CBR in Dependency-based Machine Translation

S. Zwarts, A. Nijholt, R. op den Akker & M. Poel
University of Twente, Department of Computer Science
PO Box 217, 7500 AE Enschede, The Netherlands

Abstract

A case based reasoning approach is introduced as a learning technique in the domain of machine translation of natural language. In our approach syntactical and semantic features are part of the cases in the case-base. To implement this, dependency analysers of sentences in the source and target languages are used. The case-base is filled with a learning mechanism that uses a parallel corpus of sentences with their translations. This case-base is used to make new translations.

Introduction

Progress in machine translation research is slow. Being able to deal with syntactic, semantic and pragmatic information available in a sentence to be translated and its global and local context is a prerequisite for translation. It requires models to represent this information, whatever the domain is, and it requires reasoning methods using this knowledge and taking into account similar knowledge of a sentence to be generated in the target language.

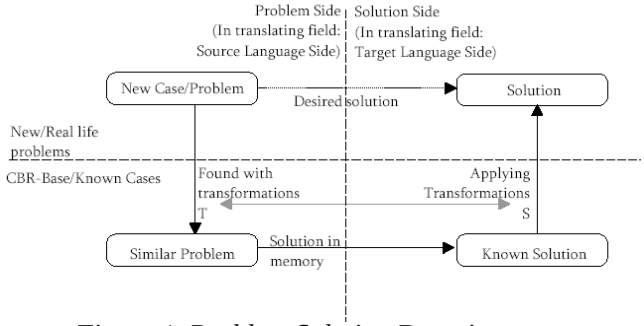

Figure 1. Problem Solution Domain

Many traditional projects in machine translation have dealt with defining rules for the transfer of syntactic or semantic representations from source language to target language. More recent corpus-based approaches to machine translation use statistical metrics to choose most probable transfers or use pattern matching techniques (e.g. in example and memory-based translation). But again, also with these approaches we can choose to have transfer at different levels of representation.

In this paper we look at a corpus-based approach using Case-Based Reasoning (CBR). CBR has been used before for machine translation purposes by Sato and Nagao (1990), but our research is based strictly on the CBR algorithm of Aamodt and Plaze (1994). They have defined a framework for CBR that can be adopted to our purposes. Generally, in CBR new cases have to be matched against previously recorded cases and then an attempt is made to build new solutions based on the solutions of the recorded cases. The cases in our approach are constructed by dependency parsers.

In this project the language pair Dutch - English is used. This article is based on the research described in the Master thesis of Zwarts (2003).

1 Translation using CBR

Our translating technique satisfies the three increasing criteria for defining Example-Based Translation as defined by Somers (1999): 1) It uses a bilingual corpus, 2) It uses a bilingual corpus as its main knowledge base, 3) It uses a bilingual corpus as its main knowledge base, at run-time.

The proposed algorithm uses a bilingual corpus, performs a syntactical grammar analysis on the sentences in the corpus and stores the sentences as well as all the sentence parts in a knowledge base. This base is used in the actual translating process. Filling this base, performing the syntactical grammar analysis, is called the training phase. The utilisation of this base is called the translating phase.

Figure 1 illustrates what we want to achieve. The New Case is a new sentence offered for translation. The system builds a transformation T which projects the new case on an existing case. The existing case has a known solution, which is used to construct the final solution. On the known solution for the existing solution a transformation S is performed which leads to the final solution. Assumed is that the functions T and S are related and S is uniquely determinable from T and the existing case with its solution.
In Aamodt and Plaze (1994) four separate phases in CBR are distinguished: Retrieve, Reuse, Revise, and Retain. Since CBR only operates on cases, a process should be present which builds cases with a well-defined case-structure from the input. The next section deals with building these cases.

2 Dependency Analysis

In order to build case structures we use a head driven dependency analysis. In the resulting tree each node, not limited to leaf nodes, hold a chunk of text from their mother node. The root node holds the entire sentence. Every node in this tree has three properties. The lexical value, the part of speech (only non empty for leaf nodes) and the grammatical role. These three values together form all the cases, in the training phase this is done for both the source as the target language, the generation phase only for the source language. The dependency parser that we use here is the Dutch Dependency Parser described in Bouma (2001).

3 Similarity

Now that we have the cases we can define one of the critical issues in CBR, the similarity value. This similarity value is used both in the Revise phase of the training session and in the Retrieve phase of the translation session. If the similarity is calculated wrongly, the selected case for reuse is wrong and so is the solution linked to this case. It is very hard to correct this afterwards.

3.1 Similarity in the training phase

Analysis of the training consists of storing the information of the dependency trees and in the alignment from the source to the target domain of the cases with all the sub cases for every sentence pair. This alignment relies heavily on the similarity value of the cases. This similarity value is calculated for every sub case in the source language with every sub case in the target language. Starting with the highest values, the sub cases from the source are aligned with the corresponding sub case from the target language until a certain threshold value is reached. The performance of this link is stored with this similarity value.
To calculate this similarity value the following definitions are given:
$\#Translations_{ij}$ is the number of occurring possible translations from words in $case_i$ with words in $case_j$ using a dictionary lookup.
$\#case_x$ is the number of occurring words in $case_x$.
$\#TranslatedTags_{ij}$ is the number of DEP tags from the children of $case_j$ if they occur in the mapping of all the DEP tags from the children of $case_i$.
$\#Subtags_x$ is the number of children of $case_x$
p and q are heuristic values between 0 and 1.
p weights grammar-similarity to word-similarity, q maintag to subtags. In this phase p,q are constant.
Now, let α_{ij} denote the similarity value in the training phase from sub case i in the source language to sub case j in the target language α_{ij} is defined:

$$\alpha_{ij} = (p\ \alpha Grammar_{ij}) + ((1-p)\ \alpha words_{ij})$$

$$\alpha words_{ij} = \#Translations_{ij}/(\max(\#case_i, \#case_j))$$

$$\alpha grammar_{ij} = (q\ \alpha maintag_{ij}) + ((1-q)\ \alpha subtags_{ij})$$

$\alpha maintag_{ij}$ = 1 if DEP from $case_i$ is equal to the mapping of DEP from $case_j$, 0 otherwise.

$$\alpha subtags_{ij} = \#TranslatedTags/(\max(\#Subtags_i, \#Subtags_j))$$

α_{ij} is a value in the domain [0,1] describing how similar a source case is to a target case. The idea is to weight the grammar (structure) and the literal word meaning. The grammatical part describes how many identical grammatical annotations are found, the other part how many matching words. The max operator makes long sentences less similar when compared to short sentences and visa versa.
α_{ij} is defined as 1 for root cases since it is given that these sentences are aligned.

A dictionary is used for the word matching, which preferable holds all the possible translations.
When possible a tag map is used, that maps grammatical annotations from the dependency grammar of the source language to that of the dependency grammar of the target language if the both Grammars use different tags.

In CBR this entire training session is represented by the phases Revise and Retain.

3.2 Similarity in the Retrieve phase

In the Retrieve phase a similarity calculation is used which is quite quite similar to the training phase. The similarity value is used to Retrieve the most similar case from the case-base to the sentence to translate, the assumption is that similar sentences have a similar translation.
The difference with the alignment phase is that this time the similarity values are between cases in the same language. When calculating the similarity of two cases the transformation is calculated as well. Which tells how to change the given problem in a known problem. (In Figure 1 the function T) The calculation of these transformations is discussed in the next section.

To calculate this similarity the following definitions are made:
$\#Equaltags_{ij}$ is the number of DEP tags from the children of $case_j$ equal to the DEP tags from the children of $case_i$.
$\#EqualAfterTransformation_{ij}$ is the number of DEP tags from the children of $case_i$ and $case_j$ that match after a transformation is applied.
Again p and q are a heuristic values between 0 and 1. For leaf nodes q is zero and $\alpha subtags_{ij}$ is not defined.
Because whole sentences are more useful as an example of grammar and simple words only have a use in direct literal meaning level p is slowly increased. On sentence level the emphasis is on the grammatical side, because larger parts can be reused, and on word level the emphasis is on words only, because only the literal word meaning can be reused.
Now, if α_{ij} is the similarity value between $case_i$ from the case-base and $case_j$ is the new case/problem which we have to solve then the similarity is defined:

$\alpha_{ij} = (p\ \alpha Grammar_{ij}) + ((1-p)\ \alpha words_{ij})$
$\alpha Grammar_{ij} = (q\ \alpha maintag_{ij}) + ((1-q)\ \alpha subtags_{ij})$
$\alpha maintag_{ij}$= 1 if DEP from $case_i$ is equal to the DEP from $case_j$, 0 otherwise.
$\alpha subtags_{ij} = (\#Equaltags_{ij} + (Transformationcost_{ij}\ \#equalAftertransformation_{ij}))/ (\max(\#Subtags_i, \#Subtags_j))$

3.3 Transformation Calculation

Because the given cases put for translations are almost never exactly the same to known cases, with solutions from the case-base, transformations should be derived. The transformations tell how the given case is different from the selected case from the case-base. The assumption is that it also has a relation between the solution we have from the known case and the solution we want to have for our given case. Because transformations are hierarchical, (a subcase from the case in question can be replaced by the result of an earlier transformation on a (sub)case etc.) a transformation list is defined.
transformlist = [transform]
transform = (c_1, c_2, c_3, ts)
Here c_1,c_2 and c_3 are (sub)cases. ts is a new transformlist.
This basically means in c_1 the case c_2 should be replaced by case c_3 where c_3 is transformed using ts.

4 Translation Generation

In the Translation phase the phases Retrieve and Reuse from the CBR phase are adopted.
The Retrieve phase calculated the most probable case from the case base to Reuse. Because it is too complex to calculate the similarity from the new constructed case with all the known cases from the case base a pre selection is done. First, this similarity value is calculated with only the cases from the knowledge base which have an equal amount of children and children with the same DEP tags. If after this the similarity is not above a certain threshold the demand for similar DEP tags is waived. If still no matching case from the case base is found also cases which one child more or one less are considered.

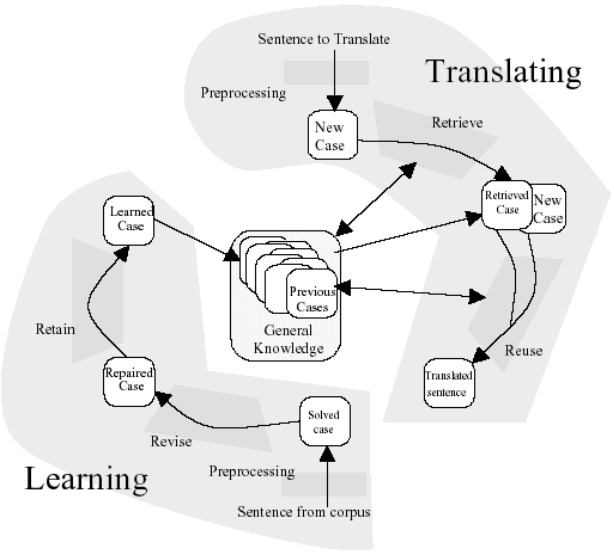

Figure 2. New CBR cycle

The result of the Retrieve phase is not only a case which should be suitable for Reuse but also a transformation list.

The case with the highest calculated α_{ij} is selected, where $Case_i$ is the case which is new and offered for translation. $Case_j$ is the selected one for reuse.

$Case_j$ has a known solution in the knowledge base. This solution/translation of $Case_j$ is used as the initial translation for the given case.

The transformation part is now important. Without transformations, we already have a solution, but when there are transformations we should "translate" this as well. This means to build the transformation S from the transformation T, which are both in Figure 1.

Every case used in the transformation list should have aligned parts, otherwise it is useless in this phase and another alternative should be used (the next similar option from the retrieve phase). But when the cases have alignment, the algorithm looks for the literal text from one of the aligned cases and tries to find this text in the (initial) translation. If not found, the next alignment is used, because cases can be aligned to more than one other case, if they are synonyms. If no match is found, the algorithm skips to the next similar case from the Retrieve phase. If a match is found, this part of the sentence is replaced by the replacement value from the transformation.

Conclusions

The testing system only achieved good results on a rather small domain, however the results on this domain have some interesting characteristics. The system does have to handle collocations in language in a special way, they are just treated as all other chunks. Because the system tries to work on the largest chunks available collocations are treated in the whole and translated without breaking them down in pieces. Because it is translated as one, the meaning is not lost. It is interesting to note how examples are used to overcome grammatical differences in the languages. If one known sentence shows that it is translated with a different grammatical structure in the target language, the system tries a new unknown sentence to reflect this grammatical change.

The algorithm as described here has a specific problem with storage. Because it is an example based learning strategy lots of examples are stored in the base. When no pruning strategy is used, all the examples which are used during the training session are stored in the Case Base. This is characteristic for example based learning strategies. Because no pruning strategy is yet used in the prototype the Case Base tends to grow rather quick, and in the Transformation Phase the whole Case Base must be stored in memory which makes the whole CBR process rather slow.

References

Aamodt, A. and Plaze, E. (1994) Case-based reasoning: Foundational issues methodological variations, and system approaches, AI Communications 7, pp. 39-59.

Bouma, G., van Noord, G. and Malouf, R. (2001) Alpino: Wide-coverage computational analysis of Dutch, 2001. Also available on http://odur.let.rug.nl/~gosse/papers.html.

Kay, M. (1989) Head driven parsing. In Proceedings of Workshop on Parsing Technologies, Pittsburg.

Sato, S. and Nagao, M. (1990) Towards memory based Translation. COLING 1990 Vol. 1: 13th Intern. Conf. on Computational Linguistics, Dept. of Electrical Engineering, Kyote University.

Somers, H. (1999). Example-based Machine Translation. Machine Translation, 14 (20), pp. 113-157.

Zwarts, S (2003), Using CBR as a learning technique for natural language translations, Master's Thesis, University of Twente