MetaMorpho TM: a linguistically enriched translation memory

Gábor Hodász and **Gábor Pohl**
Pázmány Péter Catholic University
Faculty of Information Technology
Budapest, Hungary
{hodasz, pohl}@morphologic.hu

24th September, 2005.

Introduction - 1

- Translation Memory (TM):
  the most important task is:

  searching for similar segments that are translated previously

- Example Based Machine Translation (EBMT):

  building segments from pieces in the TM which makes the Memory more reusable

  - what is a piece?
  - how to build?

Introduction - 2

- widely used search algorithms:
  - character based similarity
  - fuzzy index
  - language independent
  - not effective in the case of agglutinative languages (like hungarian)
  - quick and small algorithms

Our solution - 1

Linguistic based similarity measure:

- based on Levenshtein distance
- calculated on tokens
- uses morphological tagging

- PRO:
  - gives better result, especially in the case of agglutinative languages
- CON:
  - language dependent
  - morphological ambiguity

Our solution - 2

The use of sub-sentential segments

- noun phrases (NPs)
- morphological tagging
- automatic NP alignment

- PRO:
  - answers can be modular
  - and can be linguistically evaluated and transformed
- CON:
  - difficult task to determine a NP
  - language dependent

Levels of Analysis

- Word-level analysis
  - stemming and morphology
- NP parsing
  - shallow structure retained only
- Sentence skeletons
  - NPs are substituted by NP slots
  - with certain properties (features) registered

Process overview - 1

add new entry:

- analyse segments on both sides (SL, TL)
  - > NPs, skeletons
  - > morphological tagging
- align NPs
- store in DB:
  - skeleton pairs
  - NP pairs

Process overview - 2

entry lookup:

- analyse segment on SL side
- searching:
  - similar whole sentences
  - similar sentence skeletons
  - similar NPs
- building a tiled sentence from components
- morphological transformation according to the search sentence

Linguistic Similarity

- based on Levenshtein distance
- on morphologically analysed tokens
- calculated on 3 levels of similarity:
  - surface form (L1)
  - lemma (L2)
  - class (L3)
- ambiguity is managed with a POS tagger (Brill)

Multilayer Similarity Measure between sequences of terms

$$ED(\sigma(S_1),\sigma(S_2))=$$
$$[ed_{Li}(\sigma(S_{1,Li}),\sigma(S_{2,Li})),ed_{Li-1}(\sigma(S_{1,Li-1}),\sigma(S_{2,Li-1})),\ldots,$$
$$ed_{L1}(\sigma(S_{1,L1}),\sigma(S_{2,L1}))]$$

where

S_{n,L_i} - the i[th] similarity layer of the n[th] segment (sentence or NP)

$\sigma(S)$ - the sequence of tokens

$ed(S_i,S_j)$ - the Levenshtein distance

$ED(X,Y)$ - the similarity vector

English-Hungarian NP-alignment

Previous Work

- Corpus-based (offline) methods
  - word alignment models
  - Julian Kupiec: An Algorithm for finding Noun Phrase Correspondences in Bilingual Corpora. (ACL 1993)
    - simple NP chunk alignment
- Parse tree alignment
  - Groves, D, Hearne, M and Way, A.: Robust Sub-Sentential Alignment of Phrase-Structure Trees. (COLING'04)
    - EN-FR tree alignment

En-Hu NP-alignment
Reasons for developing a new means

- Requirements
  - speed
  - accuracy
- Statistics-based tools
  - require large corpora
  - offline processing
  - Hungarian has a rich morphology
    - stemming helps – but which stem to choose?
- Parse tree alignment
  - (So far) only for languages with similar parse trees.

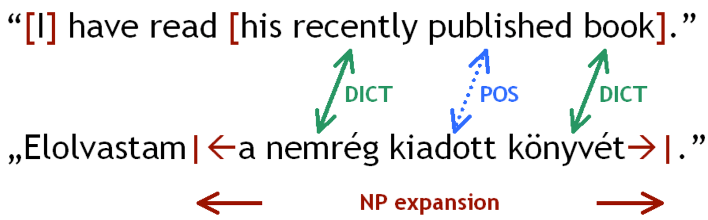
➔ Dictionary and POS-based alignment of parsed English and Hungarian NPs.

En-Hu NP-alignment
NP similarity score

- For each possible NP pair we calculate a heuristic matching score from the number of tokens matched by:
  - dictionary-based matching
    - stemmed search
    - expressions covered by longer ones are filtered (e.g. inside "hard disk drive")
  - cognate matching
  - POS-matching among lexically unmatched tokens
  - number of unmatched content words and ignored grammar words (PRON, DET, etc.)

En-Hu NP-alignment
How to find NPs?

- English side:
  - MetaMorpho English parser
- Hungarian side:
  - MetaMorpho Hungarian parser (still in early development) ➔ bad precision / recall
  - Guess Hungarian NP candidates corresponding to the parsed English NPs using dictionary matching, cognate matching, POS matching and a simple Hungarian NP grammar.



En-Hu NP-alignment
First results with guessed Hungarian NPs

- We used a relatively small dictionary
  - 116,000 word/expression pairs
- 40 test sentence pairs from a translated book on computer networks
  - average sentence length: 23 words
- Only **56%** of parsed English NPs had an alignable translation in the Hungarian side.
- **Alignment precision: 84%**
  - 91% without sentence pairs where more than half of the NPs were translated to VPs.
- **Alignment recall: 65%**

Implementation

- Relational database (MySQL)
  - storing aligned sentences and NPs
- Similarity calculation and NP alignment implemented in C++
- fully functioning graphical user interface implemented in C#

Conclusion

Our approach attempts at providing significantly higher quality translations:

- using **3 level similarity** on tokens
- using **aligned sub-sentential segments** (NPs and sentence skeletons)
- building a **tiled** suggestion of translation
- with **morphologically correct answer generation**

Future work

- Indexing algorithm to improve performance
- Automatic (offline) dictionary building to extend alignment dictionary
- Evaluation
  - Selecting the proper method
  - Doing the work
- Integration of a terminology management system
- Fallback to traditional fuzzy indexing

Благодаря!