A Constructivist Approach to Machine Translation

Michael Carl
Institut für Angewandte Informationsforschung,
Martin-Luther-Straße 14,
66111 Saarbrücken, Germany,
carl@iai.uni-sb.de

Introduction

Constructivist cognitive theories conceptualize memory as a dynamic process which is directly linked to perception, memories and conclusion/induction (cf. [Sch91]). From this point of view, memory serves to establish structures that are relevant to the cognitive system in the present context of action. The function of memory is thus to participate in coherent behavior which makes survival of the acting cognitive system easier (or possible). Memories are similar to perceptions: they are perceptions without an object. Perception on the other hand is an activity (and not a passive process) that is driven by the memory.

In some situations, for example we perceive solid bodies where — in terms of modern physics — there are no bodies at all, but processes of energy exchange; from psychoacoustics it is known that we can hear sounds that physically do not exist. Conversely, psychological experiments confirm that perception only takes place if an interpretation can be assigned to the perceived phenomenon [Foe86], [Foe73]. In successive steps of abstractions perception destroys parts of the information which cannot be expressed (or are of no importance) in the agent's model of the situation. This destruction will increase in proportion to the extent new situations cannot be handled. Confirmation of old solutions, then, will take place at the expense of new experiences.

In this paper, I present a new approach to Machine Translation (MT) that — similar to memory — does not simply 'remember' known solutions of former problems but creates new solutions for new problems. Like perception it 'perceives' a problem (i.e. a text or a sentence) to the degree it can handle it. Both, memory and MT are successful because they generate useful (e.g. consistent) behavior for an agent in a changing environment.

MT has existed since the very beginning of computer science and there are a number of different systems in research and on the market. However, with respect to many other domains, the problems of MT are due to two peculiarities of natural language: compositionality and hierarchical structure. Hierarchical structuring is well known in natural language processing (and MT) and accounts for the fact that words can be recursively grouped into constituents.

Compositionality in MT is a more basic phenomenon and refers to the observation that words or groups of words are translated identically in a different context. A source text and its translation can thus be considered to consist of composed units that are independently translated. The choice of these units, however, is much discussed in MT literature. If these units are too small, the system may become unreliable because it may be impossible to produce a correct target language text. If these units are too large the system may become too inflexible.

In this paper I propose a new MT system architecture which can accommodate a number of different translation units and is thus appropriate for a number of different user needs. In a constant interaction with the MT system, a user can tune 'her' system in order to obtain the results she would have expected. The architecture is similar to Example Based Machine Translation[1] (EBMT) and owes a lot to Case Based Reasoning (CBR). In the next section I will give a brief introduction to CBR. In order to apply the CBR paradigm to MT, I shall outline the need for a decomposition component and the way decomposition is usually treated in MT. Next, I will give a definition of compositionality. I will then show that the similarity metric — as it is used in many CBR systems — is not an appropriate means in MT either for the decomposition of the source text or for retrieval. Similarity, instead, can be based on abstraction: the more abstraction is performed, the more dissimilar two items are. In order for a MT system to react creatively on unknown input a certain degree of abstraction is required.

These considerations imply a number of constraints on the design of MT systems and offer two degrees of freedom that cannot analytically be determined. The choice of the translation units, their language correspondencies and the degree of creativity (abstraction) to which a MT system may recombine these units is a matter of user needs and depends on the goal of the application. Thus, in a very limited domain a MT system is likely to be different from an all-purpose MT system. The implications of these insights shall be discussed and related to the parameters in a MT system design.

The remainder of the paper is dedicated to an implementation of the outlined architecture which has the capacity to accommodate to a number of different user requirements. I shall call this approach Constructivist Machine

[1] The paradigm of Example Based Machine Translation has been started only recently, by a number of different authors (e.g. [SN90], [Bro96], [CC96]).

Michael Carl (1998) **A Constructivist Approach to Machine Translation.** In D.M.W. Powers (ed.) *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning,* ACL, pp 247-256.

Figure 1

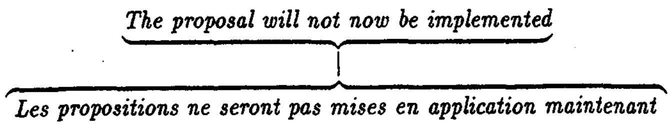
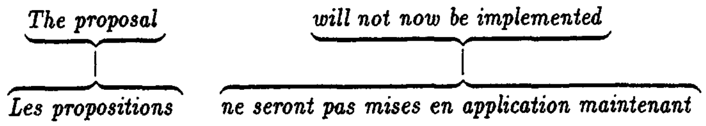

The proposal will not now be implemented
Les propositions ne seront pas mises en application maintenant

Figure 2


The proposal
will not now be implemented
Les propositions
ne seront pas mises en application maintenant

Translation because similar to memory, the success of a MT outcome — in this scenario — is solely validated by and dynamically tuned to the needs of the acting agent for whom the translation is being computed.

CBR-Systems

CBR systems are problem-solving systems that heavily rely on old experiences. For this reason they can be seen as a generalization of MT systems, too. CBR systems solve new problems by adapting (modifying) solutions to old problems that are stored in a case base.

According to Richter [Ric95] four knowledge containers are available in CBR systems:

- In the case base are stored former problems with their solutions (cases).
- A similarity metric makes possible retrieval of cases from the case base that are similar to new problems.
- An adaptation mechanism modifies the solutions of the retrieved cases according to the requirements of the problem.
- The case vocabulary contains the language in which cases are written.

Retrieval of the appropriate case(s) is based on a similarity metric which is designed such that the retrieved cases represent an instantiation of the problem concept. The problem and the retrieved case(s) are similar because they are different instantiations of the same concept.

In the adaptation step, the solution that is part of the retrieved case is modified where it differs from the problem solution. Adaptation is one of the most difficult parts in CBR systems because it depends on domain specific knowledge and on the outcome of the other CBR components.

In both, retrieval and adaptation, the original problem is thus destroyed in order to compute a solution: while classifying a problem according to the available conceptual classes in the case base, some parts of the problem are expected to be more important than other parts. In order to classify the problem appropriately, retrieval in CBR — like perception — thus evaluates parts of the information differently. The retrieved (set of) case(s) deviates in a number of properties from the original problem that seem minor to the goal of the agent. In adaptation, the retrieved solutions are further modified in order to generate a coherent outcome of the system.

However, in order to apply the CBR paradigm successfully to MT, a decomposition component is required that divides a sentence into a sequence of chunks. In the next section I shall explain the need for such a component and outline its importance in MT.

Decomposition in MT

In Machine Translation (MT) compositionality is crucial to attain a reasonable coverage. One of the main problems in MT is transfer strategy: how and when to translate a unit of the source language into a unit of the target language.

The underlying hypothesis of all MT systems is that units derived from the source language string can be mapped onto units from which the target language string can be computed. I use the following English-French translation example[2]:

1 *The proposal will not now be implemented* ⟷
Les propositions ne seront pas mises en application maintenant

Several possibilities of decomposition shall be considered in this section. The corresponding adaptation knowledge will be discussed.

If this sentence (1) consist of only one chunk i.e. decomposition takes place at a sentence level only as in figure 1, no adaptation is required. This is typical for translation memories (TM) (e.g. TRADOS [Hey96], TRANSIT) which have only a case base and a retrieval component[3]. TMs are likely to have quite long cases stored in the case base, but have a well informed distance metric to return similar cases from the case base. However, due to lack of adaptation, the less the retrieved cases match the sentence, the more incomplete and incorrect the translation is. A major shortcoming is that the increase of coverage does not follow the growth of the case base to the same extent.

[2] The example is taken from the Hansards-corpus and is discussed in [BCDP$^+$90]. For purpose of illustration I will consider the translation correct and desirable.

[3] Some TM, however, propose as an extra an interactive MT system or a batch MT system that is based on a different (e.g. 'traditional') paradigm.

Figure 3

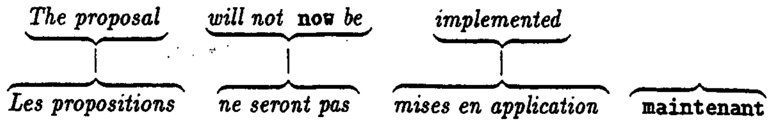
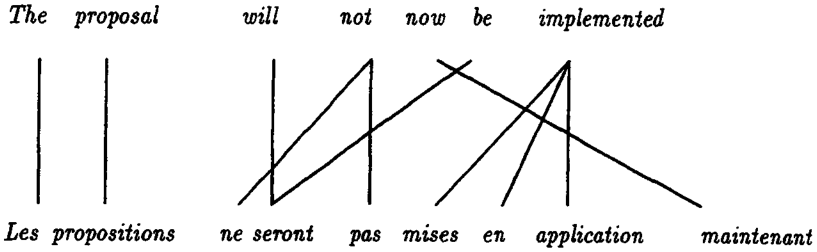


Figure 4



In figure 2, the sentence is decomposed into two chunks. Notice, that the English subject (the first chunk) is in singular, while its French translation is in plural. Because subject and predicate agree in number and person, the (auxiliary) verbs in the second chunk has to take the same features as in the first chunk. The adaptation mechanism has therefore to be able to reconstruct these agreement requirements.

If the sentence is divided into the three chunks */The proposal/*, */will not now be/* and */implemented/* as in figure 3 adaptation turns out to be much more complicated. It has to reconstruct agreement between the first and the second chunk (in the French translation for the third chunk too). Further, adaptation must take into account the discontinuity of the second chunk. Thus, although *now* and *maintenant* are translations of each other, they are separated by the interposed chunk 3 on the French side. The adaptation mechanism has to be able to integrate a continuous chunk into a discontinuous one when translating from English to French and the inverse when translating from French to English.

Example based Machine Translation (EBMT) corresponds to a decomposition granularity as shown in figures 2 and figure 3. Some systems (cf. [SN90], [CC96]) localize major constituents in the problem sentence by means of linguistic analysis. Adaptation in these systems is essentially a matter of replacing items (words, sequences of words or constituents) in the target language structure. Linguistic analysis serves to better determine the location and the appropriateness of potential items to be replaced in the target language.

In the Pangloss EBMT ([Bro96], [NBD94]) cases are selected from the case base that contain the problem case (or parts of it) as a substring. By means of a thesaurus and a bi-lingual lexicon the translation of the problem case (or its respective part) is extracted from the retrieved cases. Adaptation of the target language chunks is left to a statistical language model outside the Pangloss EBMT system.

Many traditional MT systems have an atomic case base and have no information about the similarity between a sentence and some cases i.e. only exact matching cases are retrieved from the case base. According to figure 4, sentences are very fine-grained and the main translation is carried out by the 'adaptation' mechanism. All three generations of traditional MT systems (cf. [WK95]) can be described by this schema. The direct approach seeks to map lexical items of the source language onto lexical items of the target language and then tries to rearrange the target test. The interlingual approach (cf. [Dor93]) tries to calculate a language independent meaning representation from which the target text is generated. The transfer approach (cf. [Str96]) is situated in between the two: abstractions of the source language string are computed and then transferred (mapped) into target units from which the target language string is computed. Traditional MT systems do not systematically make use of large chunks that could facilitate the adaptation mechanism. They thus fail to account for what computers can most easily do: memorization and retrieval.

The main difference, however, between traditional and more recent approaches to MT lies in the fact that the latter systems perform monolingual analysis and generation while in former systems the analysis (decomposition) of the source language is not independent from the regeneration possibilities in the target language.

Compositionality and MT

When decomposing a sentence in a particular way the sentence is classified with respect to the context[4]. In the French sentence *Le boucher sale la tranche* the word *sale* can be classified as a verb (English: *to salt*) or as an adjective (English: *dirty*), *la* can be an article (English: *the*) or a pronoun (English: *she/her*) and *tranche* can be a verb (English: *to chop*) or a noun (English: *slice*).

[4] This view is of course ot new. E.g. in [BDW96] morphological analysis (i.e. morphological classification) is defined as a decomposition task.

2 French: (*Le boucher*) *sale* (*la tranche*)
English: The butcher salts the slice

3 French: (*Le boucher sale*) (*la*) *tranche*
English: The dirty butcher chops her

If the sentence is decomposed according to (2) the phrase *Le boucher* is classified as the subject of the sentence and *la tranche* as the object. In (3) *Le boucher sale* is the subject while *la* is the object. Parsing, for instance, classifies the components of a sentence and their relationship by giving it a structure according to a grammar.

However, I will not use a grammar as a basis for decomposition but base the decomposition on examples. This, I shall show, leads to greater flexibility and better maintainability of the system.

In MT different decompositions become particularly crucial if they lead to different translations i.e if the source language and the target language express ambiguities in a different way, which is quite often the case. A source sentence S is compositionally translatable into a target sentence T, iff

- it is decomposable into a set of chunks.
  A sentence S is **decomposable**, iff it can be divided into a set of chunks $c_1 \ldots c_n$ where the intersection of the chunks' concepts equals the concept of the case: $\mathcal{C}(S) \equiv \bigcap_n \mathcal{C}(c_n)$
- the case base covers the chunks.
  A case base CB **covers** a set of chunks $c \in S$ iff there exists for each c at least one solution case $s \in CB$ where both c and s are instantiations of the same concept : $\forall c \in S \;\; \exists s \in CB : \mathcal{C}(c) \equiv \mathcal{C}(s)$
- the retrieved solutions are adaptable.
  A set of solutions $s_1 \ldots s_n$ is **adaptable**, iff it can be composed into one target sentence T and the intersection of the solutions' concepts equals the concept of the result: $\bigcap_n \mathcal{C}(s_n) \equiv \mathcal{C}(T)$

We thus obtain a chain of conceptual equivalences during all processing steps as shown in equation 1.

$$\mathcal{C}(S) \equiv \bigcap_n \mathcal{C}(c_n) \equiv \bigcap_n \mathcal{C}(s_n) \equiv \mathcal{C}(T) \qquad (1)$$

A source sentence S is decomposed into a set of chunks c for each of which a solution s is retrieved from the case base. The set of solutions is then composed into a target sentence T.

Similarity and MT

In CBR systems, the similarity metric is a means for classifying the problem according to the empirical data in the case base. In this section I shall investigate whether a similarity metric is appropriate for decomposition and classification in MT.

Similarity metrics in CBR are often based on nearest neighbor (NN) algorithms (e.g. [WD95]). NN algorithms make use of a continuous variable ω which associates a real value to the attributes a_i of a problem a. The similarity between a problem a and a case b is inversely proportional to their distance D[5]:

$$D(a,b) = \sum_i \omega_i \, * \, d(a_i, b_i)$$

The nearest neighbor in the hyperspace is then returned as the most similar known instantiation to the problem. Other approaches use symbolic distance metrics. One such approach ([Pla95]) uses anti-unification[6] that yields for each case in the case base the intersection (i.e. what is common) with the problem. By means of a predefined subsumption ordering these intersections are ordered. The cases that are most similar to a problem are those that yield most specific anti-unification results.

Other approaches (e.g. [Hut97]) calculate for each case the minimal number of changes that would have to be done to transform it into the problem. The weighted sum of changes then indicates the distance between the case and the problem.

While these metrics may be dynamically adapted in changing environments (the case base and/or the weight ω may be dynamically altered), there are at least two problems when applied to MT.

One problem occurs when decomposing a text of arbitrary length as was shown above. It is not at all evident how to decompose a text into units such that the adaptation capacity of the system is respected. Of course, the smaller the units are, the higher the probability of retrieval success will be. But we may not necessarily be sure whether the retrieved solutions lead to a compositionally correct translation because it may be impossible for the adaptation mechanism to appropriately recompose them.

Apart from the discontinuity of chunks as shown in figure 3, there are other phenomena of lexical cooccurrence that need to be treated by a MT system. For instance, if we know that German *stark* translates into English *strong* and German *Band* translates into English *volume* we are likely to translate German *starker Band* into English *strong volume* by simply concatenating the translated units. This might not always work well because here *thick volume* would be a better translation. The sequence *starker Band* should hence be seen as one undividable unit. Suitable decomposition of the problem is thus a prerequisite for valuable retrieval of cases.

A related problem is due to the way in which the solution of a case is related to the matching part. Partial similarity of a phrase and a case does not necessarily allow one to conclude that there are comparable similarities between their translations. For instance if we know that German *starke Erkältung* translates into English *bad cold* and German *Raucher* translates into English *smoker* we are likely to translate the unknown German phrase *starker Raucher* into English *bad smoker* because the first word of the unknown phrase is just another inflected form of the first word in the known phrase. We thus substitute the second word in the translation of

[5] In a symbolic task, the distance d usually is $d(a_i, b_i) = 0$ if $a_i = b_i$, else 1.

[6] While unification yields least upper bound (lub), anti-unification yields the greatest lower bound (glb) with respect to a subsumption ordering.

the known phrase (*cold*) by the translation of the second word of the unknown phrase (*smoker*) to obtain the result. However, this might not always be a good solution because *heavy smoker* (and not *bad smoker*) would be the appropriate translation. Of course, one can argue that *stark* has different readings according to the context in which it appears so that *starker Raucher* and *starke Erkältung* are not similar at all. However, the knowledge concerning the appropriateness of words which can be replaced needs to be coded in some way[7]. In the proposed architecture exceptions are stored in the case base and knowledge about replacable words is extracted (induced) from the case base. This makes possible a dynamic graduation between regularities subregularities an exceptions as it occurs in natural languages.

Merely similarity of an input text and some cases in the case base does not therefore lead to a satisfactory solution because it tells us neither how to decompose a text nor which parts in the retrieved cases are suitable for substitution. Further, from a logical point of view, similarity seems a useless notion because, as Goodman [Goo72] states, it cannot be measured in terms of, or equated with the possession of common characteristics:
Where the number of things in the universe is n, each two things have in common exactly 2^{n-2} properties out of the total of $2^n - 1$ properties; each thing has 2^{n-2} properties that the other does not, and there are $2^{n-2} - 1$ properties that neither has. [pp. 443-444]
The point here is that if two things have some properties in common this is saying nothing more than that they have these properties in common i.e. that they are equal with respect to the common properties. However, which of the shared properties are more salient is analytically untractable: it remains a matter of who makes the comparison and when. For instance in the examples above, one might find *weighty tome* to be a better translation for *starker Band* than *thick volume.* On the other hand German *starker Punkt* compositionally translates into English *strong point.* It is a decision of the present system architecture not to code these decisions into the program or into a grammar, but to leave it in the structure of the case base.

Abstraction as Similarity

Instead of having a similarity metric to classify a sentence (or parts of it) decomposition is used as a method for classification. In order to determine the similarity of a complex sentence and some cases in the case base abstraction by means of decomposition and reduction seems an appropriate means[8].

Abstractions are induced from the input sentence based on cases in the case base. The less an input sentence is known to the system, the more abstractions are needed for the sentence to be matched onto the case base. However, the more abstractions are performed, the greater will be the dissimilarity between the sentence and the matching case(es). Accordingly, it is stressed in [BW96] that the significance of similarity between a problem and a set of cases is more important the less abstract the cases are.

In the proposed system architecture a sentence is decomposed into a set of chunks according to the available cases in the case base. Chunks which share all their properties with a case are reduced and the sequence of reduced and unreduced chunks (i.e. the abstraction of the original sentence) is, again, decomposed and matched against the case base until no more decomposition and reduction is possible. In a number of steps, a sentence of length m is thus classified according to the available cases in the case base into maximal $2m$ chunks.
A sentence is regenerated from an abstraction by specifying the reduced chunks and their subsequent refinement. This is repeated until the produced sequence contains no more reduced chunks.
Abstraction (i.e. decomposition and reduction) and generation (i.e. specification and refinement) are possible if the following criteria hold for the matching chunks in the abstraction process and the reduced chunks in the generation process:

- Chunks are independent with respect to some 'fixed' features. The fixed values of one chunk does not affect the fixed values of another chunk.
- Chunks are adaptable with respect to some 'variable' features. The set of variable features for each chunk reflects its inter-chunk dependencies.

In order to translate the French sentence *Le boucher sale la tranche* into English *The butcher salts the slice* according to the classification 2 above we need the case base to contain the two concrete cases 4 and 5 and the abstract case 6:

4 *le boucher* ⟷ *the butcher*
5 *la tranche* ⟷ *the slice*
6 *X sale Y* ⟷ *X salt Y*

Based on the examples, the French sentence is decomposed into the three chunks c_1: *le boucher*, c_2: *sale* and c_3: *la tranche*. By reducing the (matching) chunks c_1 and c_3 the abstraction c_1 *salt* c_3 then matches case 6. The adaptation mechanism subsequently re-specifies and refines the solutions s_1 and s_3 of the reduced chunks c_1 and c_3. Specification consists in replacing s_1 in the abstraction by *the butcher* and s_3 by *the slice*. Refinement adapts the variable features such that the main verb *salt* agrees in number and person with the chunk inserted in position *X*. Note that the required adaptation complexity corresponds to figure 2 above.
In this translation the number of decompositions is 3. Another chunking is possible if the case base allows it. For the case base 7 below, the number of decompositions

[7] I agree with the comment of an anonymous reviewer that *a more thorough linguistic analysis may well yield a better performance for direct use of similarity metrics, subverting the needs of post hoc adaptations.*
The problem is how can you know when you have done sufficiently linguistic analyses without reference to the data?

[8] A similar idea can be found in [CC96]. However, in their approach sentences undergo a (rule-driven) syntactic analysis.

equals 1: the granularity of decomposition thus relies on the structure of the case base.

7 *le boucher sale la tranche* ⟷
the butcher salts the slice

Note that by means of case 4, 5 and 7, the abstract case 6 can be induced.

Undesirable abstraction is possible and is on the one hand an expression of the creativity of the system but on the other hand avoidable by adding further cases to the case base. Thus, abstraction 10 can be generated based on the cases 8 and 9. As outlined in the previous section, abstraction 10 has the potential to (wrongly) translate *starker Punkt* into *heavy point*. However, by adding case 11 to the case base this is no longer possible.

8 *Raucher* ⟷ *smoker*
9 *starker Raucher* ⟷ *heavy smoker*
10 *starker X* ⟷ *heavy X*
11 *starker Punkt* ⟷ *strong point*

Freedom in MT system design

In the equivalence (1) — here reproduced as (2) — the decomposition granularity, the structure of the case base and the adaptation mechanism depend on each other.

$$C(S) \equiv \bigcap_{n} C(c_n) \equiv \bigcap_{n} C(s_n) \equiv C(T) \qquad (2)$$

To preserve the conceptual equivalence between a source sentence S and its translation T, all three components need to be synchronized: a certain type of decomposition requires an adequate case base which covers the decomposed chunks and an appropriate adaptation mechanism which is able to re-combine the retrieved solutions into a target translation. However, the above equivalence contains two degrees of freedom: one degree is related to the number n of chunks the other is due to the definition of the conceptual equivalence C. Neither can analytically be determined because they are closely related to the requirements of a user and his expectations with regard to a MT outcome.

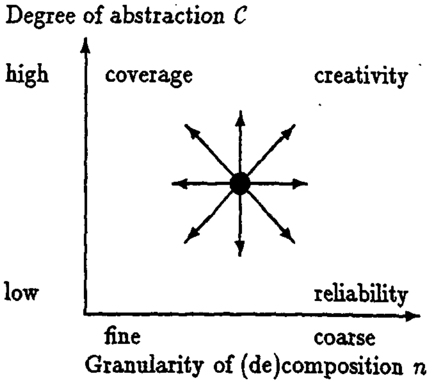
- The **coverage** of the case base increases while the length of the cases gets shorter. The coverage of the system depends on one hand on the coverage of the case base and on the other hand on the level of abstraction on which the chunks are matched. The coverage of the system thus increases with finer decomposition granularity and a high degree of abstraction.
- The **reliability** of the results is likely to increase while the length of the chunks gets longer and the system turns into a mere retrieval system of known solutions. Reliability thus increases with coarse decomposition granularity and low degree of abstraction.
- The **creativity** of the system combines both, coverage and reliability. A high degree of creativity can thus be reached with coarse decomposition granularity and high degree of abstraction.

Figure 5 shows possible realizations for MT systems. The horizontal axis represents the decomposition granularity; the vertical axis represents the degree of abstraction.

As the granularity of the decomposition becomes coarser, the system loses coverage but the translation result will become more reliable. The adaptation mechanism will be very simple. Conversely, finer granularity implies better coverage but requires a more complex adaptation mechanism. Orthogonal to decomposition granularity is the degree of abstraction that a system performs. The more abstractions are performed, the less reliable will be the outcome[9].

Creativity is necessary for a MT system unless the domain is restricted such that retrieval of already known translations is sufficient for the coverage. To attain a certain degree of creativity, the system needs to dispose of an appropriate degree of abstraction capacity joined with an appropriate decomposition granularity.

Figure 5



The more the system design moves into the upper left area (high degree of abstraction and fine decomposition granularity) in figure 5, the more the coverage of the system will increase. The more the design moves into the lower right area (low degree of abstraction and coarse decomposition granularity) the more the reliability of the system increases.

To reach reliability for unknown arbitrary texts, recent approaches to NLP prefer shallow analyses that generate flat representations (i.e. low degree of abstraction). Because the number of possible wrong assignments in an analysis tree grows exponentially with its depth, flat representations offer fewer possibilities to relate constituents and hence offer fewer possibilities to produce wrong analyses.

To attain a certain degree of creativity, an appropriate degree of abstraction and an appropriate degree of decomposition is required. What degree of creativity a user desires essentially depends on the variety of text types to be translated (i.e. the required coverage of the system)

[9] This is supported by the findings in [BDW96]: the more abstraction is performed by a system (***degree of eagerness***), the worse the generalization performance will be.

Figure 6

English Phrase Descriptor of the sentence: *The big man eats a green apple*

WD:	WD_{The}	WD_{big}	WD_{man}			WD_{eats}	WD_{a}	WD_{green}		WD_{apple}
LMA:	the	big	man			eats	a	green		apple
CAT:	art	adj	verb		noun	verb	art	adj	noun	noun
VTP:	—	—	fin	infin	—	fin	—	—	—	—
TNS:	—	—	pres	—	—	pres	—	—	—	—
NUM:	—	—	—	—	sing	—	—	—	—	sing
CAS:	—	—	—	—	n;a	—	—	—	n;a	n;a
DEG:	—	base	—	—	—	—	—	base	—	—
WNR:	1	2	3	3	3	4	5	6	6	7

German Phrase Descriptor of the sentence: *Der grosse Mann isst einen grünen Apfel*

WD:	WD_{Der}					WD_{grosse}	WD_{Mann}	WD_{isst}	WD_{einen}	$WD_{grünen}$		WD_{Apfel}
LMA:	d_art			d_rel		gross	mann	essen	ein	grün		apfel
CAT:	art			rel		adj	noun	verb	art	adj	verb	noun
VTP:	—	—	—	—	—	—	—	fin	—	—	fin	—
TNS:	—	—	—	—	—	—	—	pres	—	—	pres	—
NUM:	sg	sg	plu	sg	—	e*	sg	sg	sg	en*	plu	sg
GEN:	f	m	—	m	f	e*	m	—	m	en*	—	m
CAS:	d;g	n	g	n	g;d	e*	n;d;a	—	a	en*	—	n;d;a
DEG:	—	—	—	—	—	base	—	—	—	base	—	—
WNR:	1	1	1	1	1	2	3	4	5	6	6	7

e* and en* denote the endings (e and en) of a German adjective. They can be multiplied into a matrix of *AVM* containing information on GEN, CAS, NUM and the determination class.

LMA:	lemma (basic word form without inflectional information)	
CAT:	part-of-speech (syntactic category)	(adj; adv; art; noun;punct; rel; verb)
NUM:	number	(sing; plu)
VTP:	verb type	(fin; infin)
TNS:	tense	(pres;past)
CAS:	case	(n; g; d; a)
DEG:	degree of adjectives	(base; comp; sup)
WNR:	word number	

and the expected reliability of the results. In a constant feed-back process, a user thus has the possibility of designing the MT system according to his requirements.

In the remainder of this paper I will give a short overview of the CBAG[10] system. It can be used as a stand alone MT system or can be integrated with the Rule Based Machine Translation CAT2 [Str96]. Here, only the principal functioning shall be considered. CBAG consists of three modules: the Case Based Compilation module (CBC), the Case Base Analysis module (CBA) and the Case Base Generation module (CBG).

Case Structure in CBAG

Instead of simply storing surface strings in the case base, morphological analysis and lemmatization is carried out and added to the cases[11].

Lemmatization yields for a surface string a basic word form (lemma) that abstracts away from inflectional information which is contained in the surface form. Inflectional information such as *person*, *number* and *tense* for verbs or *case* and *number* for nouns is determined by the use of the word and is independent of the lemma. For instance, *man* and *men* are different instances of the same lemma (*man*) that only differ with respect to number (*singular* vs. *plural*).

The part of speech takes an intermediate position between lexical and grammatical information. On one hand, the part of speech is linked to a lemma, on the other hand each part of speech has typical inflectional patterns (e.g. the examples above).

The features of a word are stored in the form of sets of pairs of attribute/values *AVM*. We will refer to a set of *AVM* that belong to one word as a word descriptor *WD*. Note that the contents of each *AVM* is such that a single surface string can be regenerated from it. Morphological

[10] CBAG stands for Case Based Analysis and Generation

[11] We use MPRO (cf. [Maa96]) for morphological analysis and lemmatization. MPRO is a very powerful tool, which generates more than 95% correct analyses for arbitrary German and English texts.

analysis and lemmatization are thus reversible: a surface string can be transformed into a WD (a set of AVM) and a surface form can be generated from a AVM.

A phrase descriptor PD is a sequence of word descriptors $WD_1 \ldots WD_n$. A case $CASE$ is a pair of a source phrase descriptor PD_{source} and a target phrase descriptor PD_{target} that are considered to be translations of each other. A case base CB is a set of cases.

A PD can be represented as a $M \times N$ matrix where the columns describe the words of a phrase and the rows describe a sequence of attribute values. The figure 6[12] reproduces the $CASE$ of the following translation example:

12 *The big man eats a green apple* ⟷
Der grosse Mann isst einen grünen Apfel

Some words are ambiguous. For instance, the surface string *man* can be analyzed as a verb or as a noun as shown in the table in figure 6. The noun has an accusative or a nominative case, the verb can be the infinite form or the finite present form. The WD_{man} has thus four interpretations that are melted here together into three AVM.

Decomposition in CBAG

Decomposition is example driven and divides a PD into a set of chunks. I distinguish between two ways of decomposition:

Horizontal decomposition divides the PD matrix into a set of 'fixed' (lexical) features and into a set of 'variable' (grammatical) features. This division accounts for the distinction between lexical and grammatical information inherent in every sentence.

Agreement within a noun phrase is a grammatical phenomenon. The corresponding features are thus part of the set of variable features. In German, for instance, the determiner, the adjective and the noun in a noun phrase have to agree in number, gender and case, while the actual lexical fillers of this syntactic schema may vary.

The set of variable features comprises all features that can be altered by a different context (e.g. number and case for nouns, gender, tense, person for verbs, etc.). The set of fixed features comprises the lemma and the part of speech.

Vertical decomposition divides the PD matrix into a sequence of chunks. Vertical decomposition accounts for the compositionality of languages. In many contexts, for example, the English noun phrase *the man* would be translated into German *der Mann*. Most sentences can be considered as being composed of a sequence of chunks, that, to a certain extent, can be translated independently.

The reasoning behind the double decompositions is as follows:

- reduce the size of the case base: Each set of values needs to be stored only once. Thus *man* and *men* are instances of the same lemma, which only needs to be stored once[13]. Vertical decomposition reduces the size of the case base more dramatically: if sentences are considered to consist of words which are grouped into compositionally translatable chunks, only those groups of words must be stored as cases in the case base which do not allow for compositional translation.
- reduce retrieval time: With a smaller case base the retrieval time of cases should also decrease.
- increase coverage of the system: Cases that can be recomposed to complex solutions can occur in different contexts. These cases thus cover several problems.
- make possible case abstraction: While fixed features are specific for a chunk, variable features are typical for it. Case abstraction consists in abstracting away from the specificities of a chunk by keeping track of the variable (thus typical) features.

Abstraction in CBAG

Case abstraction is crucial in attaining a broader coverage of the system and to account for interdependencies of chunks. In the translation phase abstractions are computed from the input PD_{source} in order to match abstractions in the case base.

In the compilation phase of the case base, abstract cases are computed from the cases that are in the case base. Those chunks that match a case are reduced into a chunk descriptor CD which consists of a set of variable features and a chunk index. To reduce a chunk the head information is extracted from it, where head information is made upon those features that are necessary and sufficient to express inter-chunk dependencies such as agreement. The index of the chunk is stored with the head information in the CD. A sequence of reduced and unreduced CDs (an abstract case) may, again, be decomposed and matched against the case base.

In the following example, the case base contains two cases:

13 *the big man* ⟷ *der grosse Mann*

14 *a green apple* ⟷ *ein grüner Apfel*

As shown in the figure 7, the English sentence *The big man eats a green apple* is decomposed into the three chunks */The big man/ /eats/* and */a green apple/*. By abstraction, the sequence CD_1 CD_2 CD_3 is generated, where CD_1 and CD_3 represent reduced chunks that include information on the type of constituent (such as gender, number etc.) and the index of the matching case. If a WD cannot be integrated into a chunk — as is the case for *eat* in the example — it is integrated as an unreduced chunk into the abstract case.

Abstract cases are generated in a precompilation step as shown in figure 9. A new case base CB is incrementally created starting from an ordered set of examples. Based

[12] For reasons of space, not all features are given in the matrix.

[13] This is even more important for highly inflective languages and paradigms, as for example for Romance verbs which can have up to 40 different surface forms dependent on person, number, modality, aspect and tense.

Figure 7
Decomposition and reduction: The English *PD the big man eats a green apple* is decomposed and reduced into the abstraction CD_1 CD_2 CD_3.

$$\overset{CD_1}{\overset{\uparrow}{\overbrace{WD_{the}\ WD_{big}\ WD_{man}}}} \quad \overset{CD_2}{\overset{\uparrow}{WD_{eat}}} \quad \overset{CD_3}{\overset{\uparrow}{\overbrace{WD_{a}\ WD_{green}\ WD_{apple}}}}$$

Figure 8
Specification and refinement: The abstraction CD_1 CD_2 CD_3 is specified and refined into the German *PD Der grosse Mann isst einen grünen Apfel*.

$$\overset{CD_1}{\overset{\downarrow}{\overbrace{WD_{Der}\ WD_{gross}\ WD_{Mann}}}} \quad \overset{CD_2}{\overset{\downarrow}{WD_{essen}}} \quad \overset{CD_3}{\overset{\downarrow}{\overbrace{WD_{ein}\ WD_{grün}\ WD_{Apfel}}}}$$

Figure 9
Algorithm to induce abstract cases.

```
Sort examples E by number of words
CB <- empty
for all examples E /* shortest  first */
begin
    V <- TRUE
    while V = TRUE
    begin
        A <- reduce(decomp(E,CB))
        add E to CB
        if valid(A)
        then E <- A
        else V <- FALSE
    end
end
```

on partial matches of the example E and the case base CB, first an abstraction A is calculated. Than the original concrete example is added to the case base. This order is important because the abstractions should not rely on the same example.

Abstractions are calculated in two steps: decomposition decomp and reduction reduce. An abstraction is valid if:

1. it contains at least one unreduced chunk
2. it contains at least one reduced chunk for both, source and target language side
3. decomposition is based on the same cases on both language sides
4. if it is validated by some other cases

The requirements 1 and 2 express well-formedness conditions of abstractions. A case that only consists of unreduced *CD*s is not an abstract case because it contains no reductions. An abstraction that only consists of reduced chunks is not stored in the case base because it will never be matched by other cases. The requirement 3 accounts for the fact that abstractions are generalizations of regularities contained in the concrete case. A pair of a source and a target sentence that are equally decomposed are likely to be regular in that decomposition. Requirement 4 excludes exceptions to serve as a basis of abstractions as is the case in the *heavy smoker* example 9. However, finer grained criteria on this matter can be found in [CC96].

Generation

Generation performs the reverse task of Abstraction. To generate a target language *PD* from an abstraction, the abstract case is first specified and then refined into a sequence of lower level *CD*. This is repeated until the sequence contains no more reduced *CD* i.e. the concrete PD_{target}. Specification extends the chunk descriptors index by searching from the case base the solutions (i.e. translations) of the appropriate cases. Refinement substitutes the variable features in the specified *CD*s according to the head information.

In figure 8 the chunk descriptor CD_1 is specified into the sequence WD_{Der} WD_{gross} WD_{Mann}. The chunk descriptor CD_2 is specified into the sequence WD_{ein} $WD_{grün}$ WD_{Apfel} according to the cases in the case base given in the last section. While refining CD_1, the corresponding *WD*s are transformed into nominative singular because the first chunk is the subject of the phrase. CD_2 is refined into accusative singular because the second chunk is the (accusative) object of the phrase. The sequence of *WD*s can then be sent to a morphological generation module, which calculates the appropriate surface strings.

Implementation

The system as described is implemented in C and runs under gnu-C on sun machine. It consists of several programs, that are connected via (unix) pipes. The KURD (cf. [CSW98] in this volume) formalism is a constraint-based shallow parser that is used for chunk reduction and chunk refinement. Another program is used as a data base system for quick retrieval and decomposition of attribute value matrices as described above.

Conclusion

In this paper, a constructivist view on MT is outlined. I start by saying that MT systems work similarly to (human) memory because both create new solutions in a dynamic environment to ensure successful behavior for a cognitive agent. Applying this paradigm to MT, however, implies a number of restrictions. First, the problem (sentence, text) needs to be decomposed. A similarity metric that relies on a continuous variable is not an appropriate means for classification because very similar cases may have distinct solutions. Therefore, to compositionally classify a sentence, it is decomposed and abstracted based on the examples in the case base. An abstraction is regenerated in the target language by specification and refinement of the reductions.

According to the constructivist approach I shall argue that a text can be compositionally translated into a target language if and only if 1) it can be decomposed 2) the components can be translated into the target language and 3) these target language components can be adapted (recomposed) into a valid target language text. Whether or not the translation is valid, however, solely depends on the acting agent for whom the translation is to be computed.

Acknowledgement

I would like to thank Cath Pease, Antje Schmidt-Wigger, Oliver Streiter and the anonymous rewiever for the many valuable comments.

References

[BCDP+90] Peter F. Brown, J. Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, F. Jelinek, Robert L. Mercer, and P.S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85, 1990.

[BDW96] Antal van den Bosch, Walter Daelemans, and Ton Weijters. Morphological Analysis as Classification: an Inductive-Learning Approach. In *Proceedings of NeMLaP*, Ankara, 1996.

[Bro96] D. Ralf Brown. Example-Based Machine Translation in the Pangloss System. In *COLING-96*, 1996.

[BW96] R. Bergmann and W Wilke. On the Role of Abstractions in Case-Based Reasoning. In *EWCBR-96 European Conference on Case-Based Reasoning*. Springer, 1996.

[CC96] Bróna Collins and Pádraig Cunningham. Adaptation guided retrieval in EBMT: A case-based approach to machine translation. In *Advances in CBR*, LNAI, pages 91–104. Springer, 1996.

[CSW98] Michael Carl and Antje Schmidt-Wigger. Shallow Postmorphological Processing with KURD. In *Proceedings of NeMLaP/CoNLL*, Sydney, 1998.

[Dor93] Bonnie Jean Dorr. *Machine Translation: A View from the Lexicon*. MIT Press, Cambridge, Massachusetts. London, England, 1993.

[Foe73] Heinz von Foerster. On Constructing a Reality. In W.F.E Preiser, editor, *Environmental Design Research*. Dowden, Hutchinson & Ross, Stroudsburg, 1973.

[Foe86] Heinz von Foerster. Das Konstruieren einer Wirklichkeit. In Paul Wazlawick, editor, *Die Erfundene Wirklichkeit*. Piper, München, 1986.

[Goo72] Nelson Goodman. Seven Strictures on Similarity. In *Problems and Projects*. The Bobbs-Merrill Company, Indianapolis and New York, 1972.

[Hey96] Matthias Heyn. Integrating machine translation into translation memory systems. In *European Asscociation for Machine Translation - Workshop Proceedings*, pages 111–123, ISSCO, Geneva, 1996.

[Hut97] Alan Hutchinson. Metrics on Terms and Clauses. In *Proceedings of ECML-97*, pages 138–146. Springer, 1997.

[Maa96] Heinz-Dieter Maas. MPRO - Ein System zur Analyse und Synthese deutscher Wörter. In Roland Hausser, editor, *Linguistische Verifikation, Sprache und Information*. Max Niemeyer Verlag, Tübingen, 1996.

[NBD94] Sergei Nirenburg, S. Beale, and C. Domashnev. A full-text experiment in example-based Machine Translation. In *International Conference on New Methods in Language Processing (NeMLaP) 94*, pages 78–87, Manchester, September 1994.

[Pla95] E. Plaza. Cases as terms: a feature term approach to the structured representation of terms. In *First International Conference ICCBR-95*, LNAI, Sesimbra, Portugal, October 95 1995. Springer.

[Ric95] Michael M. Richter. The Knowledge Contained in Similarity Measures. http://wwwagr.informatik.uni-kl.de/˜lsa/CBR/Richtericcbr95remarks.html, 1995.

[Sch91] J. Siegfried Schmidt. *Gedächtnis: Probleme und Perspektiven der interdisziplinären Gedächtnisforschung*. Surkamp, Frankfurt, 1991.

[SN90] S. Sato and M. Nagao. Towards memory based translation. In *COLING-90*, 1990.

[Str96] Oliver Streiter. *Linguistic Modeling for Multilingual Machine Translation*. Informatik. Shaker Verlag, Aachen, 1996.

[WD95] D. Wettschereck and T.G Dietterich. An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. *Machine Learning 19, 5-28*, 19, 1995.

[WK95] Peter Whitelock and Kieran Kilby. *Linguistic and computational techniques in Machine Translation system design*. Computational Linguistics. UCL Press, London, 1995.