Computer Technology and Application 4 (2013) 575-583

Alignment of the Polish-English Parallel Text for a Statistical Machine Translation

Krzysztof Wołk and Krzysztof Marasek

Multimedia Department, Polish Japanese Institute of Technology, Warszawa 02-008, Poland

Received: November 02, 2013 / Accepted: November 13, 2013 / Published: November 25, 2013.

Abstract: Text alignment is crucial to the accuracy of MT (Machine Translation) systems, some NLP (Natural Language Processing) tools or any other text processing tasks requiring bilingual data. This research proposes a language independent sentence alignment approach based on Polish (not position-sensitive language) to English experiments. This alignment approach was developed on the TED (Translanguage English Database) talks corpus, but can be used for any text domain or language pair. The proposed approach implements various heuristics for sentence recognition. Some of them value synonyms and semantic text structure analysis as a part of additional information. Minimization of data loss was ensured. The solution is compared to other sentence alignment implementations. Also an improvement in MT system score with text processed with the described tool is shown.

Key words: Text alignment, NLP tools, machine learning, text corpora processing.

1. Introduction

SMT (statistical machine translation) requires a thorough understanding of the text to be translated [1]. SMT systems develop statistics from a large, parallel corpus of bilingual text as a basis for language translation. It means that the used corpus is crucial to the accuracy of SMT [2].

Before a parallel corpus can be used for training, the sentences must be aligned. Sentences in the raw corpus are misaligned, with translation lines whose placement does not correspond to the text lines in the source language. Moreover, some sentences may have no corresponding translation in the corpus. The corpus might also contain poor or indirect translations, making alignment difficult. Thus, alignment is crucial to SMT accuracy [2, 3]. Sentence alignment must also be computationally feasible in order to be of practical use in various applications [4]. As a result, sentence alignment poses a significant challenge.

Corresponding author: Krzysztof Wołk, M.Sc., research fields: NLP, machine translation. E-mail: krzysztof.wolk@pjwstk.edu.pl.

The Polish language is a particular challenge to SMT systems. It is a very complicated West-Slavic language with complex elements and grammatical rules. In addition, the Polish language has a large vocabulary due to many endings and prefixes changed by word declension. These characteristics have a significant effect on the SMT requirements for data and data structure. This challenge is increased substantially by the lack of resources for the SMT system data input. SMT systems should work best in specified, not too wide text domains, and do not perform well in general use. Good quality parallel data in a specific text domain for PL-EN pair is not available and is hard to obtain.

In addition, English is a position-sensitive language. The syntactic order (the order of words in a sentence) plays a very significant role, and the language has very limited inflection of words (e.g., due to the lack of declension endings). The word position in an English sentence is often the only indicator of the meaning. The sentence order follows the SVO (Subject-Verb-Object) schema, with the subject phrase preceding the predicate.

On the other hand, no specific word order is imposed

in Polish, and the word order has little effect on the meaning of a sentence. The same thought can be expressed in several ways. For example, the sentence "I bought myself a new car." can be written in Polish as one of the following:

"Kupiłemsobienowysamochód."

"Nowysamochódsobiekupiłem."

"Sobiekupiłemnowysamochód."

"Samochódnowysobiekupiłem."

Potential word order differences complicate the translation process, especially when a phrase-model is used with no additional lexical information.

As a result of these complexities, the progress in SMT systems for the Polish language has been slow in comparison to other languages.

This paper proposes a language independent sentence alignment method that has been applied to Polish-English parallel corpora. First, the method is described. Next, an alignment metric is discussed. A quality comparison of this method to other alignment methods is then made using data from experiments. Lastly, conclusions are drawn.

The dataset used for this research was the TED [5], provided by FBK (Fondazione Bruno Kessler) [6]. Vocabulary sizes of the English and Polish texts in TED are disproportionate. There are 41,684 unique English words and 88,158 unique Polish words. This also presents a challenge for SMT systems. This paper is structured as follows: Section 2 explains the Polish data preparation; Section 3 presents the English language issues; Section 4 describes the translation evaluation methods; Section 5 discusses the results; Sections 6 summarizes potential implications and future work.

2. Literature Overview

Early attempts at automatically aligning sentences for parallel corpora were based on sentence lengths, together with vocabulary alignment [7]. Brown's method [8] was based on measuring sentence length by the number of words. Gale and Church [9] measured the number of characters in sentences. Other researchers continued exploring various methods of combining sentence length statistics with alignment of vocabularies [10, 11].

Several text aligners implementing these methods are currently available, including Bleualign, which is an open source project developed by the University of Zurich. In addition to parallel texts, Bleualign requires a translation of one of the texts. It uses the length-based BLEU (Bilingual Evaluation Understudy) similarity metric to align the texts [12].

The Hunalign tool is another open source tool, developed by the Media Research Center. Based on Gale and Church's method, it uses sentence lengths and, optionally, a dictionary to align texts in two languages strictly on a sentence level. It does not address the sentence ordering issues [13].

ABBYY Aligner is a commercial product developed by the ABBYY Group. This product reportedly uses proprietary word databases to align text portions of sentences based on their meaning [14].

Unisex Aligner [15] is an open source project primarily developed by the University of Paris-Est Marne-la-Vallée (France). It uses the XAlign tool [16], which uses character lengths at the paragraph and sentence level for text alignment [17].

3. Proposed Sentence Aligner

A sentence aligner was designed to find an English translation of each Polish line in a corpus and place it in the correct place in the English file. This aligner was implemented as a Python script.

Our proposed approach encompasses two concepts: one that is not free and one that is free. The first concept is to use the Google Translator API (Application Programming Interface) for lines for which an English translation does not exist and also for comparison between the original and translated texts. The second concept is based on web crawling, using Google Translator, Bing Translator, and Babylon translator. These can work in a parallel manner to

improve performance. In addition, each translator can work in many instances, so that one translator can be used in parallel. Our approach can also accommodate a user-provided translation file in lieu of crowd sourcing.

First, there is a need for our script to make an English translation file in which each line corresponds to lines in the Polish file. This means that we need the translation of the first line of the Polish file in the first line of the English file, the second line of the Polish file in the second line of the English file, etc.. Next, we compare the newly translated English file with the original English file, both line by line and also by scanning neighbor lines. If we find English translations that are similar by some defined factor, we use the original line. Otherwise, we use the Google or other translation.

Our strategy is to find a correct translation of each Polish line aided by Google Translator or another translation engine. We translate all lines of the Polish file with Google Translator and put each line translation in an intermediate English translation file. This intermediate translation helps us find the correct line in the English translation file and put it in the correct position.

Let us call the source Polish file as "src.pl", the source English translation file as "src.en", and the Google translation file as "src.trans". The Polish and English files generated from aligner are "hyp.pl" and "hyp.en", respectively.

We use the similarity of each src.trans line with lines in src.en to determine which line in src.en is a translation of which line in src.pl. The algorithm accomplishes this in the following steps:

- Choose the first lines of src.trans and src.pl;
- Compare the src.trans line with, for example, the three first lines of src.en;
- Select the most similar line from src.en;
- If the selected line similarity rate is more than a specific acceptance rate, choose it as the translation of the corresponding src.pl line;
- If the selected line similarity rate in not sufficiently high, ignore it and select the src.trans line as the translation.

In reality, the actual solution is more complex. Suppose that we choose one of the src.en lines as the most similar line to the specific src.trans line and that its similarity rate is high enough to be accepted as the translation. This line can be more similar to the next line of src.trans, so that the similarity rate of this selected line and the next line of src.trans is higher. For example, consider the sentences and their similarity rating in Table 1.

In this situation, we should select "I do not go to school every day." from src.en instead of "I do not go to school every day" from src.trans, and not "I go to school every day". So, we should consider the similarity of a selected line with the next lines of src.trans to make the best possible selection in the alignment process.

There are additional complexities that must be addressed. Comparing the src.trans lines with the src.en lines is not easy, and it becomes harder when we want to use the similarity rate to choose the correct, real-world translation.

There are many strategies to compare two sentences. We can split each sentence into its words and find the number of words in both sentences. However, this approach has some problems. For example, let us compare "It is origami" to these sentences:

"The common theme what makes it origami is folding is how we create the form";

"This is origami".

With this strategy, the first sentence is more similar because it contains all 3 words. However, it is clear that the second sentence is the correct choice. We can solve this problem by dividing the number of words in both sentences by the number of total words in the sentences. However, counting stop words in the intersection of sentences sometimes cause incorrect results. So, we remove these words before comparing two sentences.

Another problem is that sometimes we find stemmed words in sentences, for example "boy" and "boys".

Table 1 Example similarity ratings.

src.trans	src.en	Sim.
I go to school every day.	I like going to school every day.	0.60
I go to school every day.	I do not go to school every day.	0.70
I go to school every day.	We will go tomorrow.	0.30
I do not go to school every day.	I like going to school every day.	0.55
I do not go to school every day.	I do not go to school every day.	0.95
I do not go to school every day.	We will go tomorrow.	0.30

Despite the fact that these two words should be counted as similarity of two sentences, with this strategy, these words are not counted.

The next comparison problem is the word order in sentences. In Python there are other ways for comparing strings that are better than counting intersection lengths. The Python, "difflib" library for string comparison contains a function that first finds matching blocks of two strings. For example, we can use difflib to find matching blocks in the strings "abxcd" and "abcd".

Difflib's "ratio" function divides the length of matching blocks by the length of two strings, and returns a measure of the sequences' similarity as a float value in the range [0, 1]. This measure is $2.0 \times M/T$, where T is the total number of elements in both sequences, and M is the number of matches. Note that this measure is 1.0 if the sequences are identical, and 0.0 if they have nothing in common. Using this function to compare strings instead of counting similar words helps us to solve the problem of the similarity of "boy" and "boys". It also solves the problem of considering the position of words in sentences.

Another problem in comparing lines is synonyms. For example, in these sentences:

"I will call you tomorrow."

"I would call you tomorrow."

If we want to know if these sentences are the same, we should know that "will" and "would" can be used interchangeably.

We used the NLTK Python module and Word Net® to find English synonyms for each word and to use these synonyms in comparing sentences. Using synonyms of each word, we created multiple sentences from each original sentence.

For example, suppose that the word "game" has the synonyms: "play", "sport", "fun", "gaming", "action", and "skittle". If we use, for example, the sentence "I do not like game.", we create the following sentences:

"I do not like play."

"I do not like sport."

"I do not like fun."

"I do not like gaming."

"I do not like action."

"I do not like skittle."

Next, we try to find the best score by comparing all these sentences instead of just comparing the main sentence. One issue is that this type of comparison takes too much time, because we need to do many comparisons for each selection.

Difflib has other functions (in Sequence Matcher and Diff class) to compare strings that are faster than the described solution, but their accuracy is worse. To overcome all these problems and obtain the best results, we consider two criteria: the speed of the comparison function and the comparison acceptance rate.

To obtain the best results, our script provides users with the ability to have multiple functions with multiple acceptance rates. Fast functions with lower quality results are tested first. If they can find results from a very high acceptance rate, we accept their selection. If the acceptance rate is not sufficient, we can use slower but higher accuracy functions. The user can configure these rates manually and test the resulting quality to get the best results.

Because we used the Google Translator API and comparison functions that are not specific to any language, the program should be able to align similarly

structured languages that are supported by Google Translator with English. Alignment between a language pair not including English would require use of a different lexical library for synonyms or not using some comparison functions.

Since the Google API is money expensive, we also implemented a free solution based on web crawling of three services, e.g., Google Translator, Bing Translator, and Babylon. It is possible to use one of them or all of them simultaneously for better performance. However, the latter solution is much slower than using Google API alone and can lead to worse results, esp. when using many different translation engines at the same time.

This solution is only intended for educational purposes. Information about each data domain would require adapting the parameters in order to provide the best alignment. In general, a text is associated with a domain, i.e. a particular subject area and mode of writing, e.g., a political science essay [18]. As discussed in Ref. [19], texts from different domains are likely to use words with different meanings. If a text domain is ignored, this can lead to translations that are misleading.

Reference and training corpora come from various text domains. For example, the TED lectures address multiple domains, and OPUS (Open Parallel Corpus) contains movie subtitles with very short sentences [5]. Research has previously shown that an advantage of SMT systems is that they can be applied to a different text domain by training them on different data from the new domain [18-20].

The crawler translator was developed to feed the native sentences to online translation services such as Google, Bing, and Babylon and retrieve the translated text. These online services have their own web sites for serving human users. The problem with that was how to simulate the behavior of a human user when entering source texts. On the other hand, since there is a potential to be blocked by the online services if the program overloads the online services, there should be a sufficient delay in feeding the sentences.

The crawler translator was developed using Python, the PyQt graphical user interface, and the PyQtwebkit for implementing a simple web browser. Webkits are frameworks for handling the web pages within the programming language. A simple GUI, based on the data provided in a configuration file, loads the web pages of online translation services. These pages are loaded in separate windows and work in parallel. This will increase the speed of the translation.

Note that if the program is correctly configured and all parameters are well-tuned, and if it then uses a translation from Google, it most likely means that a sentence pair in the original file was not translated properly or at all.

The proposed method automatically creates corpora that can be used by SMT systems. Some other aligners work in only a semi-automatic or fully manual manner. If they are unable to align, they leave an empty line. Clearly, this results in problems for SMT, where each line must be translated in order to get the best possible results.

4. Sentence Alignment Metric

We developed a special metric to evaluate aligner quality and tuned its parameters during this research. For an aligned sentence, we give 1 point. For a misaligned sentence, we give a -0.2 points penalty. For web service translations, we give 0.4 points. For translations due to disproportion between input files, we give 1 point. The score is normalized to fit between 1 and 100. A higher value is better. A score of 100 means perfect alignment. A floor function can be used to round the score to an integer value. The score S is defined as:

$$S = floor(\frac{20(5A - M + 2T + 5|\mathrm{D}|}{L}) \quad (1)$$

Where, A is the number of aligned sentences, M is the number of misaligned sentences, T is the number of translated sentences, D is the number of lines not found in both language files (one file can contain some

sentences that do not exist in the other one), and L is the total number of output lines.

Some additional scoring algorithms were also implemented. These are more suited for comparing language translation quality. We added the BLEU and native implementations of TER (Translation Edit Rate) and CER (Character Edit Rate) by using pycdec and the RIBES (Rank-based Intuitive Bilingual Evaluation Measure). BLEU and TER are well-described in the literature [21, 22]. RIBES is an automatic evaluation metric for machine translation, developed in NTT Communication Science Labs [23].

The pycdec module is a Python interface to the cdec decoding and alignment algorithms [24, 25]. The BLEU metric compares phrases from a source text with reference translations of the text, using weighted averages of the resulting matches. It has been shown that BLEU performs well in comparison to reference human translations. BLEU is defined in Ref. [26] in terms of the n-gram precision (using n-grams up to length N) and weights (positive only) whose sum is one:

$$BLEU = P_B \exp(\sum_{n=0}^{N} w_n \log p_n) \quad (2)$$

Here, P_B is the brevity penalty, which is given by Ref. [6] as:

$$P_B = \begin{cases} 1, c > r \\ e^{(1-\frac{r}{c})}, c \le r \end{cases} \quad (3)$$

In the equation above, c is the candidate phrase translation length, and r is the length of the reference translation phrase, e is Euler's constant [26].

BLEU is designed to avoid word order biases that might artificially improve the score of a SMT system. First, the position of words or phrases is not reflected in the standard BLEU metric. Second, reference translation's word count limits the word count of a candidate translation on a sentence by sentence basis. This avoids bias that would enable SMT systems to overuse high confidence words in order to boost their score. In addition, BLEU applies the brevity penalty and uses the geometric mean of the individual sentence scores [26].

The TER metric is intended to capture the quality of essential meaning and fluency of SMT system translations. It measures human translator edits required for a machine translation to match a reference translation. TER accounts for word substitutions, word insertions and deletions, and phrase and word order modifications [23].

Let E be the number of required edits and w_R be the average reference translation lengths.

Then, the metric is given by Ref. [23] as:

$$TER = \frac{E}{W_R} \quad (4)$$

Prior research in machine translation evaluation has resulted in many known solutions that we can be used to make our aligner scoring system work better. Evaluating the quality of a computer-generated translation relative to a human-generated reference translation or a set of human-generated reference translations is useful for automatic parameter tuning, to compare the quality of two machine translation systems, and to carry out minimum Bayes risk decoding [27, 28].

A Minimum Bayes-Risk decoder, as described in Ref. [27], can be used to select a translation by consensus. Such a decoder is given by Ref. [27] as:

$$\delta(F) = \text{argmin} \sum_{E,A} L((E,A),(E',A');F)P(E,A \setminus F) \quad (5)$$

Where, L is a translation loss function, E is a reference translation with word alignment A for a sentence F, and E' is a machine translation with word alignment A for the sentence.

We use these algorithms to generate likelihood scores for two sentences, to choose the best one in the alignment process. For this purpose, we used cdec and pycdec. "cdec" is a decoder, aligner, and learning framework for statistical machine translation and similar structured prediction models. It provides translation and alignment modeling based on finite-state transducers and synchronous context-free grammars, as well as implementations of several parameter learning algorithms [7, 16].

The pycdec is a Python module for the cdec decoder. It enables Python coders to use cdec's fast C++ implementation of core finite-state and context-free inference algorithms for decoding and alignment. The high-level interface allows developers to build integrated MT applications that take advantage of the rich Python ecosystem without sacrificing computational performance. The modular architecture of pycdec separates search space construction, rescoring, and inference.

"cdec" includes implementations of the basic evaluation metrics (BLEU, TER and CER), exposed in Python via the cdec.score module. For a given (reference, hypothesis) pair, sufficient statistics vectors (Sufficient Stats) can be computed. These vectors are then summed for all sentences in the corpus, and the final result is finally converted into a real-valued score.

Before aligning a big data file, it is important to determine the proper comparators and accept cerates for each one. Files of 1,000-10,000 lines result in the best performance and are commended to firstly evaluate each comparison method separately, and then combining the best ones in a specific scenario. For this purpose, we recommend using the binary search method in order to determine the best thresh old factor value.

The binary search method repeatedly divides an array to be searched in half to locate a value [29]. Given a "high" and "low" value in an array, this method use integer division to find the midpoint "mid", as shown in Ref. [29]:

$$mid = (\frac{low + high}{2}) \quad (6)$$

The time complexity of this algorithm is $O(\log_2 n)$, where n is the length of the array to be searched.

We created an evaluation script in Python using the open source Python NLTK library, described in Ref. [25, 30]. The script compares the output of alignment software with a human manually-aligned text and determines the score S described above. It takes a pair from aligner output files and searches for it in human aligned text. If this is not found, the script checks if the translation was taken from Google. If these checks fail, it means there is a mistake.

5. Comparison Experiments

Experiments were performed to compare the performance of the proposed method with several other sentence alignment implementations on the data found in Ref. [31], using the metric defined earlier. The Polish data in the TED lectures (approximately 15 MB) includes over 2 million words that are not tokenized. The transcripts themselves are provided as pure text encoded with UTF-8. In addition, they are separated into sentences (one per line) and aligned in language pairs.

The additional aligners, all created to develop parallel corpora, used in this experiment were: Bleualign, hunalign, ABBYY Aligner, Wordfast Aligner, and Unitex Aligner. The performance of the aligners was scored using the sentence alignment metric described in Section 3. Table 2 provides the results.

Clearly, the first three aligners scored well. The proposed method is fully automatic. It is important to note that Bleualign does not translate text and requires that it be done manually.

As discussed earlier, it is important not to lose lines of text in the alignment process. Table 3 shows the total lines resulting from the application of each alignment method.

All the aligners compared, other than the proposed method, lose lines of text as compared to a reference human translation. The proposed method lost no lines.

In purpose of showing the output quality with an independent metric we decided to compare results with BLEU, NIST, METEOR and TER (the lower the better), in a comparison with human B1aligned texts. Those results are presented in Table 4.

6. Conclusions

In general, sentence alignment algorithms are very important in machine translation and for creating

Table 2 Experimental results.

Aligner	Score
Proposed method	98.94
Bleualign	96.89
Hunalign	97.85
ABBYY aligner	84.00
Wordfast aligner	81.25
Unitex aligner	80.65

Table 3 Experimental results.

Aligner	Lines
Human translation	1,005
Proposed method	1,005
Bleualign	974
Hunalign	982
ABBYY aligner	866
Wordfast aligner	843
Unitex aligner	838

Table 4 BLEU Comparison results.

Aligner	BLEU	NIST	MET	TER	% correct
Human translation	100	15	100	0	100
Proposed method	98.91	13.81	99.11	1.38	98
Bleualign	91.62	13.84	95.27	9.19	92
Hunalign	93.10	14.10	96.93	6.68	94
ABBYY aligner	79.48	12.13	90.14	20.01	83
Wordfast aligner	85.64	12.81	93.31	14.33	88
Unitex aligner	82.20	12.20	92.72	16.39	86

parallel text corpora. Most aligners are not fully automatic, but the one proposed here is, which gives it a distinct advantage. It also allows creating a corpus when sentences exist just in a single language. The proposed approach is also language independent for ones with similar structure to PL or EN.

The results show that the proposed method performed very well in terms of the metric. It also lost no lines of text, unlike the other aligners. This is critical to the end goal of obtaining a translated text. Our alignment method also proved to providebetterscorewhencomparingwithtypicalmachine translationmetrics, and would most likely improve MT systems output quality.

Acknowledgments

This work is supported by the European Community from the European Social Fund within the Interkadra project UDA-POKL-04.01.01-00-014/10-00 and Eu-Bridge 7th FR EU project (grant agreement n°287658).

References

[1] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, 2010, pp. 907-910.

[2] Y. Deng, S. Kumar, W. Byrne, Segmentation and alignment of parallel text for statistical machine translation, Natural Language Engineering 12 (4) (2006) 1-26.

[3] S. Karimi, F. Scholer, A. Turpin, Machine transliteration survey, ACM Computing Surveys 43 (3) (2011) 6-46.

[4] F. Braune, A. Fraser, Improved Unsupervised Sentence Alignment for Symmetrical and Asymmetrical Parallel Corpora, in: Coling 2010: Poster Volume, 2010, pp. 81-89.

[5] K. Marasek, TED Polish-to-English translation system for the IWSLT 2012, in: Proc. of International Workshop on Spoken Language Translation (IWSLT) 2010, Hong Kong, 2012.

[6] M. Cettolo, C. Girardi, M. Federico, Wit 3: Web inventory of transcribed and translated talks, in: Proc. of 16th Conference of the European Association for Machine Translation (EAMT), Trento, Italy, 2012. pp. 261-268.

[7] A. Santos, A survey on parallel corpora alignment, in: MI-STAR, 2011, pp. 117-128.

[8] P.F. Brown, J.C. Lai, R.L. Mercer, Aligning sentences in parallel corpora, in: Proc. of 29th Annual Meeting of the ACL, Berkeley, 1991, pp. 169-176.

[9] W.A. Gale, K.W. Church, Identifying word correspondences in parallel texts, in: Proc. of DARPA Workshop on Speech and Natual Language, 1991, pp.152-157,

[10] D. Varga, P. Halacsy, A. Kornai, V. Nagy, L. Nemeth, et al., Parallel corpora for medium density languages, in: Proc. of the RANLP 2005, Borovets, Bulgaria, 2005, pp. 590-596.

[11] F. Braune, A. Fraser, Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora, in: Proc. of 23rd COLING International Conference, Beijing, China, 2010, pp. 81-89.

[12] Bleualign, https://github.com/rsennrich/Bleualign, (accessed Aug. 8, 2013).

[13] ABBYY Aligner, http://www.abbyy.com/aligner/ (accessed Aug. 7, 2013).

[14] Unitex/Gramlab, http://www-igm.univ-mlv.fr/~unitex/#, (accessed Aug. 7, 2013).

[15] S. Paumier, T. Nakamura, S. Voyatzi, UNITEX, a Corpus Processing System with Multi-Lingual Linguistic Resources, in: eLEX2009, p. 173.

[16] P. Bonhomme, L. Romary, The lingua parallel concordancing project: Managing multilingual texts for educational purpose, in: Proc. of Quinzièmes Journées Internationales IA 95, Montpellier, 1995.

[17] A. Schmidt, Statistical machine translation between new language pairs using multiple intermediaries, Ph.D. Thesis, Ruprecht-Karls-Universitat, Heidelberg, 2007.

[18] L. Specia, D. Raj, M. Turchi, Machine translation evaluate versus quality estimation, Machine Translation 24 (1) (2010) 39-50.

[19] S. Mirkin, L. Specia, N. Cancedda, I. Dagan, M .Dymetman, I. Szpektor, Source-Language entailment modeling for translating unknown terms, in: Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 791-799.

[20] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul, A study of translation edit rate with targeted human annotation, in: Proc. of 7th Conference of the Assoc. for Machine Translation in the Americas, Cambridge, 2006.

[21] V.I. Levenshtein, Binary codes with correction for deletions and insertions of the symbol 1, Problemy Peredachi Informacii 1 (1) (1965) 12-25.

[22] Linguistic Intelligence Research Group, NTT Communication Science Laboratories, RIBES: Rank-based Intuitive Bilingual Evaluation Score [Online], http://www.kecl.ntt.co.jp/icl/lirg/ribes/ (accessed Aug. 7, 2013).

[23] V. Chahuneau, N.A. Smith, C. Dyer, pycdec: A Python Interface to cdec, The Prague Bulletin of Mathematical Linguistics 98 (2012) 51-61.

[24] C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, H. Setiawan , F. Ture, et al, cdec: A decoder, Alignment, and learning framework for finite-state and context-free translation models, in: Proc. of ACL 2010 System Demonstrations, Uppsala, Sweden, pp. 7-12.

[25] K. Papineni, S. Rouskos, T. Ward, W.J. Zhu, BLEU: A method for automatic evaluation of machine translation, in: Proc. of 40th Annual Meeting of the Assoc. for Computational Linguistics, Philadelphia, 2002, pp. 311-318.

[26] S. Kumar, W. Byrne, Minimum bayes-risk decoding for statistical machine translation, in: HLT-NAACL 2004, Boston, USA, 2004.

[27] N. Ehling, R. Zens, H. Ney, Minimum bayes risk decoding for bleu, in: Proc. Assoc. for Computational Linguistics, Prague, 2007.

[28] Parkland College, The Binary Search Algorithm, http://www.csit.parkland.edu/~mbrandyberry/CS1Java/Lessons/Lesson27/BinarySearch.htm, (accessed Aug. 7, 2013.).

[29] NLTK Project Team, NLTK 2.0 Documentation, http://nltk.org/api/nltk.html, downloaded on September 8, 2013.

[30] hunalign—sentence aligner, http://mokk.bme.hu/resources/hunalign/, (accessed Aug. 8, 2013).

[31] International Workshop on Spoken Language Translation (IWSLT), http://www.iwslt2013.org/ (accessed Aug. 7, 2013).