PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 98 OCTOBER 2012 63–74

Phrasal Rank-Encoding: Exploiting Phrase Redundancy and Translational Relations for Phrase Table Compression

Marcin Junczys-Dowmunt

Information Systems Laboratory, Adam Mickiewicz University, Poznań, Poland
Global Databases Service, World Intellectual Property Organization (WIPO), Geneva, Switzerland

Abstract

We describe Phrasal Rank-Encoding (PR-Enc), a novel method for the compression of word-aligned target language data in phrase tables as used in phrase-based SMT. This method reduces the redundancy in phrase tables which is a direct effect of the phrase-based approach. A combination of PR-Enc with Huffman coding allows to reduce the size of an aggressively compressed phrase table by another 39 percent. Using this and other methods for space reduction in a new binary phrase table implementation, a size reduction by an order of magnitude is achieved when comparing to the Moses on-disk phrase table implementation. Concerning decoding speed, all variants of the new phrase table are faster than the Moses binary phrase table implementation while the PR-Enc encoded variant outperforms all other methods.

1. Introduction

Phrase tables as used in phrase-based statistical machine translation (PB-SMT) are huge. Their size is a direct consequence of the PB-SMT approach itself and the fact that precomputed phrase pairs can be accessed efficiently. Precomputation, however, leads to a combinatorial bloat of phrases and to phrase redundancy since for any phrase pair all possible subphrase pairs may be included in the phrase table.

Explicitly stored phrase tables are currently the most widely used representation of translation models in PB-SMT. The main goal of this paper is to describe Phrasal Rank-Encoding (PR-Enc) — a lossless encoding and compression method dedicated to explicitly stored phrase tables. PR-Enc aims to reduce redundancy by exploiting the phrase table itself as a compression dictionary, making use of translational relations

Corresponding author: junczys@amu.edu.pl
Cite as: Marcin Junczys-Dowmunt. Phrasal Rank-Encoding: Exploiting Phrase Redundancy and Translational Relations for Phrase Table Compression. The Prague Bulletin of Mathematical Linguistics No. 98, 2012, pp. 63–74. doi: 10.2478/v10108-012-0009-6.

between subphrases and the repetitiveness of subphrases. Previous approaches concentrate only on one of these properties and correspondingly either require additional resources, like external translation dictionaries, or is restricted to a certain type of phrasal redundancy, for instance prefixes in trie-based implementations. An efficient dynamic programming algorithm for the decompression of PR-Enc encoded target phrases is presented. The described method is employed in a previously introduced compact phrase table implementation (Junczys-Dowmunt, 2012a,b) for Moses (Koehn et al., 2007) that can be used as a replacement for the current binary phrase table.

2. Related Work

2.1. Previous Work in Machine Translation

Zens and Ney (2007) describe a phrase table architecture on which the standard binary phrase table of Moses is based. Memory requirements are low due to on-demand loading, disk space requirements, however, can become substantial. The only compression technique relies on the application of a trie for the representation of a source phrase index which collapses common source phrase prefixes into single paths.

Germann et al. (2009) introduce tightly packed tries (TBT) which they use for language models and phrase tables in the Portage SMT system (Sadat et al., 2005). TBTs are stored in byte arrays with variable byte encoding applied to reduce their space requirements. The section on phrase tables does not provide sufficient information to compare our approaches, but we have contacted the authors and may be able to provide a comparison of compression rates in the future.

Suffix-array based implementations of translation models have been introduced by Callison-Burch et al. (2005). Both sides of a parallel corpus are stored as suffix arrays. If alignment data between the parallel sentences is provided, phrase pairs can be extracted and scored on demand. The precomputation of phrase pairs is avoided altogether, which immediately solves the problem of redundancy. On-demand translation models are a promising alternative to phrase tables, but we do not compare this approach with ours, instead we concentrate on explicitly stored phrase tables.

Phrase table filtering (e.g. Johnson et al., 2007) can be seen as a type of lossy compression. Reduction rates of more than 80 percent while maintaining or even improving translation quality are not uncommon. We find this particularly interesting since a combination of phrase table filtering with our approach can yield a translation model size reduction by two orders of magnitude.

2.2. Compression of Parallel Corpora

Conley and Klein (2008) propose an encoding scheme of target language data based on word alignment and translational relations. However, they require the existence of lemmatizers and a translation lexicon. From the aligned parallel data a lexi-

con of lemmatized parallel phrases is created. Target phrases are replaced with pointers consisting of start and end indexes of the corresponding source phrase, indexes of translations, and one integer pointer per target word to its inflected form. This turns the method into a word-based method since no length reduction of the text is achieved. Actually, there are more pointers after encoding than there were target words before. Compression is achieved by the application of a Huffman coder.

The most recent work is Sanchez-Martinez et al. (2012) who propose to use "biwords" to compress parallel data sequentially. Similar as in Conley and Klein (2008), translational relations and Huffman coding are employed to take advantage of the improved entropy properties of the encoded data. This method is called Translational Relation Encoding (TRE). Again, mainly word-based translational relations are used, allowing at most many-to-one alignments. Sanchez-Martinez et al. include a list of biwords, a translation dictionary extracted from the alignment, in the compressed file.

2.3. Compact Phrase Table Implementation

Junczys-Dowmunt (2012a) introduces a compact phrase table architecture which we use for our experiments with PR-Enc. A "baseline" variant is presented that uses standard compression methods like the Simple-9 algorithm (Anh and Moffat, 2004), variable-byte encoding (Scholer et al., 2002), and Huffman coding (Huffman, 1952) of target words, scores, and alignment points. Size reduction for source phrases is achieved by using a minimal perfect hash function as an index. Junczys-Dowmunt (2012b) describes the further reduction of the source phrase index and the impact of false positive assignments of target phrases to source phrases on translation quality. That implementation achieves a size reduction of more than 77 percent when compared to the Moses binary phrase table with significantly better performance.

Also in Junczys-Dowmunt (2012a) word-based Rank-Encoding is described. This method is similar to TRE (Sanchez-Martinez et al., 2012), but does not store source words which are provided during phrase table querying. Target phrase words are replaced with pairs of pointers. The first pointer indicates the corresponding source phrase word, the second the rank of the target word among the translations of the source word. A translation lexicon generated from the alignment is included in the phrase table. Again, Huffman coding improves the compression rate.

3. Phrasal Rank Encoding

The general idea of Phrasal Rank-Encoding is similar to that of classic dictionary-based compression methods like LZ77 (Ziv and Lempel, 1977). Repetitive subphrases are replaced by pointers to subphrases in a phrase dictionary which should result in a reduction of data length. Decompression relies on the look-up and reinsertion of subphrases based on the pointer symbols. Something similar, though in an rather ineffective way, has been attempted by Conley and Klein (2008). If we simplify their

method by dropping all external data requirements and move it onto the ground of phrase tables, we get a basic version of Phrasal Rank-Encoding. Instead of compressing a bitext with a translation lexicon of phrases, we compress the lexicon itself. In Phrasal Rank-Encoding the compressed phrase table is its own phrase dictionary.

Although Phrasal Rank-Encoding shares some properties with word-based Rank-Encoding and mentioned bitext compression methods, compression is achieved in a different way: sequential data is implicitly turned into a graph-like structure similar to a trie or finite-state automaton, which is more visible during the discussion of the decoding algorithm. Translational relations and entropy coding help to compress the graph structure itself and not so much the data in it, which is not unlikely to the work presented by Germann et al. (2009). Phrasal Rank-Encoding could also be used to turn the target data of the Moses binary phrase table based on Zens and Ney (2007) into a graph-like structure without changing the underlying implementation (much).

3.1. Encoding Procedure

The encoding procedure is presented in Figure 1. PR-Enc requires the phrase table to contain word alignment information. In order to perform encoding efficiently, it should be possible to look-up phrase pairs and to retrieve the rank of a target phrase relative to its corresponding source phrase. By rank we mean the position of a target phrase among a list of phrase pairs with the same source phrase. The list is ordered decreasingly by the translation probability $P(\mathbf{t}|\mathbf{s})$, i.e. the most probable translation has rank 0. In our implementation this is achieved by creating a minimal perfect hash function with concatenations of source and target phrases as keys which are mapped to ranks. This searchable phrase table is passed to the algorithm as RankedPT.

We illustrate the algorithm with an example. Given are a Spanish-English phrase pair and the internal word alignment depicted by the black boxes in Figure 2:

```
es: Maria no daba una bofetada a la bruja verde
en: Mary did not slap the green witch
```

Phrase pairs are represented as quadruples where the values correspond to the source phrase start position, the target phrase start position, the length of the source phrase, and the length of the target phrase. In line 3 of the algorithm, all true subphrase pairs of the encoded phrase pair are computed. The result is marked in Figure 2 by filled and empty rectangles — with one exception: the complete phrase pair itself is ruled out for not being a true subphrase pair. The first condition of the expression in line 3 requires subphrase pairs to lie within the boundaries of the encoded phrase. The second, introduced by Zens et al. (2002), defines subphrase pairs that are consistent with the underlying alignment. The same procedure is used during phrase pair extraction when the translation model is created. In order to avoid self-references, the third condition forbids to add the input phrase pair itself.

Next, the subphrase pairs are inserted into a queue (line 4) according to the following order: subphrase pairs are ordered decreasingly by length and increasingly by

```
1  Function EncodeTargetPhrase(s, t, A, Order, RankedPT)
2      t̂ ← ⟨ ⟩; Â ← A
       P ← {⟨i, j, m, n⟩ : (0 ≤ i < i + m ≤ |s| ∧ 0 ≤ j < j + n ≤ |t|)
3                          ∧ ∀⟨i′, j′⟩ ∈ A : (i ≤ i′ < i + m ⇔ j ≤ j′ < j + n)
                           ∧ (m < |s| ∨ n < |t|)}
4      Q ← Queue(P, Order)
5      while |Q| > 0 do
6          ⟨i, j, m, n⟩ ← Pop(Q)
7          s′ ← Substring(s, i, m)
8          t′ ← Substring(t, j, n)
9          if ∃r : ⟨s′, t′, r⟩ ∈ RankedPT then
10             T[j] ← ⟨i − j, |s| − (i + m), r⟩
11             S[j] ← n
12             Â ← Â \ {⟨i′, j′⟩ ∈ Â : i ≤ i′ < i + m ∧ j ≤ j′ < j + n}
13             P ← P \ {⟨i′, j′, m′, n′⟩ : i ≤ i′ < i + m ∨ j ≤ j′ < j + n
                                           ∨ i′ ≤ i < i′ + m′ ∨ j′ ≤ j < j′ + n′}
14             Q ← Queue(P, Order)
15     j ← 0
16     while j < |t| do
17         if S[j] > 0 then
18             t̂ ← t̂ · ⟨T[j]⟩
19             j ← j + S[j]
20         else
21             t̂ ← t̂ · ⟨t_j⟩
22             j ← j + 1
23     return ⟨t̂, Â⟩
```

Figure 1. Algorithm for Phrasal Rank-Encoding

the start position of the target phrase, then by length and start position of the source phrase. For our example, the first phrase pair popped from the queue is

```
es: no daba una bofetada a la bruja verde
en: did not slap the green witch
```

which is checked for inclusion in the ranked phrase table (line 9). A rank of 0 is assigned. The target subphrase is replaced with a pointer symbol

```
es: Maria no daba una bofetada a la bruja verde
en: Mary (0,0,0)
```

The integer values of the pointer triple have the following interpretation:

- The first is the difference of source and target start positions of the subphrase pair. Due to general monotonicity this yields smaller integers than positions.

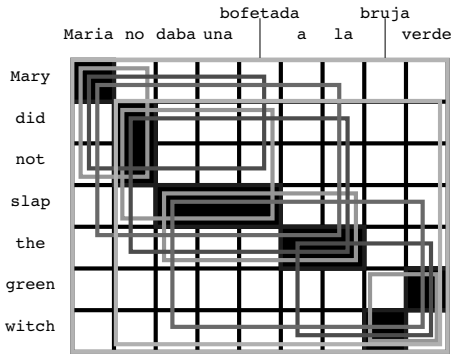

Figure 2. Archetypical example for phrase pair extraction by Knight and Koehn (2003)

- The second value is the distance of the right source subphrase boundary from the end of the encoded phrase.
- The last value is the rank of the selected subphrase pair.

All alignment points lying within the boundaries of the chosen subphrase pair are removed (line 12) and all subphrase pairs that overlap with the subphrase pair are deleted from the queue (line 13 and 14). Only one phrase pair is left in the queue:

```
es: Maria
en: Mary
```

Applying the same procedure again, the following encoded phrase pair is produced:

```
es: Maria no daba una bofetada a la bruja verde
en: (0,8,0) (0,0,0)
```

Target subphrases for which no substitution has been found are kept as plain words.

3.2. Decoding Procedure

A naive decoding procedure processes a mainly-binary tree (Figure 3) in potentially exponential time. However, if all target phrases for a sentence are considered, a dynamic programming algorithm with linear time-complexity per phrase can be constructed. Moses queries the phrase table processing sentences in a left-to-right fashion, starting with subphrases of length 1 and increasing the length until a limit is reached. Then it moves to the next word, starting at length 1. Hence, if a phrase is retrieved, its prefixes have already been processed. If all queried phrases are cached for decoding and all phrases used for decoding are cached for look-up, the total number of phrase table accesses is the same as in a linear phrase table. With caching, a target phrase for "Maria no daba una bofetada" would be found immediately, avoiding the descent into the left branch of the graph. The subphrase "a la" will still be processed, but when Moses queries that phrase, it will be retrieved from the cache.

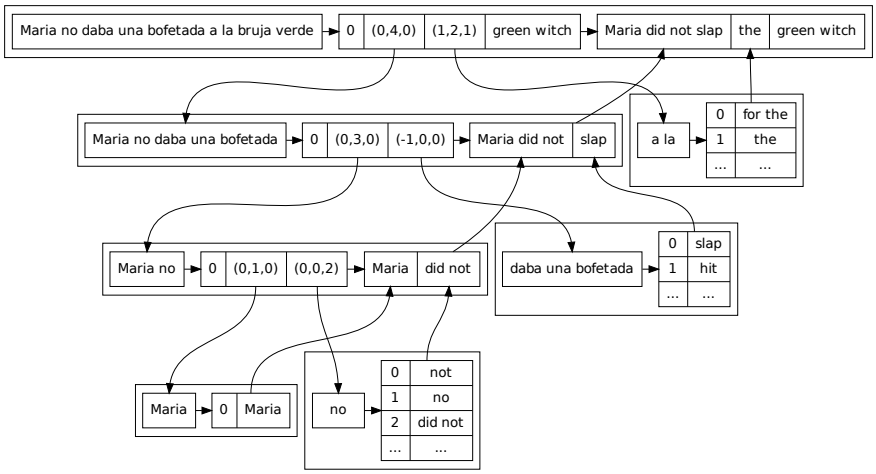

Figure 3. Phrasal Rank-Decoding without Caching

This is implemented in the algorithm in Figure 4. A target phrase collection for a source phrase is created. If a target phrase with the given rank has been seen before, it is retrieved from the cache. Otherwise an encoded version is loaded from the phrase table and passed to DecodeTargetPhrase. If $\hat{t}_j$ is a plain word, the symbol is concatenated with the decoded target phrase. If not, a pointer for the subphrase $\mathbf{s}'$ is reconstructed and the rank r' of the target phrase is determined. If the target subphrase $\mathbf{t}'$ has been decoded before, it is retrieved from the cache, else the encoded versions of $\mathbf{t}'$ and $\hat{A}'$ are fetched from the phrase table. $\mathbf{t}'$ and A' are obtained by recursively calling DecodeTargetPhrase. The decoded target subphrase is then concatenated with the current target phrase and subphrase alignment points are added to the output alignment, shifted accordingly. Results are cached before return.

4. Results

Coppa, the Corpus Of Parallel Patent Applications (Pouliquen and Mazenc, 2011) is used for phrase table generation. It comprises ca. 8.7 million parallel segments, 198.8 million English and 232.3 million French tokens. The generated phrase table consists of 247 million phrase pairs. All phrase table variants include alignment information, a requirement for several WIPO applications. Performance test were conducted on an Amazon EC2 server with 8 cores and 70 GB RAM. The first unique 1000 sentence pairs from the WIPO test set[1] are translated for performance tests using all

[1] `http://www.wipo.int/patentscope/translate/coppa/testset2011.tmx.tgz`

```
1  Function GetTargetPhraseCollection(s)
2      T ← ⟨ ⟩; r ← 0
3      while r < NumberOfTargetPhrases(s) do
4          if InCache(s, r) then
5              ⟨t, A⟩ ← GetFromCache(s, r)
6          else
7              ⟨t̂, Â⟩ ← GetFromPhraseTable(s, r)
8              ⟨t, A⟩ ← DecodeTargetPhrase(s, r, t̂, Â)
9          T ← T · ⟨⟨t, A⟩⟩
10         r ← r + 1
11     return T
12 Function DecodeTargetPhrase(s, r, t̂, Â)
13     t ← ⟨ ⟩; A ← Â
14     j ← 0
15     while j < |t̂| do
16         if Type(t̂_j) = Pointer then
17             ⟨k, l, r′⟩ ← t̂_j
18             i ← k + |t|
19             m ← |s| − l + 1
20             s′ ← Substring(s, i, m)
21             if InCache(s′, r′) then
22                 ⟨t′, A′⟩ ← GetFromCache(s′, r′)
23             else
24                 ⟨t̂′, Â′⟩ ← GetFromPhraseTable(s′, r′)
25                 ⟨t′, A′⟩ ← DecodeTargetPhrase(s′, r′, t̂′, Â′)
26             t ← t · t′
27             A ← A ∪ {⟨i + i′, j + j′⟩ : ⟨i′, j′⟩ ∈ A′}
28         else if Type(t̂_j) = Word then
29             t ← t · ⟨t̂_j⟩
30         j ← j + 1
31     AddToCache(s, r, ⟨t, A⟩)
32     return ⟨t, A⟩
```

Figure 4. Retrieving a set of target phrases

cores. Results are reported in Table 1 for: translation model size (Files), peak resident memory usage (RSS), peak operation system cache usage (Cached), warm-up time (Load), translation time without warm-up (Trans.), and time until the first translation is produced after warm-up (First). Before za "1st run" the operation system I/O caches have been dropped. During the "2nd run" all I/O caches of the first run are available.

System name	Memory (GB)			1st run (s)			2nd run (s)		
	Files	RSS	Cached	Load	Trans.	First	Load	Trans.	First
Moses	22.01	2.54	3.48	14	251	9	0	151	9
Moses+LR	68.40	4.00	5.84	15	470	8	3	215	8
Compact	4.77	1.76	4.85	18	195	2	4	135	1
Compact+LR	6.19	2.02	5.84	17	330	4	4	192	1
PR-Enc	2.93	1.84	3.98	15	170	1	3	137	<1
PR-Enc+LR	4.36	2.11	4.69	15	256	2	3	199	<1
PR-Enc (mem)	2.93	4.49	4.16	51	137	<1	5	138	1
PR-Enc+LR (mem)	4.36	5.91	5.59	66	196	<1	8	198	<1

Table 1. Comparison of phrase table implementations

We compare the following configurations: the standard Moses binary phrase table ("Moses"), the compact phrase table from Junczys-Dowmunt (2012a) without any encoding methods ("Compact") and with PR-Enc ("PR-Enc"). Additionally, phrase tables are combined with corresponding implementations of lexical reordering models ("+LR"), i.e. the Moses phrase table is used with the Moses binary reordering table, the compact phrase table is combined with a reordering model based on the compact implementation. In-memory variants of the compact phrase and reordering tables are denoted by "(mem)". All systems use the same 3-gram KenLM (file size 1.1 GB), which is responsible for a part of the load time and memory usage.

To sum up the results in Table 1: during second runs, speed is nearly identical for both variants of the compact table — with or without PR-Enc — and always better than for the Moses binary tables. During first runs, however, we see how reduced file size and reduced disk access result in increased speed. The complex PR-Enc decompression procedure has only a minor influence on speed, visible during second runs. PR-Enc reduces the size of the compact phrase table by another 39 percent. Compared to the Moses phrase table, size reduction reaches an order of magnitude, for lexical reordering models even more (46.4 GB versus 1.4 GB). Memory requirements for on-disk access are also more modest. If the phrase table and reordering table are loaded into memory, RSS memory usage is higher than for the Moses tables, but read cache usage is comparable. Translation speed, however, is much better despite increased load time. For more than 1000 sentences, cache usage will keep increasing in case of the Moses binary tables, for the compact in-memory version, it remains constant.

5. Conclusions

We introduced Phrasal Rank Encoding, a new method for the compression of translation phrase tables. The size reduction and performance improvement com-

pared to the Moses binary phrase table and a basic version of our phrase table are significant. Our implementation can be successfully used in place of the Moses binary phrase table. Experiments were performed for a medium sized phrase table and it is planned to repeat them with a phrase table that contains billions of phrase pairs. We suspect that increasing phrase table size might work to the advantage of our implementation, as should the usage of much weaker machines.

6. Usage

We include only basic instructions, see the Moses website for more information on compact phrase tables[2]nd lexical reordering tables[3]

Download and install the CMPH library[4], next recompile a current Moses version:

```
./bjam --with-cmph=/path/to/cmph
```

For PR-Enc, the phrase table should include alignment information. The following command creates a compact binary phrase table `phrase-table.minphr` from a standard text version with PR-Enc enabled by default:

```
mosesdecoder/bin/processPhraseTableMin -in phrase-table.gz \
-out phrase-table -use-alignment -threads 4
```

The compact phrase table variant without PR-Enc can be created by adding the option `-encoding None`. In the Moses config file, the filename stem `phrase-table` has to be specified and the type is to be set to 12, i.e.:

```
[ttable-file]
12 0 0 5 phrase-table
```

A compact lexical reordering model `reordering-table.minlexr` can be created with the following command:

```
mosesdecoder/bin/processLexicalTableMin -in reordering-table.gz \
-out reordering-table -threads 4
```

If only the file stem is given in the configuration file, the compact model is loaded instead of any other present lexical reordering model.

In-memory storage of the phrase table and the reordering model can be forced by running Moses with the options `-minphr-memory` and `-minlexr-memory` correspondingly. These can also be specified in the configuration file.

Acknowledgements

Part of this research was carried out at and funded by the World Intellectual Property Organization (WIPO) in Geneva.

[2] `http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc5`

[3] `http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc6`

[4] `http://sourceforge.net/projects/cmph/`

Bibliography

Anh, Vo Ngoc and Alistair Moffat. Index Compression using Fixed Binary Codewords. In *Proceedings of the 15th Australasian Database Conference*, volume 27, pages 61–67, 2004.

Callison-Burch, Chris, Colin Bannard, and Josh Schroeder. Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 255–262, 2005.

Conley, Ehud S. and Shmuel T. Klein. Using Alignment for Multilingual Text Compression. *International Journal of Foundations of Computer Science*, 19(1):89–101, 2008.

Germann, Ulrich, Eric Joanis, and Samuel Larkin. Tightly Packed Tries: How to Fit Large Models into Memory, and Make them Load Fast, Too. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 31–39, 2009.

Huffman, David. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, 1952.

Johnson, J. Howard, Joel Martin, George Fost, and Roland Kuhn. Improving Translation Quality by Discarding Most of the Phrasetable. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 967–975, 2007.

Junczys-Dowmunt, Marcin. A Phrase Table without Phrases: Rank Encoding for Better Phrase Table Compression. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 245–252, 2012a.

Junczys-Dowmunt, Marcin. A Space-Efficient Phrase Table Implementation Using Minimal Perfect Hash Functions. In *Proceedings of 15th International Conference on Text, Speech and Dialogue*, volume 7499 of *Lecture Notes in Computer Science*, pages 320–328. Springer Verlag, 2012b.

Knight, Kevin and Philipp Koehn. What's New in Statistical Machine Translation. Tutorial at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2003.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, 2007.

Pouliquen, Bruno and Christophe Mazenc. COPPA, CLIR and TAPTA: Three Tools to Assist in Overcoming the Language Barrier at WIPO. In *Proceedings of Machine Translation Summit XIII*, 2011.

Sadat, Fatiha, Howard Johnson, Akakpo Agbago, George Foster, Joel Martin, and Aaron Tikuisis. Portage: A Phrase-based Machine Translation System. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 129–132, 2005.

Sanchez-Martinez, Felipe, Rafael C. Carrasco, Miguel A. Martinez-Prieto, and Joaquin Adiego. Generalized Biwords for Bitext Compression and Translation Spotting. *Journal of Artificial Intelligence Research*, pages 389–418, 2012.

Scholer, Falk, Hugh E. Williams, John Yiannis, and Justin Zobel. Compression of Inverted Indexes for Fast Query Evaluation. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229, 2002.

Zens, Richard and Hermann Ney. Efficient Phrase-table Representation for Machine Translation with Applications to Online MT and Speech Translation. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2007.

Zens, Richard, Franz Josef Och, and Hermann Ney. Phrase-Based Statistical Machine Translation. In *Proceedings of the 25th Annual German Conference on AI: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 18–32. Springer Verlag, 2002.

Ziv, Jacob and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transaction on Information Theory*, 23(3):337–343, 1977.

Address for correspondence:
Marcin Junczys-Dowmunt
junczys@amu.edu.pl
Information Systems Laboratory,
Faculty of Mathematics and Computer Science,
Adam Mickiewicz University
ul. Umultowska 87
61-614 Poznań, Poland