Multi-Engine Machine Translation by Recursive Sentence Decomposition

Bart Mellebeek
School of Computing
Dublin City University
Dublin 9, Ireland
mellebeek@computing.dcu.ie

Karolina Owczarzak
School of Computing
Dublin City University
Dublin 9, Ireland
owczarzak@computing.dcu.ie

Josef Van Genabith
School of Computing
Dublin City University
Dublin 9, Ireland
josef@computing.dcu.ie

Andy Way
School of Computing
Dublin City University
Dublin 9, Ireland
away@computing.dcu.ie

Abstract

In this paper, we present a novel approach to combine the outputs of multiple MT engines into a consensus translation. In contrast to previous Multi-Engine Machine Translation (MEMT) techniques, we do not rely on word alignments of output hypotheses, but prepare the input sentence for multi-engine processing. We do this by using a recursive decomposition algorithm that produces simple chunks as input to the MT engines. A consensus translation is produced by combining the best chunk translations, selected through majority voting, a trigram language model score and a confidence score assigned to each MT engine. We report statistically significant relative improvements of up to 9% BLEU score in experiments (English→Spanish) carried out on an 800-sentence test set extracted from the Penn-II Treebank.

1 Introduction

In this paper, we present a novel approach to combine the outputs of multiple MT engines into a consensus translation.

Multi-Engine Machine Translation (MEMT) is a term coined by (Frederking and Nirenburg, 1994), who were the first to apply the idea of a multi-engine approach in Natural Language Processing to Machine Translation (MT). Researchers in other areas of language technology such as Speech Recognition (Fiscus, 1997), Text Categorization (Larkey and Croft, 1996) and POS Tagging (Roth and Zelenko, 1998) have also experimented with multi-system approaches. Since then, several researchers in the MT community have come up with different techniques to calculate consensus translations from multiple MT engines (cf. section 2). All these previously proposed techniques share one important characteristic: they translate the entire input sentence *as is* and operate on the resulting target language sentences to calculate a consensus output. Their main difference lies in the method they use to compute word alignments between the multiple output sentences.

In contrast to previous MEMT approaches, the technique we present does not rely on word alignments of target language sentences, but is based on a recursive chunking algorithm that produces simple constituents as input to the MT engines. The outputs of these syntactically meaningful chunks are compared to each other and the highest ranked translations are used to compose the output sentence. Our approach, therefore, prepares the input sentence for multi-engine processing on the input side. It draws its strength from the simple fact that short input strings result in better translations than longer ones.

The paper is organised as follows. In section 2, we

give a short overview of the most relevant MEMT techniques. We explain our approach in section 3 and demonstrate it with a worked example. Section 4 contains the description, results and analysis of our experiments. We give avenues for future research in section 5 and summarize our findings in section 6.

2 Related Research

(Frederking and Nirenburg, 1994) produced the first MEMT system (Pangloss) by combining the output sentences of three different MT engines, developed in house. In order to calculate a consensus translation, the authors rely on their knowledge of the inner workings of the engines.

In (Nomoto, 2004), by contrast, the MT engines are treated as black boxes. He presents a number of statistical confidence models, based on a large array of language models and the IBM1 translation model (Brown et al., 1993) to select the best output string at sentence level.

Most of the other recent approaches to MEMT rely on word alignment techniques in the translation hypotheses to infer the units for comparison between the MT systems. (Bangalore et al., 2001) produces alignments between the different hypotheses using edit distance (Levenshtein, 1965). For each aligned unit, a winner is calculated by majority voting and a N-gram language model. Since edit distance only focuses on insertions, deletions and substitutions, the model cannot handle translation hypotheses with a significantly different word order. (Jayaraman and Lavie, 2005) try to overcome this problem by introducing a method that can find non-monotone alignments. They compose a consensus from these alignments by using a language model and confidence score specific to each MT engine. (van Zaanen and Somers, 2005) present 'Democrat', a 'plug-and-play' MEMT system that relies solely on a simple edit distance-based alignment of the translation hypotheses and does not use additional heuristics to compute the consensus translation. Finally, (Matusov et al., 2006) use well-established techniques from the Statistical MT community to produce alignments of hypotheses based on pairwise word alignments in an entire corpus instead of at the sentence level.

To date, to the best of our knowledge, all previously known MEMT proposals operate on MT output for complete input sentences. In the research presented here, we pursue a different approach: we decompose MT input into chunks, choose the best chunk translation and recompose the translated chunks in output.

3 Description of the Algorithm

Given N different MT engines ($E_1 \ldots E_N$), the proposed method recursively decomposes an input sentence S into M syntactically meaningful chunks $C_1 \ldots C_M$. Each chunk C_i ($1 < i < M$) is embedded in a minimal necessary context and translated by all MT engines. For each chunk C_i, the translated output candidates $C_i^1 - C_i^N$ are retrieved and a winner C_i^{best} is calculated based on majority voting, a language model trained on a large target language corpus and a confidence score assigned to each MT engine. In a final step, the output sentence S' is composed by assembling all C_i^{best} ($1 < i < M$) in their correct target position. A flow chart representing the entire MEMT architecture can be found in Figure 1.

The decomposition into chunks, the tracking of the output chunks in target and the final composition of the output are based on the TransBooster architecture presented in (Mellebeek et al., 2005).

In the following subsections, we will explain the decomposition of the input sentence, the translation of the input chunks, the calculation of the best output chunk and the composition of the output sentence. We will also demonstrate the approach with a worked example.

3.1 Decomposition of Input

Our approach presupposes the existence of some sort of syntactic analysis of the input sentence. We report experiments on human parse-annotated sentences (the Penn II Treebank (Marcus et al., 1994)) and on the output of two state-of-the-art statistical parsers (Charniak, 2000; Bikel, 2002) in section 4.

In a first step, the input sentence is decomposed into a number of syntactically meaningful chunks as in (1).

(1) [*ARG*$_1$] [*ADJ*$_1$]...[*ARG*$_L$] [*ADJ*$_l$] **pivot** [*ARG*$_{L+1}$] [*ADJ*$_{l+1}$]...[*ARG*$_{L+R}$] [*ADJ*$_{l+r}$]

where **pivot** = the nucleus of the sentence, *ARG* =

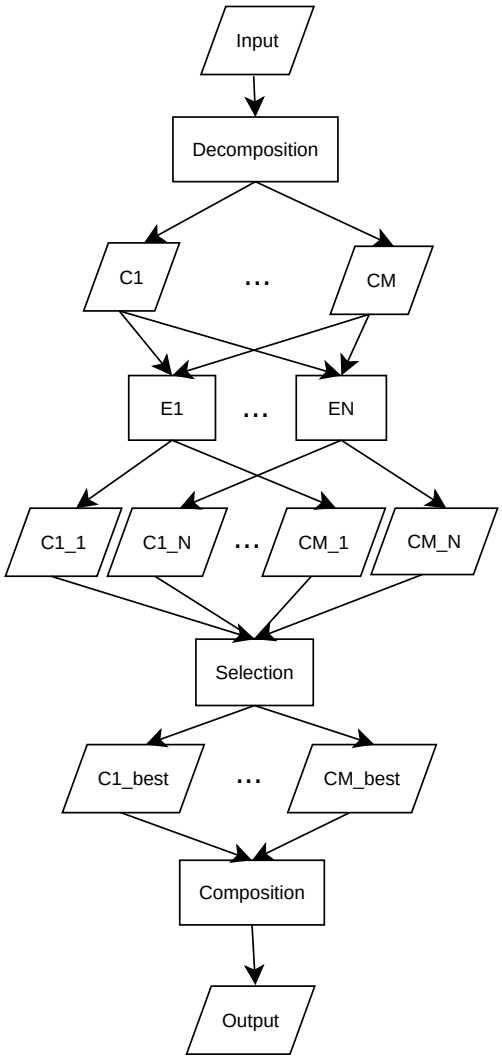

Figure 1: A flow chart of the entire MEMT system, with C_i the i^{th} input chunk ($1 < i < M$), E_j the j^{th} MT engine ($1 < j < N$) and C_{i_j} the translation of C_i by E_j.

argument, *ADJ* = adjunct, {l,r} = number of *ADJ*s to left/right of **pivot**, and {L,R} = number of *ARG*s to left/right of **pivot**.

In order to determine the pivot, we compute the head of the local tree by adapting the head-lexicalised grammar annotation scheme of (Magerman, 1995). In certain cases, we derive a 'complex pivot' consisting of the head terminal together with some of its neighbours, e.g. phrasal verbs or strings of auxiliaries. The procedure used for argument/adjunct identification is an adapted version of Hockenmaier's algorithm for CCG (Hockenmaier, 2003). The result of this first step on a worked example can be seen in (5).

In a next step, we replace the arguments by similar but simpler strings, which we call 'Substitution Variables'. The purpose of Substitution Variables is: (i) to help to reduce the complexity of the original arguments, which often leads to an improved translation of the pivot; (ii) to help keep track of the location of the translation of the arguments in target. In choosing an optimal Substitution Variable for a constituent, there exists a trade-off between accuracy and retrievability. 'Static' or previously defined Substitution Variables (e.g. 'cars' to replace the NP 'fast and confidential deals' as explained in section 3.5) are easy to track in target, since their translation by a specific MT engine is known in advance, but they might distort the translation of the pivot because of syntactic/semantic differences with the original constituent. 'Dynamic' Substitution Variables comprise the real heads of the constituent (e.g. 'deals' to replace the NP 'fast and confidential deals' as outlined in section 3.5) guarantee a maximum similarity, but are more difficult to track in target. Our algorithm employs Dynamic Substitution Variables first and automatically backs off to Static Substitution Variables if problems occur. By replacing the arguments by their Substitution Variables and leaving out the adjuncts in (1), we obtain the skeleton in (2)

(2) $[V_{ARG_1}]$ ...$[V_{ARG_L}]$ **pivot** $[V_{ARG_{L+1}}]$ ...$[V_{ARG_{L+R}}]$

where V_{ARG_i} is the simpler string substituting ARG_i

By matching the previously established translations of the Substitution Variables V_{ARG_i} ($1 <= i <= L + R$) in the translation of the skeleton in (2), we are able to (i) extract the translation of the pivot and (ii) track the location of the translated arguments in target. The result of this second step on the worked example is shown in (6).

Adjuncts are located in target by using a similar strategy in which adjunct Substitution Variables are added to the skeleton in (2).

3.2 Translation of Input Chunks in Context

Since translating individual chunks out of context is likely to produce a deficient output or lead to bound-

ary friction, we need to ensure that each chunk is translated in a simple context that mimics the original. As in the case of the Substitution Variables, this context can be static (a previously established template, the translation of which is known in advance) or dynamic (a simpler version of the original context).

Our approach is based on the idea that by reducing the complexity of the original context, the analysis modules of the MT engines are more likely to produce a better translation of the input chunk C_i than if it were left intact in the original sentence, which contains more syntactic and semantic ambiguities. In other words, we try to improve on the translation C_i^j of chunk C_i by MT engine j through input simplification. (cf. section 3.5 for more details)

After obtaining the translations of all input chunks by all MT engines ($C_i^1 - C_i^N$), all that remains to be done is to select the best output translation C_i^{best} for each chunk C_i and derive the output by composing all C_i^{best}. This is possible since we have kept track of the position of each C_i^j by the Substitution Variables.

3.3 Selection of the Best Output Chunk

The selection of the best translation C_i^{best} for each input chunk C_i is based on three heuristics.

1. *Majority Voting*. Since identical translations by different MT systems are a good indicator of the relative quality of the candidate translations $C_i^1 - C_i^N$, the translation that was produced by the highest number of MT engines is considered to be the best. If no clear winner is found at this stage, a language model score will select the best translation between the remaining candidates.

2. *Language Modeling*. In case Majority Voting produces more than 1 candidate translation, the translation among the selected candidates with the best language model score is considered to be the best. This score is an approximation of the likelihood of the hypothesis translation in the target language and therefore rewards fluency. We used the SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1988), on 213M words of target language text.[1]

3. *Confidence Score*. In the rare cases that no winner is found by either of the previous two heuristics, the best translation is the one produced by the MT engine that obtained the highest BLEU score on the entire development corpus.

3.4 Composition of Output

The input decomposition procedure is recursively applied to each constituent until a certain threshold is reached. Constituents below this threshold are sent to the MT engines for translation, the best of which is selected as described in section 3.3. Currently, the threshold is related to the number of lexical items that each node dominates. Its optimal value depends on the syntactic environment of the constituent and is empirically established. After all constituents have been decomposed and a best output translation has been selected, they are recombined to yield the target string output to the user.

This recombination is performed by recursively substituting the retrieved translation of the constituents (cf. section 3.2) for the translated Substitution Variables in (2). In case the baseline MT engines use a different reordering of Substitution Variables, we select the reordering of the MT engine that obtained the highest BLEU score on the entire development corpus.

3.5 A Worked Example

Consider the following input sentence:

(3) The chairman, a long-time rival of Bill Gates, likes fast and confidential deals.

The parsed output by (Charniak, 2000)

[1] The entire training section of the Spanish Europarl Corpus augmented with a corpus of a Spanish newspaper ('La Vanguardia').

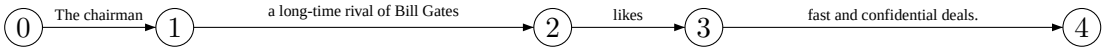

Figure 2: Decomposition of Input.

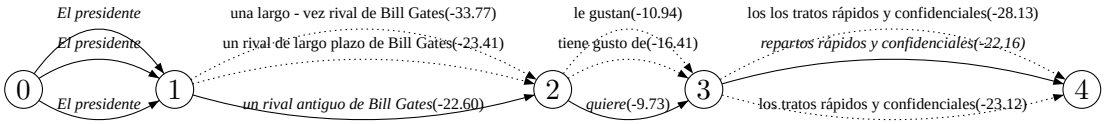

Figure 3: Selection of best output chunk. The optimal combination follows the arcs in full.

(4) (S (NP (NP (DT the) (NN chairman)) (, ,) (NP (NP (DT a) (JJ long-time) (NN rival)) (PP (IN of) (NP (NNP Bill) (NNP Gates)))) (, ,)) (VP (VBZ likes) (NP (ADJP (JJ fast) (CC and) (JJ confidential)) (NNS deals))) (. .))

is used as input to the decomposition module. In a first step, the pivot, arguments and adjuncts are calculated.

(5) [The chairman, a long-time rival of Bill Gates,]$_{ARG1}$ [likes]$_{pivot}$ [fast and confidential deals]$_{ARG2}$.

In a second step, the arguments are replaced by syntactically simpler Substitution Variables.

(6) [The chairman]$_{VARG1}$ [likes]$_{pivot}$ [deals]$_{VARG2}$.

The resulting string is translated by the MT engines. For example, the translation produced by Systran is

(7) El presidente tiene gusto de repartos.

This translation allows us (i) to extract the translation of the pivot (ii) to determine the location of the translated arguments. This is possible because we determine the translations of the Substitution Variables (*the chairman*, *deals*) at runtime. If these translations are not found in (7), we replace the arguments by previously defined Static Substitution Variables. For example, in (5), we replace *'The chairman, a long-time rival of Bill Gates'* by *'The man'* and *'fast and confidential deals'* by *'cars'*. In case the translations of the Static Substitution Variables are not found (7), we interrupt the decomposition and have the entire input string (3) translated by the MT engine.

We now apply the procedure recursively to the identified chunks *'The chairman, a long-time rival of Bill Gates'* and *'fast and confidential deals'*.

Since the chunk *'fast and confidential deals'* contains fewer words than a previously set threshold - this threshold depends on the number of leaf nodes and the syntactic nature of the input - it is ready to be translated by the MT engines. As explained in subsection 3.2, the chunk has to be embedded in an appropriate context. Again, we can determine the context dynamically (e.g. *The chairman likes*) or use a static predefined context template (e.g. *The man sees*). As mentioned in section 3.2, static context templates are previously established templates that mimick the original context of the chunk to be translated. Their exact nature depends on the syntactic environment of the candidate chunk. For example, the previously mentioned template *'The man sees'* mimicks a direct object context for an NP.

(8) shows how the chunk *'fast and confidential deals'* is embedded in a Dynamic Context.

(8) [The chairman likes]$_{DynamicContext}$ [fast and confidential deals]$_{ARG_2}$.

This string is sent to the MT engines for translation. For example, the translation produced by Systran is

(9) El presidente tiene gusto de repartos rápidos y confidenciales.

Like Dynamic Substitution Variables, the translations of Dynamic Contexts are determined at runtime. If we find the translation of the Dynamic Context in (9), it is easy to deduce the translation of the chunk *'fast and confidential deals'*. If, on the contrary, the translation of the Dynamic Context is not found in (9), we back off to a previously defined Static Context template (e.g. *The man sees*). In case the translation of this context is not found either, we back off to translating the input chunk *'fast and confidential deals'* without context.

Since the remaining chunk *'The chairman, a long-time rival of Bill Gates'* contains more words than a previously set threshold, it is judged too complex for direct translation. The decomposition and translation procedure is now recursively applied to this chunk: it is decomposed into smaller chunks, which may or may not be suited for direct translation, and so forth.

The recursive decomposition algorithm splits the initial input string into a number of optimal chunks, which are translated by all MT engines as described above. A simple graph representation of the full decomposition of the input sentence is shown in Figure 2. The recovered translations with logprob language model scores are shown in Figure 3. From these, the best translations (in italics) are selected as described in subsection 3.3.

Table 1 shows that the MEMT combination outperforms the outputs produced by Systran and LogoMedia and is similar in quality to the output produced by SDL. Note that our approach is not limited to a blind combination of previously produced output chunks. In the case of Systran, the complexity reduction of the input leads the system to improve on its *own* translation. In the complete translation (Table 1), Systran erroneously analyses the verb *'likes'* as a noun (→*'gustos'*) and identifies the adjective *'fast'* as a verb (→*'ayuna'*). By contrast, examples (8) and (9) show that submitting the chunk *'fast and confidential deals'* in a simplified context improves the translation of the adjective *'fast'* from the erroneous *'ayuna'* in the original translation of the entire sentence by Systran to the correct *'rápidos'*. Also, the translation of the verb *'likes'* improves to *'tiene gustos de'*, which can only contribute to a better overall MEMT score.

Original	The chairman, a long-time rival of Bill Gates, likes fast and confidential deals.
LogoMedia	Al presidente, un rival de mucho tiempo de Bill Gates, les gustan los los tratos rápidos y confidenciales
Systran	El presidente, rival de largo plazo de Bill Gates, gustos ayuna y los repartos confidenciales.
SDL	El presidente, un rival antiguo de Bill Gates, quiere los tratos rápidos y confidenciales.
MEMT	El presidente, un rival antiguo de Bill Gates, quiere repartos rápidos y confidenciales.

Table 1: Example sentence 1: MEMT result vs. the original MT engines

Original	Mr. Pierce said Elcotel should realize a minimum of $10 of recurring net earnings for each machine each month.
LogoMedia	El Sr. Pierce dijo que Elcotel debe ganar *a minimum of* $10 de ganancias netas *se repitiendo* para cada máquina todos los meses.
Systran	Sr. *Elcotel* dicho *Pierce* debe realizar un mínimo de $10 de las ganancias netas que se repiten para cada máquina cada mes.
SDL	Sr. *Perfora* dijo que Elcotel debe darse cuenta de un mínimo de $10 de ganancias netas peridicas para cada máquina cada mes.
MEMT	El Sr. Pierce dijo Elcotel debe realizar un mínimo de $10 de las ganancias netas que se repiten para cada máquina cada mes.

Table 2: Example sentence 2: MEMT result vs. the original MT engines

Table 2 contains the output of a second example sentence, where the MEMT combination clearly outperforms the individual MT engine contributions. LogoMedia leaves *'a minimum of'* untranslated and uses a grammatically incorrect gerund *'se repitiendo'*. Systran switches the target positions of *'Pierce'* and *'Elcotel'*, which severely distorts the accuracy of the translation. SDL interprets *'Pierce'* as a verb, which makes the translation unintelligible. The MEMT combination, however, combines the best parts of each engine and is both accurate and relatively fluent.

4 Evaluation

4.1 Experimental Setup

To test the performance of our algorithm, we translated an 800-sentence test set (min. 1 word, max. 54 words, ave. 19.75 words), randomly extracted from Section 23 of the Penn-II Treebank, by three online MT systems (LogoMedia[2], Systran[3], SDL[4]) from English→Spanish. Groups of 200 sentences from the test set were translated by four native speakers of Spanish, each of whom was a certified translator, in order to obtain a set of reference translations for use with automatic evaluation metrics.

We used three different syntactic analyses of the test set as input to our algorithm.

1. The original human parse-annotated Penn-II Treebank structure.

2. The output parse of the test set by (Charniak, 2000).

3. The output parse of the test set by (Bikel, 2002).

In each of these three cases, our algorithm decomposed the input into chunks and combined the chunk outputs of the MT engines as described in section 3.

4.2 Results

In Tables 3–6, we compare automatic evaluation results (BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and F-Score (Turian et al., 2003)) of the MEMT output against the original output of the 3 MT systems.

MEMT1 refers to the results of the MEMT algorithm on original Penn-II Treebank trees. MEMT2 and MEMT3 refer to the MEMT results based on the input parsed by (Charniak, 2000) and (Bikel, 2002), respectively.

	BLEU	NIST	F-Score
LogoMedia	0.3140	7.3272	0.5627
Systran	0.3003	7.1674	0.5553
SDL	0.3037	7.2792	0.5663
MEMT1	0.3295	7.6822	0.5802
MEMT2	0.3209	7.5865	0.5744
MEMT3	0.3178	7.5658	0.5731

Table 3: MEMT results vs. the original MT engines - absolute scores

	BLEU	NIST	F-Score
LogoMedia	104.9	104.8	103.1
Systran	109.7	107.1	104.4
SDL	108.4	105.5	102.4

Table 4: MEMT1 vs. the original MT engines - relative scores %

	BLEU	NIST	F-Score
LogoMedia	102.1	103.5	102.0
Systran	106.8	105.8	103.4
SDL	105.6	104.2	101.4

Table 5: MEMT2 vs. the original MT engines - relative scores %

	BLEU	NIST	F-Score
LogoMedia	101.2	103.2	101.8
Systran	105.8	105.5	103.2
SDL	104.6	103.9	101.2

Table 6: MEMT3 vs. the original MT engines - relative scores %

Table 3 contains the absolute scores of all three baseline MT systems and the MEMT combinations on original Penn-II Treebank trees and the output of

[2]www.lec.com
[3]www.systransoft.co.uk
[4]www.freetranslation.com

the two statistical parsers. Tables 4-6 contain the relative scores of the three MEMT runs against the baseline MT systems.

In each case, the statistical significance[5] of BLEU and NIST was established by using the BLEU/NIST resampling toolkit described in (Zhang and Vogel, 2004). In all three experiments, the MEMT combination significantly outperforms each of the contributing MT engines. There are two main factors that explain the improvements over the baseline systems:

1. The selection procedure (cf. subsection 3.3) eliminates bad chunk translations. This is a characteristic shared by all MEMT approaches. To use the words of (Frederking and Nirenburg, 1994): "three heads are better than one".

2. The decomposition of the input sentence into syntactically simpler chunks allows the individual MT systems to improve on their *own* translations. This is the main novelty of our approach in comparison with previous MEMT techniques. Since short input chunks contain fewer ambiguities than the original longer sentences, the MT systems are more likely to analyse the input correctly, which can lead to an inproved output. When this is not the case, the selection procedure is a safe guarantee that a reasonable consensus translation will be produced.

As expected, the scores based on parser output are slightly lower than the scores based on human parse-annotated sentences, but the differences are minimal and even the worst MEMT experiment obtains a relative rise of 1.2%-5.8% BLEU score with respect to the baseline systems.

5 Future research

Proposals for future research include the following:

- Experiment with a variety of language models. (Nomoto, 2004).

- Replace the similarity measure used in the selection procedure (cf. subsection 3.3) by an edit distance metric.

- It would be interesting to see whether a word graph-based MEMT consensus at the level of the output chunks has the potential of improving our approach. Instead of simply selecting the best output chunk based on the previously described heuristics (cf. subsection 3.3), an existing MEMT approach could be used to form a word-graph consensus translation at chunk level.

6 Conclusions

We have presented a novel approach to Multi-Engine Machine Translation that, in contrast to previous proposals in this area, does not exclusively rely on target sentence combination. Our approach is based on a recursive decomposition of the input sentence into smaller chunks which are more likely to be correctly translated than the longer input sentence. A selection procedure based on majority voting, a language model score and a confidence score assigned to each MT engine finds the best translation hypothesis for each input chunk. The best chunk translations are then recomposed into target language sentences. In experiments (English→Spanish) carried out on an 800-sentence test set extracted from the Penn-II Treebank, we report statistically significant relative improvements of up to 9% BLEU score.

Aknowledgements

This work was made possible by Enterprise Ireland grant #SC/2003/0282. We would like to thank the reviewers for their insightful comments which served to improve this paper.

References

S. Bangalore, G. Bordel, and G. Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 351–354, Trento, Italy.

D. M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of Human Language Technology Conference (HLT 2002)*, pages 24–27, San Diego, CA.

P. F. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine

[5] measured on 2000 resampled test sets in a 95% confidence interval

translation: parameter estimation. *Computational Linguistics*, pages 263–311.

E. Charniak. 2000. A maximum entropy inspired parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*, pages 132–139, Seattle, WA.

S.F. Chen and J. Goodman. 1988. An empirical study of smoothing techniques for language modeling. Technical report tr-10-98, Center for Research in Computing Technology (Harvard University).

G. Doddington. 2002. Automatic evaluation of MT quality using N-gram co-occurrence statistics. pages 128–132, San Diego.

J. G. Fiscus. 1997. A post-processing system to yield reduced word error rates: recognizer output voting error reduction (rover). In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 238–245, Santa Barbara, CA.

R. Frederking and S. Nirenburg. 1994. Three heads are better than one. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 95–100, Stuttgart, Germany.

J. Hockenmaier. 2003. Parsing with Generative models of Predicate-Argument Structure. In *Proceedings of the ACL 2003*, pages 359–366, Sapporo, Japan.

S. Jayaraman and A. Lavie. 2005. Multi-Engine machine translation guided by explicit word matching. In *Proceedings of the 10th Conference of the European Association for Machine Translation*, pages 143–152, Budapest, Hungary.

L. Larkey and B. Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 289–297, Zurich, Switzerland.

Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSR, 163(4)*, pages 845–848.

D. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, MA.

M. Marcus, G. Kim, M.A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Human Language Technology workshop*, pages 114–119, Plainsboro, NJ.

E. Matusov, N. Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *11th Conference of the European Chapter of the Association for Computational Lingustics*, pages 33–40, Trento, Italy.

B. Mellebeek, A. Khasin, K. Owczarzak, J. Van Genabith, and A. Way. 2005. Improving online machine translation systems. In *Proceedings of MT Summit X*, pages 290–297, Phuket, Thailand.

T. Nomoto. 2004. Multi-Engine machine translation with voted language model. In *Proceedings of the 42nd Conference of the Association for Computational Linguistics (ACL)*, pages 494–501, Barcelona, Spain.

K. Papineni, S. Roukos, T. Ward, and W-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. pages 311–318, Philadelphia, PA.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING) and 36th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 1136–1142, Montreal, Canada.

A. Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, Denver, Co.

J. Turian, L. Shen, and D. Melamed. 2003. Evaluation of machine translation and its evaluation. In *Proceedings of MT Summit IX*, pages 386–393, New Orleans, LO.

M. van Zaanen and H. Somers. 2005. DEMOCRAT: deciding between multiple outputs created by automatic translation. In *Proceedings of the 10th Machine Translation Summit*, pages 173–180, Phuket, Thailand.

Y. Zhang and S. Vogel. 2004. Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of the Tenth Conference on Theoretical and Methodological Issues in Machine Translation*, pages 85–94, Baltimore, MD.