The GREYC Machine Translation Memory for the IWSLT 2009 campaign: one step beyond translation memory

Yves Lepage Adrien Lardilleux Julien Gosme

firstname.lastname@info.unicaen.fr

GREYC, University of Caen Basse-Normandie, France

System, tasks and conditions

System: one step beyond translation memory

- start from translation memory principle
- improve with translation by analogy principle

Tasks: all BTEC read speech tasks

- Arabic → English
- Turkish → English
- Chinese → English

Conditions: use only data delivered by the organizers

- use `devset1` and `devset2` (shared by all tasks) as test set for preliminary experiments
- merge development sets with training data for primary runs

Priniciple of translation by analogy

Solve all possible analogical equations of the type:

$$A : x :: C : D$$

where A is the input sentence, and C and D are from the training data (sentences or alignment phrases). If $x = B$ is a solution and B belongs to the training data, then we know its translation $\hat{B}$.

Knowing $\hat{B}$, solve:

$$y : \hat{B} :: \hat{C} : \hat{D}$$

Any solution $y = \hat{A}$ is a translation hypothesis for A.

PRINICIPLE OF TRANSLATION MEMORY

Pick up sentence B from the training data that is closest to input sentence A and output its translation $\hat{B}$.

QUESTION: can we go one step further by modifying translation output $\hat{B}$, so that it reflects, in the target language, the modification of B into A in the source language?

ANSWER: apply transformations to B so that it becomes closer and closer to A. For that, use the translation by analogy principle.

One step beyond translation memory

1. Pick up sentence B_0 from training data closest to input sentence A.

2. Alter B_0 into B_1 by applying transformations illustrated by pairs $C \rightarrow D$ from training data or alignment tables.
   Try to make B_1 closer to A than B_0 is by imposing constraints.

$$\begin{array}{ccccccc} B_1 & : & B_0 & :: & C & : & D \\ \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow \\ \hat{B}_1 & : & \hat{B}_0 & :: & \hat{C} & : & \hat{D} \end{array} \quad \begin{array}{l} \text{with } C\ (\neq \varepsilon) \sqsubset A \text{ and } D \text{ (possibly } \varepsilon) \sqsubset B_0 \\ \\ \text{with } \hat{D} \sqsubset \hat{B}_0 \end{array}$$

3. Apply process recursively to $(B_n, \hat{B}_n)$.
   Verify $d(A, B_{n+1}) < d(A, B_n)$ at each step.

4. Output $B = argmin\ d(B_i, A)$. In case of multiple hypotheses, pick up hypothesis with lowest recursion level and highest frequency.

Results (1): improvement over a translation memory

Conditions: `devset1` and `devset2`, BLEU scores no_case+no_punc.

Track	translation memory	this system	increase
BTEC_AE	0.35	**0.38**	+0.03
BTEC_CE	0.31	**0.32**	+0.01
BTEC_TE	0.38	**0.40**	+0.02

Results (2): analysis of behavior

Conditions: primary runs on test set

Track	# of unknown words in test set	# of sentences matched in translation memory	totally translated	partially translated	back-off to translation memory
BTEC_AE	189	44	148	245	32
BTEC_CE	105	66	51	213	139
BTEC_TE	134	70	144	198	57

Results (3): standard evaluation scores

Conditions: primary runs on test set

Track	case+punc	bleu	meteor	f1	prec	recl	wer	per	ter	gtm	nist
BTEC_AE	yes	0.329	0.617	0.686	0.734	0.644	0.512	0.453	43.262	0.661	5.654
BTEC_AE	no	0,307	0,566	0,633	0,688	0,587	0,587	0,510	48,836	0,623	5,536
BTEC_CE	yes	0.280	0.554	0.613	0.637	0.590	0.592	0.532	51.609	0.596	5.657
BTEC_CE	no	0,277	0,510	0,564	0,591	0,539	0,655	0,579	57,212	0,565	5,927
BTEC_TE	yes	0.355	0.648	0.708	0.745	0.674	0.509	0.437	41.633	0.678	6.347
BTEC_TE	no	0.346	0.600	0.658	0.704	0.617	0.570	0.484	47.052	0.644	6.425

Conclusion

- last in all tracks... ☹
- slight improvement over a translation memory
- maybe not enough data for a more significant improvement?
- maybe not the right unit of processing: would chunks be better?

Appendix (1): pre- and post-processing units

Language	Lower-case	Isolated punctuation	Processing unit
Arabic	nr	yes	hyperwords
Chinese	nr	nr	characters
Turkish	yes	yes	words
English	yes	yes	words

- for Chinese, preliminary experiments delivered better results in characters than in words (+1.34 BLEU points on `devset1` and `devset2`)
- on English output, Moses recaser and detokeniser

Appendix (2): morphological synthesizer

Translation of unknown words using same technique as [Denoual, 2007] or [Langlais & al., 2006], but here we generate all possible words from all words in the data using an analogy solver written in Python:

$$
\begin{array}{ccccccccc}
x & : & b & :: & c & : & d & \Rightarrow & x = a \\
 & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow \\
\hat{x} & : & \hat{b} & :: & \hat{c} & : & \hat{d} & \Rightarrow & \hat{x} = \hat{a}
\end{array}
$$

Track	total number of word-to-word alignments produced where the source word is a new word	number of unique new source words	# of translations / new word	# of new words that are unknown words in test set	# of unknown words in test set
BTEC_AE	4,852,505	3,140,013	1.6	84	189
BTEC_CE	73,997	54,728	1.4	0	105
BTEC_TE	9,609,402	7,109,448	1.4	66	134

Appendix (3): translation tables

Results of preliminary experiments for the three language pairs on `devset1` and `devset2` (devsets common to the 3 tasks):

- GIZA++ beats *anymalign* for Arabic and Turkish
- *anymalign* beats GIZA++ for Chinese