtest      kanntest
   wußten       kannten
   wußtet       kanntet
   ```

The transfer of a syntactic structure functions similarly to the transfer of word forms. First, the syntactic structure of the source language sentence is analyzed. Second, a corresponding syntactic structure of the target language is determined (transfer). Third, the target language structure is supplied with the target language word forms (synthesis), whereby correct handling of agreement, domain-specific lexical selection, correct positioning of pronouns, rhetorically suitable word order, and other issues of this kind must be resolved.

Due to similarities between the direct and the transfer methods, they have the following shortcomings in common:

2.4.6 Shortcomings of the Direct and the Transfer Approach

1. Each language pair requires a special source-target component.
2. Analysis and synthesis are limited to single sentences.
3. Semantic and pragmatic analyses are avoided, by attempting automatic translation without understanding the source language.

Thus, the advantage of the transfer approach over the direct approach is limited to the re-usability of certain components, specifically the source language analysis and the target language synthesis for additional language pairs.

2.5 Machine Translation Today

The importance of language *understanding* for adequate translation is illustrated by the following examples:

2.5.1 Syntactic Ambiguity in the Source Language

1. Julia flew and crashed the airplane.

   Julia (flew and crashed the airplane)
   (Julia flew) and (crashed the airplane)

2. Susanne observed the yacht with a telescope.
   Susanne observed the man with a beard.
3. The mixture gives off dangerous cyanide and chlorine fumes.

   (dangerous cyanide) and (chlorine fumes)
   dangerous (cyanide and chlorine) fumes

The first example is ambiguous between using the verb fly transitively (someone flies an airplane) or intransitively (someone/-thing flies). The second example provides a choice between an adnominal and an adverbial interpretation of the prepositional phrase (Sect. 12.5). The third example exhibits a scope ambiguity regarding dangerous. A human translator recognizes these structural ambiguities, determines the intended reading, and recreates the proper meaning in the target language.

A second type of problem for translation without understanding the source language arises from lexical differences between source and target language:

2.5.2 Lexical Differences Between Source and Target

1. The men killed the women. Three days later they were caught.
   The men killed the women. Three days later they were buried.
2. know: wissen savoir
   kennen connaître
3. The watch included two new recruits that night.

When translating example 1 into French, it must be decided whether they should be mapped into ils or elles – an easy task for someone understanding the source language. Example 2 illustrates the phenomenon of a *lexical gap*: whereas French and German distinguish between savoir–wissen and connaître–kennen, English provides only one word, know. Therefore a translation from English into French or German makes it necessary to choose the correct variant in the target language. Example 3 shows a language-specific lexical homonymy. For translation, it must decided whether watch should be treated as a variant of clock or of guard in the target language.

A third type of problem arises from syntactic differences between the source and the target language:

2.5.3 Syntactic Differences Between Source and Target

1. German:
   Auf dem Hof sahen wir einen kleinen Jungen, der einem Ferkel nachlief.
   Dem Jungen folgte ein großer Hund.

2. English:
   In the courtyard we saw a small boy running after a piglet.
   (a) A large dog followed the boy.
   (b) The boy was followed by a large dog.

German with its free word order can front the dative dem Jungen in the second sentence, providing textual cohesion by continuing with the topic. This cannot be precisely mirrored by the English translation because of its fixed word order. Instead, one can either keep the active verb construction of the source language in the translation (a), losing the textual cohesion, or one can take the liberty of changing the construction into passive (b). Rhetorically the second choice would be preferable in this case.

A fourth type of problem is caused by the fact that sequences of words may become more or less stable in a language, depending on the context of use. These fixed sequences range from frequently used 'proverbial' phrases to collocations and idioms.

2.5.4 Collocation and Idiom

strong current | high voltage (but: *high current | *strong voltage)
bite the dust | ins Gras beißen (but: *bite the grass | *in den Staub beißen)

For adequate translation, colloquial and idiomatic relations such as those above must be taken into account.

The problems illustrated in 2.5.1–2.5.4 cannot be treated within morphology and syntax alone. Thus, any attempt to avoid a semantic and pragmatic interpretation in machine translation leads quickly to a huge number of special cases. As a consequence, such systems cannot be effectively maintained.

In light of these difficulties, many practically oriented researchers have turned away from the goal of fully automatic high quality translation (FAHQT) to work instead on partial solutions which promise quick help in high volume translation.

2.5.5 Partial Solutions for Practical Machine Translation

1. *Machine-aided translation* (MAT) supports human translators with comfortable tools such as online dictionaries, text processing, morphological analysis, etc.
2. *Rough translation* – as provided by an automatic transfer system – arguably reduces the translators' work to correcting the automatic output.
3. *Restricted language* provides a fully automatic translation, but only for texts which fulfill canonical restrictions on lexical items and syntactic structures.

Systems of restricted language constitute a positive example of a smart solution. They utilize the fact that the texts to be translated fast and routinely into numerous different languages, such as maintenance manuals, are typically of a highly schematic nature. By combining aspects of automatic text generation and machine translation, the structural restrictions of the translation texts can be exploited in a twofold manner.

First, an online text processing system helps the authors of the original text with highly structured schemata which only need to be filled in (text production). Second, the online text system accepts only words and syntactic constructions for which correct translations into the various target languages have been carefully prepared and implemented (machine translation).

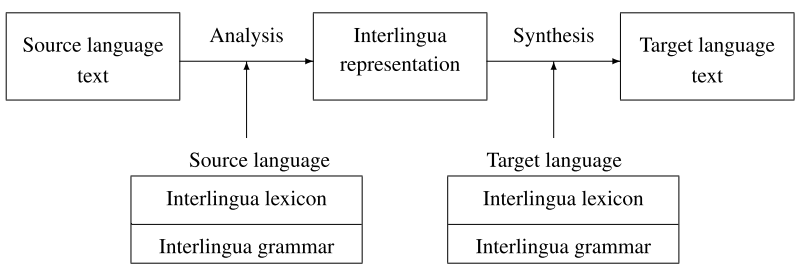
The use of restricted language may be compared to the use of a car. To take advantage of motorized transportation, one has to stay on the road. In this way one may travel much longer distances than one could on foot. However, there are always places a car cannot go. There one can leave the car and continue by walking.

Similarly, due to their automatic input restrictions, systems of restricted language provide reliable machine translation which is sufficiently correct in terms of form and content. If the text to be translated does not conform to the restricted language, however, one may switch off the automatic translation system and look for a human translator.

Besides these smart partial solutions, the solid goal of fully automatic high quality translation (FAHQT) for nonrestricted language has not been abandoned. Today's theoretical research concentrates especially on the interlingua approach, including knowledge-based systems of artificial intelligence. In contrast to the direct and the transfer approach, the interlingua approach does not attempt to avoid semantic and pragmatic interpretation from the outset.

The interlingua approach is based on a general, language-independent level called the interlingua. It is designed to represent contents derived from different source languages in a uniform format. From this representation, the surfaces of different target languages are generated.

2.5.6 Schema of the Interlingua Approach



An interlingua system handles translation in two independent steps. The first step translates the source language text into the interlingua representation (analysis). The second step maps the interlingua representation into the target language (synthesis).

It follows from the basic structure of the interlingua approach that for $n(n-1)$ language pairs only $2n$ interlingual components are needed (namely n analysis and n synthesis modules), in contrast to the direct and the transfer approach which re-

quire $n(n-1)$ components. Thus, as soon as more than three languages ($n > 3$) are involved, the interlingua approach has a substantial advantage over the other two.

The crucial question, however, is the exact nature of the interlingua. The following interlinguas have been proposed:

1. an artificial logical language,
2. a semi-natural language like Esperanto which is man-made, but functions like a natural language,
3. a set of semantic primitives common to both the source and the target language, serving as a kind of universal vocabulary.

Closer inspection shows, however, that these proposals have not yet resulted in theoretically and practically acceptable results. Existing interlingua systems are highly experimental, usually illustrating theoretical principles by translating tiny amounts of data by means of huge systems.

The experimental character of these attempts is not surprising because a general solution to interlingua translation may almost be equated with modeling the mechanism of natural language communication. After all, interlingua translation requires (i) a language-independent representation of cognitive content in the interlingua, (ii) the automatic translation of the natural source language into the language-independent content representation, and (iii) the automatic generation of the natural target language from the language-independent content representation.

Conversely, as soon as natural communication has been modeled on the computer in a general way, fully automatic high quality translation (FAHQT) is within reach. At the same time, all the other applications of computational linguistics mentioned in 1.1.2, such as human-computer communication in natural language, a concept-based indexing of textual databases with maximal recall and precision, etc., can be provided with solid solutions as well, using available off-the-shelf modules.

These applications are one reason why the SLIM theory of language aims from the outset at modeling the mechanism of natural language communication in general. Thereby verbal and nonverbal contents are represented alike as concatenated propositions, defined as sets of bidirectionally connected *proplets* in a content-addressable database. Though partially language-independent, these contents may differ from language to language in terms lexicalization and agreement information.

Exercises

Section 2.1

1. Explain the notions recall and precision using the example of a database containing 300 texts relevant for a given query, whereby 1000 texts are retrieved, of which 50 turn out to be relevant.
2. What are the weaknesses of purely technology-based indexing and retrieval in textual databases?
3. Give examples in which truncation retrieves irrelevant and misses relevant word forms.
4. Read Chap. 11, *Language Analysis and Understanding*, in Salton (1989) (pp. 377–424). Give a written summary of three to five pages of the linguistic methods for improving information retrieval described there.

Section 2.2

1. Which components of a textual database system are susceptible to linguistically based optimization?
2. What is the difference between on-the-fly processing and batch mode processing? Illustrate the difference using the examples of query expansion, indexing, and processing of a query result.
3. Why is high quality indexing better suited for fast search than preprocessing the query or postprocessing the retrieved data?
4. What is the cost of high quality indexing as compared to preprocessing the query or postprocessing the retrieved data?
5. Investigate whether the thesaurus of *WordNet* raises or lowers precision.

Section 2.3

1. What are the different ways of improving retrieval from a textual database, and how are they connected with each other?
2. Describe the different advantages and disadvantages of smart versus solid solutions in applications of computational linguistics.
3. What kinds of applications are not suitable for smart solutions?
4. Call up the Eliza program in the Emacs editor with ‘meta-x doctor’ and check it out. Explain why the Eliza program is a smart solution. What is the function of grammatical components in Eliza?

Section 2.4

1. Describe the differences between the direct and the transfer approach to machine translation.
2. What is the ultimate goal of machine translation?
3. In 1995, the EU was expanded from 12 to 15 member states, increasing the number of different languages from nine to 11. How many additional language pairs resulted from this expansion?

4. Which components of a transfer system can be re-used, and which cannot? Explain your answer with the example of six language pairs for the languages English, French, and German. Enumerate the components necessary in such a system.

Section 2.5

1. Which phenomena of language use make an understanding of the source language necessary for adequate translation?
2. It is sometimes pointed out that English has no word corresponding to the German Schadenfreude. Does this mean in your opinion that the corresponding concept is alien to speakers of English and cannot be expressed? Provide two further examples of *lexical gaps* relative to language pairs of your choice.
3. Provide two examples of collocations.
4. Where in MT could one use off-the-shelf components of grammar?
5. What linguistic insights could be gained from building a system of controlled language?
6. Why is machine translation with controlled language an example of a smart solution?
7. What is an interlingua?
8. How many additional components are needed when adding three new languages to an interlingua system?

http://www.springer.com/978-3-642-41430-5

Foundations of Computational Linguistics
Human-Computer Communication in Natural Language
Hausser, R.
2014, XXVIII, 518 p. 233 illus. in color., Hardcover
ISBN: 978-3-642-41430-5