A Flexible Architecture for CAT Applications

Saša Hasan, Shahram Khadivi, Richard Zens, and Hermann Ney

Lehrstuhl für Informatik VI – Computer Science Department,
RWTH Aachen University, D-52056 Aachen, Germany

{hasan,khadivi,zens,ney}@cs.rwth-aachen.de

Abstract

We present an intuitive technical framework for making Computer Assisted Translation (CAT) adaptable and more suitable for rapid application development. The framework is a client-server-based architecture that uses an approach similar to "message passing", a technique widely used in computer science. We define a "translation object", a structure holding all necessary data, that is passed to server-like processes via sockets. This method can be easily enhanced in a modular manner where several recipients build a chain, one passing the processed object to the next one. We enhance a state-of-the-art phrase-based translation system with server and interactive generation capabilities and evaluate this prototype on different language pairs.

1 Introduction

Computer Assisted Translation (CAT) aims at helping professional translators to faster translate texts from one language into another. The broad term covers many aspects, reaching from electronic dictionaries, terminology databases, automatic translation systems and other modules, such as translation memories. A crucial component is the machine translation system, as it imposes most of the computation and memory requirement constraints. Obviously, a separation of the translator's environment and a dedicated translation server is intelligible (Och, Zens, & Ney, 2003).

Generally, there might be additional components involved in the overall translation process, such as preprocessing, on-the-fly reranking and eventual postprocessing (e.g. truecasing). We present a straightforward framework that allows for several modules to be connected in series, employing a common interface and defined data structures as input and output. Thus, the overall maintenance effort is facilitated.

The idea is to use *translation objects* that hold all necessary information and pass them from one application to another. For flexibility reasons and ease of use, we choose TCP/IP sockets to accomplish this task. Socket modules are available in all major programming languages, such as C++ or Python. Each program therefore has to incorporate only a small set of basic capabilities, i.e. receiving, parsing and sending the object, in order to be usable in the application chain. One major advantage is that many such modules can be provided by different research groups and easily set up for experimentation. By using TCP/IP, the servers even do not need to be in one intranet but can be located anywhere on the internet instead.

Furthermore, to test our basic framework, we incorporated server-like capabilities and an interactive search mode in a state-of-the-art phrase-based machine translation (MT) system (Zens et al., 2005). The current performance is similar to an interactive machine translation (IMT) system based on alignment templates.

Related work in the CAT domain is referred to in the next section. In Section 3, we review the theoretical framework for interactive machine translation being derived from a statistical MT viewpoint. In Section 4, we present a detailed overview on the techni-

cal architecture, whereas Section 5 addresses some preliminary experiments using a state-of-the-art phrase-based machine translation system within the presented framework. We conclude the paper in Section 6.

2 Related work

A multi-level design of interactive machine translation was already suggested by (Melby, 1983) based on work presented in (Kay, 1980). The main idea is to provide an environment with interactive capabilities to a human translator that suggests extensions of a partly translated sentence. The user can either accept or reject these completions. A recent implementation of such a tool was performed within the TransType project (Foster, Isabelle, & Plamondon, 1996, 1997; Langlais, Foster, & Lapalme, 2000). The assistance tool was then refined for the TransType2 project (Esteban, Lorenzo, Valderrábanos, & Lapalme, 2004). Furthermore, an earlier prototype demonstrating this concept was already presented in (Isabelle et al., 1993).

In this paper, we enhance a phrase-based SMT system with interactive search capabilities. Another phrase-based approach using alignment templates was presented in (Och et al., 2003). It uses a word-graph as a compact representation of the search space and locates nodes that correspond to word sequences with minimum edit distance to a given prefix. An investigation on different search strategies based on this approach is reported in (Bender, Hasan, Vilar, Zens, & Ney, 2005).

Other approaches use stochastic finite-state transducers that represent weighted graphs and, thus, efficiently code possible source-target sentence pairs in a compact manner (Civera et al., 2004).

3 Machine translation engine

In this section, we shortly summarize the theoretical background of an interactive statistical machine translation system. First, we review the underlying non-interactive SMT part. Then, we describe the translation model for interactive machine translation from a statistical viewpoint. We also present an extension that allows for arbitrary text as input, without limitation of the phrases to a specific test corpus. Thus, the system is capable of being employed in a real-world translation environment.

3.1 Baseline statistical machine translation system

In statistical machine translation, we are given a source language sentence $f_1^J = f_1 \ldots f_j \ldots f_J$, which is to be translated into a target language sentence $e_1^I = e_1 \ldots e_i \ldots e_I$. Among all possible target language sentences, we will choose the sentence with the highest probability:

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I | f_1^J) \right\} \quad (1)$$

The posterior probability $Pr(e_1^I | f_1^J)$ is modeled directly using a log-linear combination of several models (Och & Ney, 2002):

$$Pr(e_1^I | f_1^J) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum_{I', e'^{I'}_1} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e'^{I'}_1, f_1^J)\right)} \quad (2)$$

The denominator represents a normalization factor that depends only on the source sentence f_1^J. Therefore, we can omit it during the search process. As a decision rule, we obtain:

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J) \right\} \quad (3)$$

This approach is a generalization of the source-channel approach (Brown et al., 1990). It has the advantage that additional models $h(\cdot)$ can be easily integrated into the overall system. The model scaling factors λ_1^M are trained with respect to the final translation quality measured by an error criterion (Och, 2003).

We use a state-of-the-art phrase-based translation system including the following models: an n-gram language model, a phrase translation model and a word-based

lexicon model. The latter two models are used for both directions: $p(f|e)$ and $p(e|f)$. Additionally, we use a word penalty and a phrase penalty. The reordering model of the baseline system is distance-based, i.e. it assigns costs based on the distance from the end position of a phrase to the start position of the next phrase.

3.2 Interactive machine translation

In interactive machine translation, we have to find an extension e_{i+1}^I for a given prefix e_1^i. Hence, we constrain the search to those sentences e_1^I which contain e_1^i as prefix:

$$\hat{e}_{i+1}^{\hat{I}} = \underset{I, e_{i+1}^I}{\operatorname{argmax}} \left\{ Pr(e_{i+1}^I | e_1^i, f_1^J) \right\} \quad (4)$$

Thus, we maximize over all possible extensions e_{i+1}^I. For simplicity, this equation is formulated on the word level. We do not include the case where the prefix contains the first characters of the word e_{i+1}. In that case, we have to optimize over all target language words e_{i+1} that have the same word prefix. In the actual implementation, the method is applied on the character level, and the search for an extension can be performed after each keystroke of the human translator.

A crucial factor is an efficient maximization of Eq. 4, because human translators will only accept response times of fractions of a second. Using state-of-the-art search algorithms this is not achievable without putting up with a large number of search errors. To overcome this problem, we can compute a word graph which represents a subset of possible extensions (Ney & Aubert, 1994; Ueffing, Och, & Ney, 2002). The generation is then constrained to this set of extensions. For a more detailed description of this word-graph based approach to interactive machine translation, see (Och et al., 2003).

3.3 Dynamically loaded phrase table

It is a common approach in phrase-based systems to limit the phrase table to a specific test corpus. This results in a significant reduction of the size of the table and enables the usage of long phrases. The disadvantage is that a new phrase table has to be generated for previously unknown source sentences. As the generation of a phrase table is very time consuming, this approach is not feasible for an interactive application.

To overcome this limitation, we generate a phrase table that contains all phrases from the training corpus up to a certain length (in our case about five or six source words). Experiments have shown that the use of phrases beyond that length results only in very small improvements. Unfortunately, this full phrase table is too large to fit into memory. Therefore, we store the phrase table on disk and dynamically load only the parts into memory that are required to translate the current source sentence. To ensure fast loading, we use a binary file format that is a one-to-one mapping of the representation in memory. Experiments have shown that there is no penalty in terms of translation speed.

4 Architecture

In this section, we motivate the architecture by considering an example scenario. A detailed definition of the translation object and the interaction between the different server processes is described.

4.1 Example scenario

To give an overview of the architecture, we start with an example scenario: a translation request for some source language sentence is issued by the client to a dispatcher which creates a translation object containing the source sentence and passes it to a preprocessing engine. After this step (which might involve sentence segmentation, lowercasing[1], tokenization, categorization), the preprocessed sentence is passed to the translation engine. After having produced n-best translations, the modified object holding the hypotheses is passed into a module that applies, e.g., domain-specific reranking on them. Finally, the reranked translation

[1]The performance of SMT systems is sometimes better when trained on lowercase data, with a separate truecasing step during postprocessing.

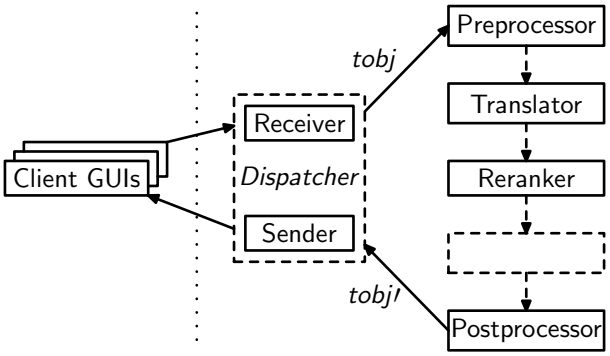

Figure 1: Architecture overview: translation objects are passed to the server modules and processed accordingly.

object is sent to a postprocessing module that forwards the final object back to the dispatcher and from there to the original client. In a multi-client setting (e.g. in an environment where many translators work at their desktops), this kind of parallelism helps to speed up the overall workflow.

4.2 Realization

Figure 1 depicts the setting from our example scenario. The path of modules to be taken for processing the translation object is not fixed but rather coded within the object according to the client's settings and needs. Thus, some of the servers can be skipped. It is also possible to set up more dispatcher processes that reflect several available configurations (such as specific language pairs or translation domains). The server module IP addresses are stored in a queue. Each module that finishes processing removes itself from the queue and sends the resulting translation object to the following server. The original client IP address is stored separately. The last process sends the final translation object back to the client.

The following data structures within a translation object are currently envisaged or already incorporated in the prototype:

- the source sentence to be translated
- the current prefix that is already translated (empty for initial request)
- a list of completion objects that hold
  - translation hypotheses
  - model scores
  - alignment information
  - word-level confidences
- other information, such as the number of requested n-best completions
- queue with server addresses (application chain used for processing the translation object)
- original client address

The representation of a translation object can be coded in various ways. Currently, we use low-level string representations, but an extendable XML-like structure is envisaged. If the format of the translation object's encoding is changed, only the input and output routines have to be adapted accordingly.

4.3 Enhancements

At present, the dispatcher prototype we are implementing is configured beforehand. Thus, the chain of applications the translation object moves through is predetermined. A possible extension could be an analysis component that analyzes the incoming data and decides which modules can be skipped, e.g. the preprocessing step.

4.4 Summary

The overall architecture could be classified as a hybrid peer-to-peer network. The communication between the clients and the dispatcher modules is situated in a client-server framework, i.e. the client sends a request and waits for the response from the server. The data flow after this process is organized more in a peer-to-peer fashion. Each server module picks the next peer the translation object has to be send to via the IP queue rather than communicating its results back to the dispatcher. Finally, the overall result is propagated to the dispatcher which then sends it to the originating client machine.

5 Experiments

In order to evaluate the prototype, we conduct experiments on different language-pairs

		XRCE		EU	
		English	Spanish	English	French
TRAIN	Sentence pairs	55 761		215 215	
	Tokens	571 972	657 309	6 001 777	6 647 861
	Vocabulary entries	25 627	29 565	83 743	91 304
	Singletons	9 765	11 966	37 669	39 659
TEST	Sentence pairs	1 125		800	
	Tokens	7 634	9 359	20 015	22 556
	Out-Of-Vocabulary words (OOV)	341	362	113	119
	Vocabulary entries	2 114	2 153	4 230	4 715

Table 1: Statistics of the XRCE (raw) and EU corpora used for the IMT experiments.

from the Xerox (XRCE) and EU corpora. The domain is translation of technical manuals for the former, whereas the latter deals with texts from the EU news bulletin. Exemplary corpus statistics for two language pairs, namely English-Spanish (for XRCE) and English-French (for EU) are shown in Table 1. We compare the results to previous experiments carried out with an SMT system based on alignment templates (Bender et al., 2005). Currently, not all modules are operational (cf. Figure 1). The pre- and postprocessing of the data is done client-side, and no additional reranking modules are active. The core translator is processing each translation object and able to provide n-best completions for a given prefix, as well as alignment information, word-level confidence and model scores.

Automatic evaluation measures of the two systems are given in Table 2. We consider the following measures:

- WER (word error rate): The WER is computed as the minimum number of substitution, insertion and deletion operations that have to be performed to convert the generated sentence into the reference sentence.
- PER (position-independent word error rate): A shortcoming of the WER is that it requires a perfect word order. The word order of an acceptable sentence can be different from that of the target sentence, so that the WER measure alone could be misleading. The PER compares the words in the two sentences ignoring the word order.
- BLEU and NIST scores: These scores are a weighted n-gram precision in combination with a penalty for sentences which are too short, and were defined in (Papineni, Roukos, Ward, & Zhu, 2002) and (Doddington, 2002), respectively. Both measure accuracy, i.e. higher scores are better.

In order to determine the effort a human translator would need to produce a reference translation, we use the following measure:

- KSMR (keystroke and mouse action ratio): This is the overall number of interactions of the user with the CAT system divided by the number of running characters for each sentence. As an interaction, we count keystrokes when typing in characters for parts where the system does not offer appropriate extensions as well as mouse actions (i.e. mouse clicks) that are needed to accept a specific part of the provided extension.

The KSMR is obtained by simulating a human translator that types each reference sentence by using the system's translations and extensions of an already fixed prefix of the reference sentence. The KSMR is a bit optimistic since it does not account for the actual time a user needs to read a proposed extension and then to select the longest matching part. However, for a comparison of systems and as an upper bound of their usability in a CAT setting, it is admissible.

	WER [%]	PER [%]	BLEU [%]	NIST	KSMR [%]	WER [%]	PER [%]	BLEU [%]	NIST	KSMR [%]
XRCE	English-Spanish					Spanish-English				
AT	33.4	28.3	62.0	9.5	23.2	40.2	34.4	57.2	8.7	24.0
PBT	32.8	27.5	64.7	9.7	19.7	37.4	31.8	61.0	8.9	19.6
EU	English-French					French-English				
AT	45.1	36.0	42.1	8.7	34.2	44.0	32.5	44.6	9.0	28.6
PBT	43.0	33.2	47.0	9.1	27.5	42.4	31.0	47.9	9.0	26.0

Table 2: Automatic evaluation measures for the interactive prototype of a phrase-based translation system (PBT) in comparison to a system based on alignment templates (AT) for the XRCE and EU corpora.

Interactive Session for XRCE Spanish-English ma/ks

PREFIX:	|	
EXTENSION:	The Font Management Utility is deleted from the system.	
PREFIX:	The Font Management Utility is r|	1/1
EXTENSION:	emoved from the system.	
PREFIX:	The Font Management Utility is removed from yo|	1/2
EXTENSION:	ur	
PREFIX:	The Font Management Utility is removed from your |	1/-
EXTENSION:	the system.	
PREFIX:	The Font Management Utility is removed from your s|	-/1
EXTENSION:	ystem.	
PREFIX:	The Font Management Utility is removed from your system.|	1/-
EXTENSION:		

SOURCE: La Utilidad de administración de fuentes queda eliminada del sistema.
REFERENCE: The Font Management Utility is removed from your system.

Figure 2: An interactive example session of the PBT prototype for a sentence of the Spanish-English XRCE test set. The numbers to the right denote mouse actions (ma) and keystrokes (ks). In total, the system results in a KSMR of $\frac{8}{56} = 0.143$.

5.1 Discussion

As can be seen from Table 2, the new PBT system clearly outperforms the system based on ATs. For all evaluation measures, we achieve significant improvements. Furthermore, the KSMR value is improved by up to 20% relative when compared to the AT system.[2] The average time to generate an extension for a given prefix is between 12 and 100 milliseconds on the server side, depending on the translation task (EU is somewhat slower than XRCE due to the larger corpus). For a fair evaluation, we would have to measure the overall runtime on the client side, since we have some overhead due to the client-server architecture. A manual experiment with our client prototype showed no noticeable delays even if the server ran on a distinct machine. Thus, at least for a setting in a local area network, the network communication overhead is negligible.

5.2 Examples

In Figures 2 and 3, we show interactive sessions demonstrating actual output of our system for sentences of the Spanish-English and English-French task. For a given source sentence, the user requests a translation. The result is the extension returned in the first row. Subsequently, we generate the ref-

[2]The KSR values of the AT system presented in (Bender et al., 2005) have been calculated differently. Thus, they are more optimistic than KSMR. Here, we show the corrected values.

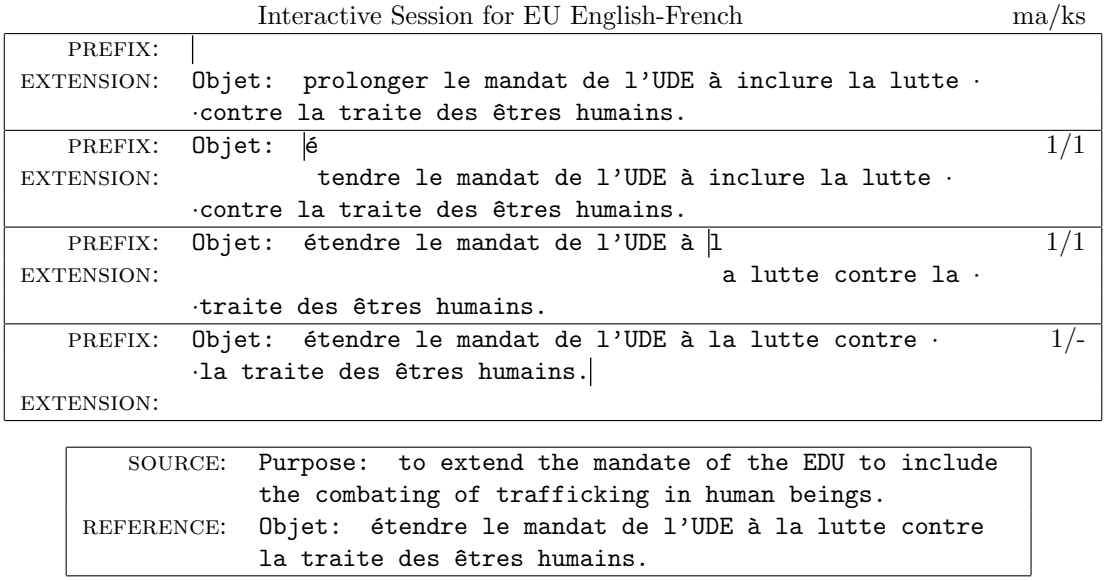

Figure 3: An interactive example session of the PBT prototype for a sentence of the English-French EU test set. The numbers to the right denote mouse actions (ma) and keystrokes (ks). In total, the system results in a KSMR of $\frac{5}{80} = 0.063$.

erence translation by selecting the longest match of the extension and, if necessary, produce the next character. A new extension is then provided for the source sentence given this new prefix. This process is iterated until the reference has been completely generated. The number of keystrokes and mouse actions is tracked and used for the calculation of the KSMR.

6 Conclusion

We introduced an extensible framework for CAT applications that allows for flexible setup of several components that interact on a client-server basis. The basic idea is to pass translation objects to the different applications, similar to message passing as known from computer science. The objects are processed through the chain of server modules and returned to the client with the final result, such as n-best completions of a partly translated source sentence.

We partially implemented the presented capabilities within a phrase-based SMT system and showed that the prototype outperforms another interactive system by up to 20% relative with comparable time constraints. The flexible architecture allows for easy extension with additional modules.

Future work will investigate more language pairs in detail and tune the system for performance. Additionally, we will incorporate more processing modules, such as an on-the-fly reranking server that uses additional models to rescore the n-best extensions produced by the phrase-based translation system. We will provide a detailed runtime analysis, though the current prototype is already real-time capable.

Acknowledgment

This work has been partly supported by the R&D project TRAMES managed by Bertin Technologies as prime contractor and operated by the French DGA (Délégation Générale pour l'Armement).

References

Bender, O., Hasan, S., Vilar, D., Zens, R., & Ney, H. (2005). Comparison of generation strategies for interactive machine translation. In *Proc. of the 10th Annual Conf. of the European Association for Machine Translation (EAMT)* (pp. 33–40). Budapest, Hungary.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Roossin, P. S.

(1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), 79–85.

Civera, J., Vilar, J. M., Cubel, E., Lagarda, A. L., Barrachina, S., Vidal, E., Casacuberta, F., Picó, D., & González, J. (2004). From machine translation to computer assisted translation using finite-state models. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP).* Barcelona, Spain.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. ARPA Workshop on Human Language Technology.*

Esteban, J., Lorenzo, J., Valderrábanos, A. S., & Lapalme, G. (2004). TransType2 — an innovative computer-assisted translation system. In *The Companion Volume to the Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics.* Barcelona, Spain.

Foster, G., Isabelle, P., & Plamondon, P. (1996). Word completion: A first step toward target-text mediated IMT. In *COLING '96: The 16th Int. Conf. on Computational Linguistics* (pp. 394–399). Copenhagen, Denmark.

Foster, G., Isabelle, P., & Plamondon, P. (1997). Target-text mediated interactive machine translation. *Machine Translation*, *12*(1), 175–194.

Isabelle, P., Dymetman, M., Foster, G., Jutras, J.-M., Macklovitch, E., Fran c. P., Ren, X., & Simard, M. (1993). Translation analysis and translation automation. In *CASCON '93: Proc. of the Conf. of the Centre for Advanced Studies on Collaborative Research* (pp. 1133–1147). IBM Press.

Kay, M. (1980). *The proper place of men and machines in language translation* (Tech. Rep.). Palo Alto, CA: Xerox Palo Alto Research Center. (Research Report CSL-80-11)

Langlais, P., Foster, G., & Lapalme, G. (2000). TransType: a computer-aided translation typing system. In *Workshop on Embedded Machine Translation Systems* (pp. 46–51). Seattle, WA.

Melby, A. K. (1983). Computer-assisted translation systems: The standard design and a multi-level design. In *Proc. of the First Conf. on Applied Natural Language Processing* (p. 174-177). Santa Monica, CA.

Ney, H., & Aubert, X. (1994). A word graph algorithm for large vocabulary continuous speech recognition. In *Proc. Int. Conf. on Spoken Language Processing* (pp. 1355–1358). Yokohama, Japan.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)* (p. 160-167). Sapporo, Japan.

Och, F. J., & Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics* (p. 295-302). Philadelphia, PA.

Och, F. J., Zens, R., & Ney, H. (2003). Efficient Search for Interactive Statistical Machine Translation. In *EACL03: 10th Conf. of the Europ. Chapter of the Association for Computational Linguistics* (p. 387-393). Budapest, Hungary.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)* (p. 311-318). Philadelphia, PA.

Ueffing, N., Och, F. J., & Ney, H. (2002). Generation of word graphs in statistical machine translation. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP)* (p. 156-163). Philadelphia, PA.

Zens, R., Bender, O., Hasan, S., Khadivi, S., Matusov, E., Xu, J., Zhang, Y., & Ney, H. (2005). The RWTH phrase-based statistical machine translation system. In *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT)* (p. 155-162). Pittsburgh, PA.