The GREYC Machine Translation System for the IWSLT 2008 Evaluation Campaign

Yves Lepage, Adrien Lardilleux, Julien Gosme and Jean-Luc Manguin

GREYC, University of Caen Basse-Normandie, France
firstname.lastname@info.unicaen.fr

Abstract

This year's GREYC machine translation (MT) system presents three major changes relative to the system presented during the previous campaign, while, of course, remaining a pure example-based MT system that exploits proportional analogies. Firstly, the analogy solver has been replaced with a truly non-deterministic one. Secondly, the engine has been re-engineered and a better control has been introduced. Thirdly, the data used for translation were the data provided by the organizers plus alignments obtained using a new alignment method. This year we chose to have the engine run with the word as the processing unit on the contrary to previous years where the processing unit used to be the character. The tracks the system participated in are all classic BTEC tracks (Arabic-English, Chinese-English and Chinese-Spanish) plus the so-called PIVOT task, where the test set had to be translated from Chinese into Spanish by way of English.

1. Introduction

This paper gives a sketch of the GREYC machine translation system that participated in the IWSLT 2008 evaluation campaign. This system is an on-going effort to re-engineer the ALEPH machine translation system presented at the IWSLT 2005 evaluation campaign [1] in the Python programming language. It incorporates three major changes over the system presented at the IWSLT 2007 evaluation campaign [2].

The system participated in all read speech tasks, *i.e.*, the three BTEC tasks with the following source-target languages: Arabic-English, Chinese-English, Chinese-Spanish, and the PIVOT task where the test set had to be translated from Chinese into Spanish by way of English. In each of these tasks, the translation of two test sets was performed, correct recognition result (CRR in the tables in Section 5) and 1-best output of read speech (ASR.1). The results obtained constitute our *primary* results and reflect our improvement (or regression for this year) in the development of our example-based MT engine.

As for comparison, and so to have an idea of the performance of our system against off-the-shelf tools, we performed translation of all the test sets for the above-mentioned tracks using the two tools listed on the IWSLT 2008 resources page,[1] Giza++ [3][2] and Moses [4][3]. These results were submitted as our *contrast2* result sets.

In the same spirit as for our participations in previous evaluation campaigns, we intentionally did not use any data outside of the training data provided by the organizers. This year, this constraint was imposed by the organizers to all participants. In the same vein, and in addition, we stress the fact that we did not either use any data from the development sets for the final runs.

In previous experiments and during a previous participation to IWSLT in 2005, the system was shown to be unable to translate in the absence of sufficient data. [5] reports a fall in BLEU from 0.53 with a training corpus of 160,000 aligned sentences to 0.42 when using only a quarter of the same training corpus. Last year, to address this problem, we compiled the training data in order to add n-grams and chunks extracted from the training data. This year, we inflated the input data by adding sub-sentential alignments that were obtained from the training data, using a new alignment technique that will be described in Section 4.2. This was the only tool that we used outside of the MT engine for the translation itself. We also submitted runs using Moses with translation tables that were obtained with this new alignment tool. These runs constitute our *contrast1* results and were obtained for comparison with the translations output by Giza++ and Moses (*contrast2*).

2. Preprocessing of the data

2.1. Encoding

In previous years, our translation engine used an analogy-solver programmed in C that required constant-length encoding for the data [6]. This year's re-engineering of the tool in Python alleviates this problem. The new engine deals directly with any encoding scheme, and of course `utf-8`, the encoding in which the data were provided by the organizers. Consequently, no encoding change of any type was required to handle this year's data.

[1] http://www.slc.atr.jp/IWSLT2008/archives/2008/10/resources.html
[2] http://code.google.com/p/giza-pp/
[3] http://www.statmt.org/moses

2.2. Sentence splitting

Previous experiments have confirmed the intuition that analogies are relatively much more important in number with short utterances [7]. In last year's campaign, we applied a pre-processing step in which we split utterances containing several sentences where possible (*i.e.*, when the number of sentences in the source and target languages was the same). This pre-processing was not performed this year, the reason being that we expected the alignment method to perform this task for free.

2.3. Punctuation normalization

All test sets that were released by the organizers appeared to be punctuation free and in lower-case, on the contrary to the training set, which were correctly transcribed sentences with punctuation and capitalized letters where due. We chose to eliminate any punctuation from the training data and to lowercase everything, and then to apply our alignment tool to the punctuation-free training data, so as to run the test sets against consistent data.

However, as the output required by the organizers for the purpose of official evaluation was data with exact case and punctuation, we applied Moses's detokeniser and recaser on all our outputs. For each task, the detokenizer was trained on exactly the same data as the data required for the task.

3. The translation principle

The translation engine used in this year is an extension of the one used in previous years' campaign ([1], [2]). Its principle has been described in [5]. It is an example-based engine that also backs off to a translation memory.

For a new sentence to translate, the engine first looks for the sentence in the training data. If the sentence is found in the training set, the translation engine just outputs its translation without any further computation. This is the most felicitous case.

If the sentence does not already exist in the training data, then computation is carried out using the principle of corresponding proportional analogies between two language domains. Figure 1 illustrates the principle with actual data. [6] thoroughly presents the notion of proportional analogies between strings of symbols, and inspects some properties in formal language theory that are driven by the fact that a recursive application of the principle amounts to making use of languages of analogical strings as defined in the same paper. Regarding this point, it is important to say, because this may explain part of our deceptive results, that this year we did not make use of the recursive application for engineering reasons.

We shall quickly recall the principle of translation by proportional analogy: to translate a new sentence, A, the engine basically solves all possible analogical equations of the type:

$$A \ : x :: C : D \tag{1}$$

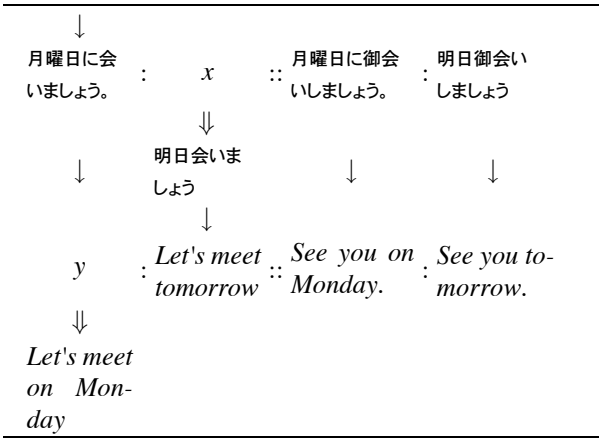

Figure 1: A sketch of the translation principle by corresponding proportional analogies, illustrated on actual data from IWSLT 2007. The Japanese sentence on the left is the input. The Japanese strings on the same line are from the training data or the alignment results. Solving the Japanese equation yields the Japanese string in the middle, which, incidentally, belongs to the alignment data. From the corresponding translations, a candidate translation for the input is obtained in place of y.

where C and D are two source sentences from the training data. If the solution of the equation $x = B$ belongs to the training data, then its translation $\hat{B}$ is known and the analogical equation:

$$y : \hat{B} :: \hat{C} : \hat{D} \tag{2}$$

can be built and possibly solved in the target language. The principle states that any solution $y = \hat{A}$ to this equation is a possible translation of A. There may exist a plurality of solutions for equation (1) as well as for equation (2). This year's analogy solver is non-deterministic and renders all the solutions in such cases.

When no solution at all can be found by analogy, the engine backs off to the basic behavior of a translation memory, *i.e.*, it outputs the translation of the source sentence closest to the input sentence.

4. Three major changes to the MT system

4.1. A non-deterministic analogy-solver

Previous versions of the MT system used an analogy solver programmed in C that was extremely fast. It has been described in [8] and [5]. Theoretically, an analogical equation may have no solution, one solution or several solutions. The version implemented in C was programmed in a way such that the first solution encountered by the program, if any, would be the only one outputted. This decision was in a good part responsible for the speed of the program.

The analogy solver that we used this year is truly non-deterministic and is a much slower implementation in Python.

Table 1: Distribution of the number of solutions according to whether they have no solution, one solution, or multiple solutions. The countings are summed over all translations in the three BTEC tasks.

Number of solutions	Number of analogies
No solution	3,090,087
One solution only	3,319,800
Multiple solutions	110,809

It also profits from new representational insights into proportional analogy.

The explanation of proportional analogy we use is that if $A : B :: C : D$ holds, then the set of following equalities must hold:

$$\begin{cases} \text{dist}(A,B) = \text{dist}(C,D) \\ \text{dist}(A,C) = \text{dist}(B,D) \\ |A|_a + |D|_a = |B|_a + |C|_a, \forall a \end{cases}$$

where $\text{dist}(A,B)$ is the canonical distance (insertion, deletion, no substitution) between strings A and B and where $|A|_a$ is the number of occurrences of character (or word) a in the string A.

The algorithm in C was based on a representation of analogical equations that used a matrix representation. This originates in the standard way of computing edit distances using the dynamic programming method of the Wagner and Fischer algorithm [9]. The new algorithm takes another view and uses a representation that consists of one string only onto which a series of reversals (a notion used in reversal edit distance [10]) is applied to compose the solution of the analogical equation. This comes from the observation that, if $A : B :: C : D$ holds, then there exists a series of reversals that transforms $B \bullet C^{-1}$ into $A \bullet D^{-1}$, where X^{-1} denotes the mirror string of X. This property enforces in a natural way the third constraint in the definition of analogy given above.

To take an example, a succession of reversals on the string: $unreachable \bullet tius$ will lead to a state where the string becomes $reach \bullet \mathbf{x}^{-1}$, that solves the analogical equation $reach : unreachable :: suit : \mathbf{x}$ as $\mathbf{x}$ has been built step by step during application of reversals. For instance, possible steps are:

$$\boxed{unreachable \bullet tius}$$
$$\boxed{suit \bullet elbahcaer}nu$$
$$reach\boxed{able \bullet}tiusnu$$
$$reach \bullet elbatiusnu$$

In this example, each string to be reversed is represented in a frame. This delivers the solution $unsuitable$, to be read on the last line from right to left after the symbol $\bullet$.

Table 2: Details of the distribution of the analogical equations with multiple solutions according to their number of solutions.

Number of solutions	Number of analogies	Percentage
2	62,624	56.5%
3	21,411	19.3%
4	11,691	10.6%
5	6,194	5.6%
6	4,785	4.3%
7	1,649	1.5%
8	825	0.7%
9	498	0.5%
≥ 10	1132	1.0%
total	110,809	100.0%

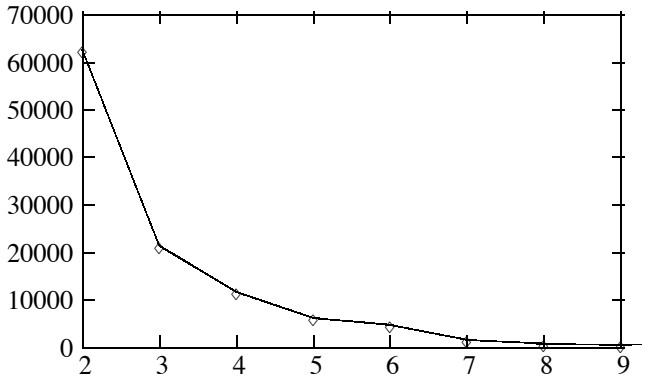

Figure 2: Distribution of the number of analogical equations with the same number of solutions. This graph plots the figures given in Table 2 that were obtained from the three runs on the BTEC tasks. In abscissae, the number of solutions. In ordinates, the corresponding number of analogies having the same number of solutions.

The non-determinism comes from the fact that, at each step, there may exist several possible reversals. Memorizing these reversals in a stack allows us to produce all possible solutions.

We inspected the number of solutions obtained when translating over all the three BTEC tasks. The result of this inspection is given in Tables 1 and 2. The first table shows that more than half of the analogical equations have one or more solutions. This point will be addressed again in Section 4.3 below. The second table shows that the distribution of the number of analogies decreases rapidly with the number of solutions (see also Figure 2).

4.2. A new alignment method

The new alignment method we used relies entirely on those alignments that get a maximal score of 1 according to any of the standard similarity measures like the cosine similarity, Jaccard index or Dice coefficient. We call such alignments *perfect alignments*. Basically, with this definition, perfect alignments contain just those (non-necessarily contiguous sequences of) words that appear exactly on the same lines.

As this is obviously not sufficient to align all (non-necessarily contiguous sequences of) words in a corpus, one has, so to say, to enforce perfect alignments. An idea to fill this requirement is to look at all possible subcorpora of different sizes. The method will thus reduce to looking for (non-necessarily contiguous sequences of) words that strictly appear on the same lines in any possible subcorpus of the initial corpus.

Looking for all possible subcorpora of a corpus is not feasible in practice, because there are 2^N possible subcorpora, in a corpus of N lines, N being several thousands in practice. The solution consists in resorting to sampling. This can be done iteratively, with various sample sizes. Practically, a full coverage of the corpus is ensured by partitioning. This permits fast processing while maintaining the subcorpora representative of the initial corpus as much as possible.

The alignment process is performed in several iterations. At each iteration, the initial corpus is randomly partitioned into M subcorpora of n lines. Iterations are independent.

For each subcorpus obtained by partitioning, we proceed as follows:

1. group all words with the same number of occurences on each line of the subcorpus;

2. for each group of words, go through the lines it appears in. Two sequences of words are extracted from each line:

   (a) the very group of words;

   (b) the context of the group of words in the line, *i.e.*, the line minus the group of words.

   We say that (a) delivers direct sequences, and that (b) delivers context sequences. Of course, word order is preserved in both types of extracted sequences. The sequences are not necessarily contiguous.

Any alignment may be obtained a plurality of times, from different iterations, different subcorpora and different lines. The result of the process is a list of alignments with the count of the number of times they have been obtained.

Finally, each alignment is weighted by an observed probability reflecting the number of times it has been obtained during the alignment generation process. The observed probability that a sequence of words S_i in language i translates into S_j in language j is the number of alignments that exactly contain both sequences in the respective languages, $C(S_j, S_i)$, over the total number of alignments where S_i appears, $C(S_i)$:

$$P(S_j|S_i) = \frac{C(S_j, S_i)}{C(S_i)}$$

This is similar to the way Koehn *et al.* estimate phrase translation probabilities [11]. Consequently, the proposed technique outputs translation tables directly usable by a statistical machine translation software, like Moses. However, our tables contain no lexical weightings.

We tested the above-described alignment method against Giza++ on the task of translating half of the development set number 3 that has been released with the training data. Our alignment tables just replaced those alignment tables obtained from Giza++, the next steps for machine translation (training and tuning) being performed using Moses in exactly the same way. The results are given in Table 3 and they show that our alignment method comes close to the scores obtained with Giza++, being even superior in one of the tasks, the Chinese-to-Spanish one.

Table 3: BLEU scores obtained with Moses using alignment tables obtained with Giza++ and with our method. The first half of development set 3 was used for tuning (253 lines). The second half was used as a test set (253 lines also). 16 references were used for evaluation.

BTEC Task	Translation tables obtained from Giza++ (IBM model 4)	Our method
Arabic to English	**0.47**	0.45
Chinese to English	**0.39**	0.37
Chinese to Spanish	0.28	**0.30**

4.3. Re-engineering of the engine

A re-engineering of our example-based engine is an ongoing work. As for programming languages, we moved from C to Python. As for conceptual differences, a better control has been introduced.

The new engine can process both character sequences and word sequences. In previous years, the unit of processing was the character, whereas we opted for the word this year. This choice was made in order to reduce the average length of the lines to be translated (counted in unit of processing), so as to also reduce the processing time.

As described above in Section 3, the basic task of the engine is to form analogical equations of type (1) involving the sentence to be translated by selecting two other strings from the input data (in our current setting for IWSLT, the training data plus alignment data). Using the same notation again, given an input sentence A, two sentences C and D have to be selected so as to form the following analogical

equation:

$$A : \mathbf{x} :: C : D$$

The size of input data amounts to more than ten hundred thousand lines, resulting in more than ten billion possible analogical equations. It is thus necessary to resort to a heuristic to select those most promising analogical equations.

The heuristic we rely on, is based on on two observations. Firstly, strings which exhibit smaller prefix or suffix differences relative to the input string A seem better candidates for C. Following this observation, we use the notion of longest common substring, noted *LCSubstr*.[4] The heuristic thus consists in choosing candidate strings C and D by using two sorts applied in a row: firstly, the set of the input data is sorted by decreasing order of lengths of *LCSubstr* in common with the sentence A to be translated. Secondly, the set of input data is sorted anew in decreasing order of the length of the *LCSubstr* in common between C and D.

As we look for substrings in the entire source language data, and as a way to speed up the process, we chose to limit the search to substrings of length $|S|-1, |S|-2$ for all strings of length $|S|$ in the source data. In addition, we also consider all n-grams, with $n = 1, 2, 3$. The limits the search space to a size linear in the size of the source data.

A benefit of the previous heuristic has been to increase the proportion of analogical equations that deliver a solution. [5] reported a proportion of only 28% of analogical equations with a solution. The same measure on all three BTEC tracks shows that this ratio raised up to 52% (see Table 1).

Another purpose in re-engineering the engine was to introduce more control on the order in which analogical equations are processed. A controller has been designed, that dynamically re-ranks analogical equations to be solved during the translation process. Each analogical equation, in the source language as well as in the target language, is assigned a priority value. Target language analogical equations get a higher priority than source language ones, because reaching for a translation as soon as possible is the primary goal of the engine. Also, source language equations involving strings closer to the sentence to be translated are given a higher priority because they have intuitively more chances to lead to a translation than other ones. From the engineering point of view, the controller stands as a separate module, allowing us to possibly re-define the priority function.

5. Results

The scores obtained by the system described above are shown in Tables 4, 5, 6 and 7 for all the three BTEC tracks, Arabic-to-Chinese, Chinese-to-English and Chinese-to-Spanish, and for the PIVOT track, Chinese-English-Spanish. The first score is BLEU, the second one is METEOR.

In addition to the scores obtained by the system, the tables contain the scores obtained by our two contrastive runs. The

[4]Caution: the *LCSubstr* is one contiguous substring, not to be confused with longest common subsequence.

Table 4: Scores for the three runs in the BTEC AE track.

BTEC_AE (case+punc)			
primary	ASR.1	0.19	0.42
	CRR	0.22	0.46
contrast1	ASR.1	0.28	0.51
	CRR	0.37	0.58
contrast2	ASR.1	0.32	0.56
	CRR	0.41	0.63
BTEC_AE (no_case+no_punc)			
primary	ASR.1	0.22	0.41
	CRR	0.26	0.45
contrast1	ASR.1	0.31	0.50
	CRR	0.41	0.57
contrast2	ASR.1	0.35	0.55
	CRR	0.46	0.63

Table 5: Scores for the three runs in the BTEC CE track.

BTEC_CE (case+punc)			
primary	ASR.1	0.20	0.44
	CRR	0.21	0.45
contrast1	ASR.1	0.28	0.51
	CRR	0.32	0.54
contrast2	ASR.1	0.29	0.53
	CRR	0.32	0.56
BTEC_CE (no_case+no_punc)			
primary	ASR.1	0.23	0.42
	CRR	0.24	0.43
contrast1	ASR.1	0.31	0.50
	CRR	0.35	0.53
contrast2	ASR.1	0.31	0.53
	CRR	0.35	0.56

first contrastive runs (*contrast1*) were obtained using our new alignment method instead of Giza++, followed by the use of Moses. The second contrastive runs (*contrast2*) were obtained by a standard application of Giza++ and Moses. In the case of the PIVOT track, we were not able to produce a *contrast2* run, as Giza++ would crash for reasons we were unable to fix.

In all cases, in BLEU and in METEOR, the ranking is *primary* < *contrast1* $\leq$ *contrast2*, except for the Chinese-to-Spanish BLEU scores in the case+punc conditions where *contrast2* < *contrast1*.

The results obtained on these runs are in the trend of previous experiments we performed with various data sets to test our new alignment technique: in these previous experiments, our alignment tool, used as a replacement for Giza++, allowed us to obtain similar BLEU scores or even slightly better scores, depending on the data and the language pairs considered. However, the differences in scores observed here with IWSLT 2008 data show slightly larger gaps than in these previous experiments.

If one sees the *contrast2* runs as a kind of baseline, then the conclusion is that the current state of changes we introduced to our engine did not allow us to even reach the baseline of statistical machine translation.

The main explanation of such results is due in good part for the fact that our engine was not finished at the time when the test sets were released. As we mentioned in the introduction, a major flaw in our current engine is the lack of recursion in the translation process.

6. Conclusion

This paper has given a sketch of the GREYC machine translation system that participated in the IWSLT 2008 evaluation campaign in all BTEC tasks and the PIVOT one. The system is an example-based system that makes use of the principle of corresponding proportional analogies between two languages.

In conformity to what we did in previous participations, we did not use any data outside of the training data provided by the organizers, a condition that was given by the organizers this year. In addition, and again, in conformity to what we did in previous participations, we did not use the development sets.

We tested, with no success, three changes to our system: the introduction of a non-deterministic version of the algorithm for solving analogical equations, a new alignment method and a re-engineering of the translation engine. The combination of all these changes did not allow us to obtain results comparable to those obtained with the standard application of Giza++ and Moses that produces a baseline statistical machine translation system, as the scores obtained on our primary and contrastive runs show.

Table 6: Scores for the three runs in the BTEC CS track.

BTEC_CS (case+punc)			
primary	ASR.1	0.19	0.23
	CRR	0.19	0.23
contrast1	ASR.1	0.22	0.26
	CRR	0.25	0.28
contrast2	ASR.1	0.22	0.27
	CRR	0.24	0.28
BTEC_CS ((no_case+no_punc))			
primary	ASR.1	0.20	0.23
	CRR	0.21	0.24
contrast1	ASR.1	0.24	0.27
	CRR	0.24	0.28
contrast2	ASR.1	0.27	0.29
	CRR	0.27	0.29

Table 7: Scores for the three runs in the PIVOT CES track.

PIVOT_CES (case+punc)			
primary	ASR.1	0.16	0.21
	CRR	0.17	0.22
contrast1	ASR.1	0.24	0.26
	CRR	0.27	0.28
PIVOT_CES (no_case+no_punc)			
primary	ASR.1	0.18	0.20
	CRR	0.19	0.21
contrast1	ASR.1	0.26	0.26
	CRR	0.29	0.28

7. References

[1] Y. Lepage and E. Denoual, ``Aleph: an EBMT system based on the preservation of proportional analogies between sentences across languages,'' in *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2005)*, Pittsburgh, PA., Oct. 2005, pp. 47--54. [Online]. Available: http://www.slt.atr.co.jp/~lepage/pdf/iwslt2005.pdf.gz

[2] Y. Lepage and A. Lardilleux, ``The GREYC machine translation system for the IWSLT 2007 evaluation campaign,'' in *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2005)*, Trento, 2007, pp. 49--54.

[3] F. J. Och and H. Ney, ``A systematic comparison of various statistical alignment models,'' *Computational Linguistics*, vol. 29, no. 1, pp. 19--51, 2003.

[4] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, ``Moses: Open source toolkit for statistical machine translation,'' in *Proceddings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, Prague, Czech Republic, 2007.

[5] Y. Lepage and E. Denoual, ``Purest ever example-based machine translation: detailed presentation and assessment,'' *Machine Translation*, vol. 19, pp. 251--282, 2005.

[6] Y. Lepage, ``Analogy and formal languages,'' *Electronic notes in theoretical computer science*, vol. 47, pp. 180--191, Apr. 2004.

[7] Y. Lepage, J. Migeot, and E. Guillerm, ``Analogies of form between chunks in Japanese are massive and far from being misleading,'' in *Proceedings of the 4th language and technology conference*, Poznan, 2007, pp. 503--507.

[8] Y. Lepage, ``Solving analogies on words: an algorithm,'' in *Proceedings of COLING-ACL'98*, vol. I, Montreal, Aug. 1998, pp. 728--735. [Online]. Available: http://www.slt.atr.co.jp/ lepage/pdf/coling98.pdf.gz

[9] R. A. Wagner and M. J. Fischer, ``The string-to-string correction problem,'' *Journal for the Association of Computing Machinery*, vol. 21, no. 1, pp. 168--173, Jan. 1974. [Online]. Available: http://portal.acm.org/citation.cfm?id=321811

[10] S. Hannenhalli and P. A. Pevzner, ``Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals,'' *J. ACM*, vol. 46, no. 1, pp. 1--27, 1999.

[11] P. Koehn, F. J. Och, and D. Marcu, ``Statistical phrase-based translation,'' in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003)*, vol. 1, 2003, pp. 48--54.

[12] T. Eck and C. Hori, ``Overview of the IWSLT 2005 evaluation campaign,'' in *Proc. of the International Workshop on Spoken Language Translation*. Pittsburgh, PA., 2005, pp. 1--22.