Packed Rules for Automatic Transfer-Rule Induction

Yvette Graham and Josef van Genabith
National Centre for Language Technology,
School of Computing,
Dublin City University,
Dublin 9, Ireland
josef,ygraham@computing.dcu.ie

Abstract

We present a method of encoding transfer rules in a highly efficient packed structure using contextualized constraints (Maxwell and Kaplan, 1991), an existing method of encoding adopted from LFG parsing (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001). The packed representation allows us to encode $O(2^n)$ transfer rules in a single packed representation only requiring $O(n)$ storage space. Besides reducing space requirements, the representation also has a high impact on the amount of time taken to load large numbers of transfer rules to memory with very little trade-off in time needed to unpack the rules. We include an experimental evaluation which shows a considerable reduction in space and time requirements for a large set of automatically induced transfer rules by storing the rules in the packed representation.

1 Introduction

Probabilistic Transfer-Based Machine Translation is one of several current approaches to machine translation that combine data-driven statistical methods with the use of linguistic information (Quirk et al., 2005; Koehn and Hoang, 2007; Ding and Palmer, 2005; Charniak et al., 2003; Lavie, 2008; Riezler and Maxwell, 2006; Bojar and Hajič, 2008).

Traditionally, transfer rules were manually developed. Recently, methods of automatically inducing transfer rules from bilingual corpora have emerged (Hajič et al., 2002; Eisner, 2003; Bojar and Hajič, 2008; Riezler and Maxwell, 2006). Acquiring transfer rules automatically from bilingual corpora has several advantages. One obvious advantage is that automatic methods of rule induction are much quicker than manual rule development. This means that a much larger quantity of transfer rules can now be produced.

Riezler and Maxwell (2006) use feature structures of the Lexical Functional Grammar (LFG) formalism (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) for deep transfer. They impose a limit of a maximum of three *primitive* rules to construct a *complex* rule [1]. We believe removing arbitrary limits on the number of transfer rules induced could result in improved translations, and therefore we wish to induce as many different size rules as possible from a pair of parsed training sentences [2]. Short rules [3] are needed for high coverage of unseen sentences, but where possible larger rules [4] are preferred so as to increase the likelihood of a fluent target language sentence (all other things being equal).

Another issue for transfer rule induction is the amount of linguistic information that should be kept in the transfer rules. We would like to investigate the effects of keeping all or most of the linguistic information in the rules. If we both increase the number of induced rules and increase the amount of infor-

[1] Riezler and Maxwell (2006) construct *primitive* transfer rules using SMT phrases and then construct larger rules by combining contiguous primitive rules.

[2] The notion of different size rules we refer to is related to different length phrases in Phrase-based SMT.

[3] Rules that cover a small part of the source language structure.

[4] Rules that cover a large part of the target language structure.

mation contained in the rules, storing the rules in the conventional way of enumerating each rule separately will require large amounts of storage space and time to load the rules to memory. We address these problems by providing a packed data structure that efficiently stores large numbers of linguistically rich transfer rules, greatly reducing both the required storage space and load time.

The paper is structured as follows. In Section 2 we describe dependency-based transfer rules, Section 3 describes in detail our packed transfer rule representation, Section 4 describes an algorithm for unpacking the transfer rules. Finally, in Section 5 we report an experimental evaluation in which we extract a large number of transfer rules automatically from a bilingual corpus and compare the space and time requirements of the packed representation to that of storing each rule separately. Section 6 describes our conclusions and future work.

2 Dependency-Based Transfer-Rules

In our research we use LFG f-structures as the intermediate representation for transfer. F-structures are attribute-value structure encodings of bilexical labelled dependencies. In order to automatically induce transfer rules from a source and target f-structure pair, correspondences between pairs of source and target local f-structures are drawn using the predicate (PRED) of the local f-structures. Figure 1(a) shows an example f-structure pair with correspondences between local f-structures depicted by lines linking the predicates. F-structures encode the grammatical relations between the words of a sentence and this motivates their use as a representation for transfer-based machine translation. Sentences often contain long distance dependencies between words. One advantage of using f-structures for transfer-based machine translation is that two non-adjacent dependent words in a sentence are adjacent in the f-structure representation. In addition to these grammatical dependencies, the f-structure also contains information about the atomic grammatical features of words, such as *case*, *number*, *person* and *tense*. On the LHS of a transfer rule the atomic features are useful to guide translation by choosing a rule that appropriately *fits* the f-structure of the source language sentence, and on the RHS of the rules the atomic features are needed to correctly inflect the words in the target language sentence during generation.

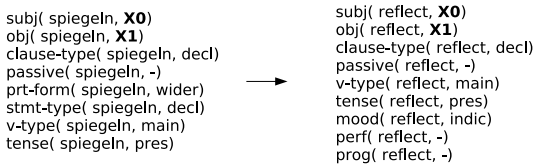

Figure 3: Example Constraint-based Encoding for Transfer Rule of Figure 2(a)

There are many ways to visualize an f-structure. In Figure 1(a) the f-structure is shown in the conventional LFG format [5]. Figure 1(b) shows a simplified graph-based visualization we use for most of the examples in this paper. Each local f-structure is represented by a node in the dependency structure labelled by its predicate value, with branches labelled with the grammatical dependencies between local f-structures [6].

Riezler and Maxwell (2006) automatically induce transfer rules composed of a snippet of the original source language f-structure on the LHS and a snippet of the target language f-structure on the RHS. Figure 2 shows a subset of the rules that can be induced from the f-structure pair shown in Figure 1. In a transfer rule, corresponding leaf-level arguments can be replaced by a variable, X_i, on either side of the rule to map equivalent arguments in the LHS structure to the appropriate place in the RHS structure. For example, the rule in Figure 2(a) maps the subject of *spiegeln* to the subject of *reflect* and the object of *spiegeln* to the object of *reflect*. F-structure based transfer rules are each stored as two sets of constraints, encoding the LHS and RHS of the rule, respectively. For every dependency relation that exists between two words in the sentence, a constraint will encode this relation. Figure 3 shows the transfer rule in Figure 2(a) represented in terms of constraints [7].

[5] Without atomic features and values.

[6] Atomic features and reentrancy are left out of the simplified representation. Figure 1(a) shows an example of reentrancy (German local f-structure 1, value of TOPIC). The transfer rules we induce do contain the atomic grammatical feature and reentrancy information.

[7] With atomic features and values.

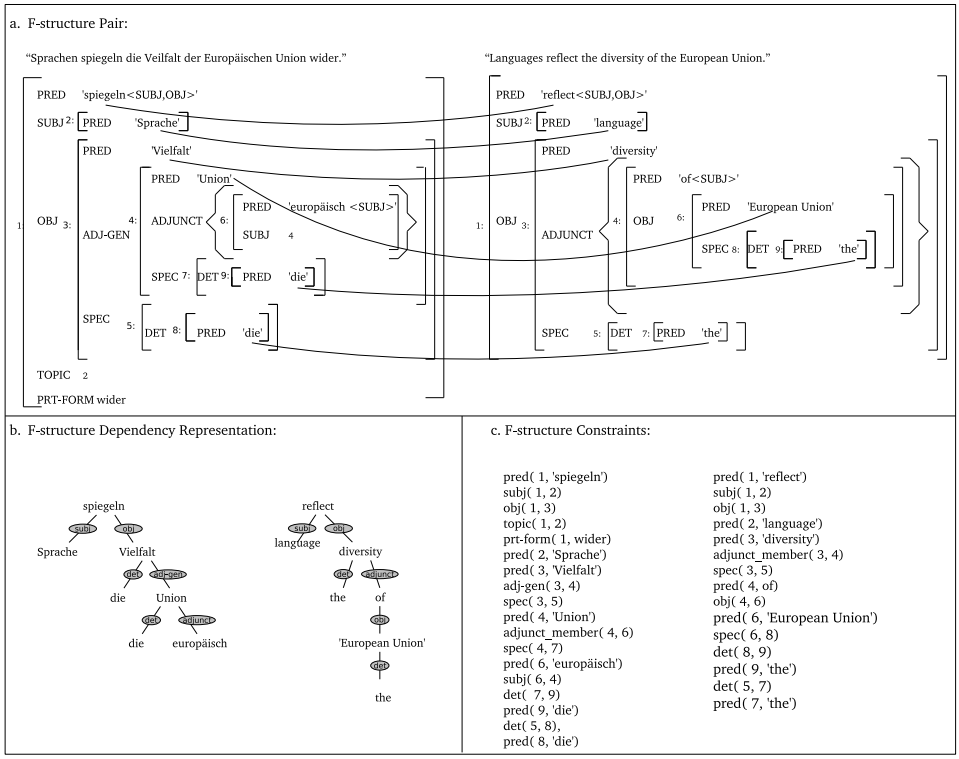

Figure 1: Example (a) F-structure Pair, (b) Dependency Relations in Simplified Representation, (c) Constraint Encoding for the parsed Sentences *"Sprachen spiegeln die Vielfalt der Europäischen Union wider."* and *"Languages reflect the diversity of the European Union."*

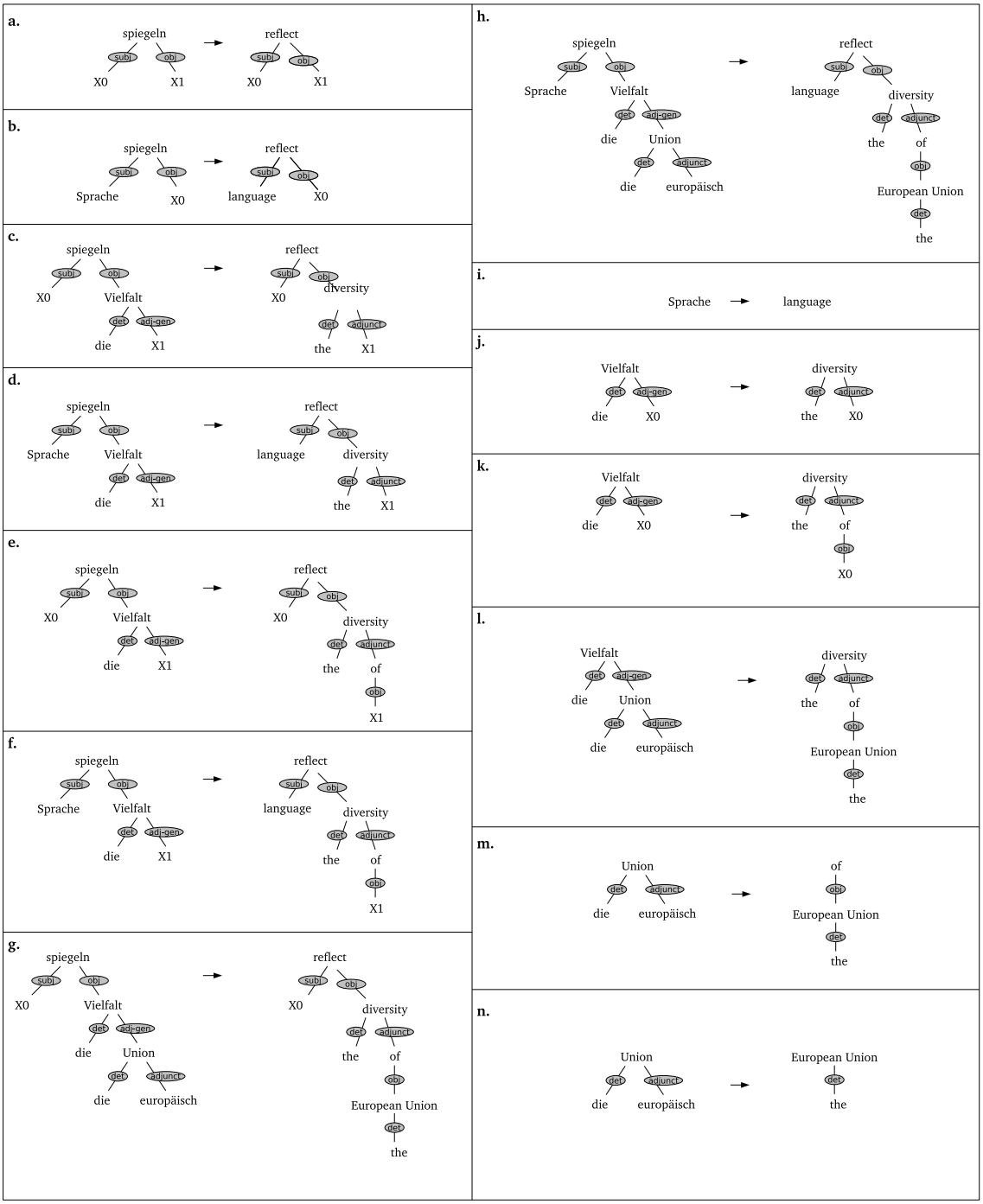

Figure 2: Example Transfer Rules: A subset of the transfer rules automatically induced from training f-structure pair shown in Figure 1.

For automatic rule induction, multiple transfer rules are extracted from each f-structure pair in the corpus. When several rules are extracted from a single f-structure pair, the resulting set of rules often contains a large amount of duplicated data. The only existing method of encoding transfer rules, to the best of our knowledge, involves enumerating the entire set of LHS and RHS constraints of each rule separately (Figure 3). This method of encoding results in a large number of constraints being recorded repeatedly, once for each transfer rule they end up in. Figure 2(h) shows the transfer rule that maps the entire source language set of constraints to the entire target language set of constraints. Every other rule induced from the f-structure pair will consist of a subset of these constraints. Recording each subsequent rule separately involves duplicating the constraints already recorded in rule 2(h). Since the number of transfer rules that can be induced from a given f-structure pair is O(2^n), where n is the number of local f-structures, storing the rules by enumerating each rule separately is highly inefficient.

3 Packed Representations for Dependency-Based Transfer Rules

Our method of storing transfer rules involves packing all the transfer rules induced from the same training f-structure pair into a single packed transfer rule data structure. Our packing method can encode O(2^n) transfer rules without duplicating any constraints. The packed rule representation uses contextualized constraints (Maxwell and Kaplan, 1991), a well-established method of encoding grammars in LFG parsing (Kaplan and Bresnan, 1982). Constraints are contextualized to improve the efficiency of processing disjunctive constraints of a grammar and thus simplify the encoding of grammatical possibilities, by allowing disjunctive statements as constraints. For example, the following constraint for the German word *die* is taken from (Maxwell and Kaplan, 1991):

case(*die*, nom) $\vee$ case(*die*, acc)

In this example, the value of the atomic feature, *case*, of the word *die* can be either *nominative* or *accusative*, depending on a given context.

We adopt this approach but adapt it for our own purposes of translation as opposed to parsing, and use contextualized constraints to encode that each constraint of the original f-structure pair may be included or excluded from a transfer rule, depending on the context. A packed rule contains a single instance of each of the constraints of the original f-structure pair with each constraint being assigned a context variable. This enables the encoding of O(2^n) rules in a single $O(n)$ size packed structure. Figure 4(a) shows an example packed rule structure [8] and Figure 4(b) shows the same rule using the simplified visualisation. The entire set of source language f-structure constraints forms the LHS of the packed rule, and the whole set of constraints of the target f-structure forms the RHS. Each constraint is given a context variable, A_i, which is used to determine whether or not the constraint should be included or excluded from a particular transfer rule.

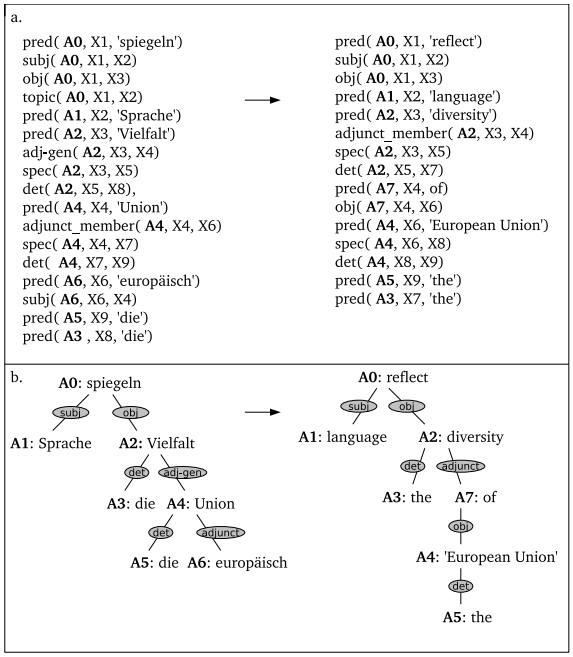

Figure 4: (a)Packed Transfer Rule with Context Variables (b) Simplified Representation of Transfer Rule

3.1 Assigning Context Variables to the F-structure Constraints

The constraints of each local f-structure on the LHS of the packed rule is labelled with a context variable (see variables A0-A6 on LHS of the rule in Fig-

[8] Atomic features and values have been left out of this diagram to save space.

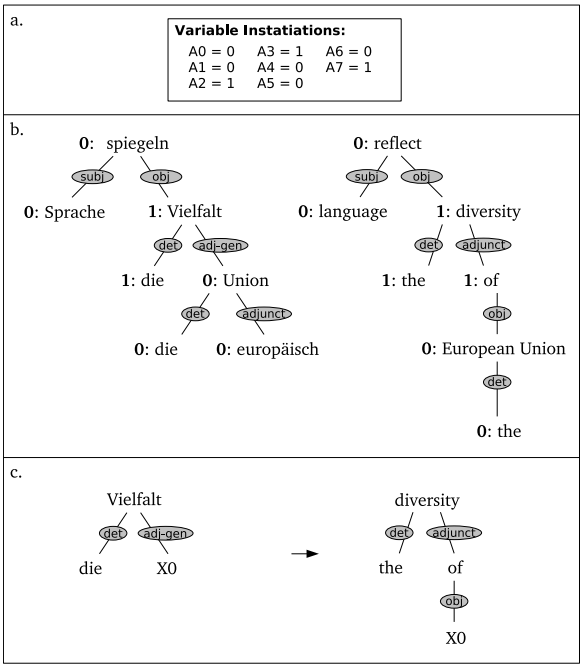

Figure 5: Context Variable instantiations producing a Transfer Rule: (a) the values each context variable of the packed rule is instantiated to, (b) the instantiated packed rule structure (c) the transfer rule produced by the variable instantiations

ure 4). The constraints of the corresponding local f-structure on the RHS is given the context variable of its LHS counterpart (see variables A0-A5 on the RHS of the rule in Figure 4). The constraints of any remaining unaligned local f-structures on the RHS are each assigned another distinct variable (see variable A7 in Figure 4). Extracting a particular transfer rule from the packed structure now simply involves assigning the value *true* to the constraints of the extracted rule and *false* to the constraints that are not part of the rule. Figure 5(a) shows one of the possible combinations of boolean values for the set of context variables given to the constraints of the packed rule shown in Figure 4. Figure 5(b) shows the packed rule with context variables instantiated and Figure 5(c) shows the rule that results by taking the constraints labelled *true* for this particular combination of boolean values [9].

[9]Notation: *true*=1 and *false*=0.

```
dep( <parent>, <optional dependents>, <obligatory dependents>).
```

Figure 6: Parent-Dependent Relation Statement Format

3.2 Contextualizing the Context Variables

The variables are given a context in order to constrain the types of rules that can be induced. Although it is possible to encode O(2^n) transfer rules within the packed structure, many of these rules are actually undesirable, and we therefore give a context to the variables to eliminate such rules.

Riezler and Maxwell (2006) define a contiguity constraint for transfer rules, that states that neither side of a rule may contain any gaps in the structure. To enforce this constraint we encode the relations between the context variables in a series of *parent-dependent relation statements*. A *relation statement* consists of the context variable of a single local f-structure, which we call the *parent* f-structure and specifies two lists, each containing context variables belonging to the *dependent* f-structures of the parent f-structure. Dependents of a parent are split into two disjoint sets, *optional* dependents and *obligatory* dependents. The inclusion in a transfer rule of the constraints labelled by an *obligatory dependent* context variable is entailed by the inclusion of the constraints of its *parent* in the rule. The constraints of an *optional dependent*, on the other hand, may either be included or omitted from a rule that includes its parent's constraints. The distinction between *obligatory* and *optional* dependents is useful to permit a rule induction algorithm to constrain the rules so that the inclusion of a given local f-structure in the rule entails the inclusion of one or more of its dependents [10]. Figure 6 shows the format of the relation statements and Figure 7 shows the *relation statements* that constrain the rules encoded in the packed structure of Figure 4.

4 Unpacking the Rules

Unpacking a transfer rule involves instantiating the context variables of the constraints that are part of the rule to true and the rest of the context variables in the packed structure to false. Unpacking all of

[10]However, if such constraints are not required, then it is possible to make all of the dependents in the rules *optional*.

Source Relations:

```
dep( A0, [A1,A2], [] ).
dep( A1, [], [] ).
dep( A2, [A4], [A3] ).
dep( A3, [], [] ).
dep( A4, [A6],[A5] ).
dep( A5, [], [] ).
dep( A6, [], [] ).
```

Target Relations:

```
dep( A0, [A1,A2], [] ).
dep( A1, [], [] ).
dep( A2, [A7], [A3] ).
dep( A3, [], [] ).
dep( A7, [A4], [] ).
dep( A4, [], [A5] ).
dep( A5, [], [] ).
```

Figure 7: Node Relations for Packed Rules of Dependency Structures in Figure 3

```
Algorithm:

Input: f-structure f and
       rule root r

Output: vector v of possible boolean
        values for the context
        variable of f

// A
if f==r then
  v = { true }
// B
else if parent(f)==false then
  v = { false }
// C
else if obligatory(f) then
  v = { true }
// D
else
  v = { true , false }
end if
```

Figure 8: Algorithm for Unpacking Transfer Rules from the Packed Rule Representation.

the rules from the structure involves assigning all possible combinations of true and false values to the context variables with respect to the contiguity constraint (Riezler and Maxwell, 2006) and relation statements. The algorithm in Figure 8 is applied to each constraint variable recursively in a top-down fashion starting with the context variable of the outermost f-structure. The set of solutions of the algorithm retrieves all unpacked rules from the packed representation.

5 Experimental Evaluation

In order to evaluate the effects of using the packed rule representation on the space required to store transfer rules, we ran an automatic transfer rule induction algorithm on sentences of the Europarl corpus. We restricted the test corpus to German-English sentences within the length range of 5 to 15 words. This resulted in 219,666 sentence pairs. We reserved 2000 of these sentences as a development set. Each side of the corpus was parsed with a monolingual LFG grammar (Butt et al., 2002; Riezler et al., 2002). The automatic rule induction algorithm used a bilingual dictionary (Richter, 2007) and Giza++ word alignments (Och and Ney, 2000) to align local f-structures. A packed transfer representation for each input f-structure pair was induced. All of the rules were then unpacked and counted. Our rule induction algorithm induced 5,148,874 transfer rules from the training data f-structure pairs. This resulted in an average of 23.65 rules being induced from each aligned f-structure pair [11]. The total time taken for the rule extraction algorithm was approximately 3.5 hours running the algorithm on 8 parallel processors (28 CPU hours).

In order to determine the effect of the packed representation we randomly selected 10 sets of 1000 sentences from the training data and examined the amount of space required to store the rules induced from these sentences in the packed representation and in the conventional way of storing rules, i.e. enumerating each rule separately. Time and space requirements were recorded for each of the 10 sets. The results for each set of rules are shown in Table 1, as well as the average of these results and an *All Rules Estimate*, i.e. an estimate of results for rules extracted from the entire training corpus [12]. The average number of rules induced from a set of 1,000 training sentences was 23,955. The packed representation reduces the average disk space required to store rules extracted from 1,000 training sentences from 95.96M to 7.17M , and the estimated disk space required for rules induced from the entire training corpus is reduced from 20.4G to 1.52G. The average amount of time taken to load 23,955 rules to memory is reduced from 207.4 seconds to 18.1 seconds, and the reduction in load time for the *All Rules Estimate* is from 12 hours 32 minutes to 1 hour 6 minutes. The average time to retrieve 23,955 rules from memory as expected is slightly increased from 1.8 seconds for the enumerated representation to 2.6 seconds for the packed representation, with the *All Rules Estimate* increasing from 6 minutes 31 seconds to 9 minutes 24 seconds. The average time to

[11] We refer to the number of rule tokens as opposed to types here.

[12] Resources did not permit unpacking all of the induced rules, therefore estimates were calculated. The *All Rules Estimate* was calculated by multiplying the average result for a set of 1,000 sentence pairs by 217.666.

		Disk Space		Write Time		Load Time		Unpacking Time	
Set	No. Rules	Enum.	Packed	Enum.	Packed	Enum.	Packed	Enum.	Packed
1	24,121	96.37M	7.2M	144s	128s	211s	17s	2s	3s
2	24,486	98.89M	7.16M	145s	127s	215s	19s	2s	3s
3	23,650	93.58M	7.17M	142s	133s	200s	18s	2s	2s
4	23,882	96.83M	7.22M	149s	118s	210s	18s	2s	2s
5	24,146	98.03M	7.15M	148s	128s	212s	17s	2s	3s
6	23,355	91.75M	7.1M	140s	128s	198s	21s	2s	3s
7	23,620	94.55M	7.21M	141s	142s	204s	18s	2s	2s
8	23,687	94.02M	7.11M	137s	124s	201s	17s	1s	3s
9	23,534	94.95M	7.12M	142s	120s	204s	17s	1s	3s
10	25,069	100.66M	7.26M	152s	231s	219s	19s	2s	2s
Average	23,955	95.96M	7.17M	144s	137.9s	207.4s	18.1s	1.8s	2.6s
All Rules Estimate	5,214,189	20.4G	1.52G	8h43m	8h20m	12h32m	1h06m	6m31s	9m24s

Table 1: Space and Time Comparison of Enumerated Rules (Enum.) Versus Packed Representation (Packed): Results shown are for rules induced from 10 randomly selected sets of 1000 training sentence pairs. An average result for a set of 1000 sentence pairs is also included and an estimate of the space and time requirements for inducing rules from the *All Rules Estimate* (M = megabytes, G = gigabytes, h = hours, m = minutes, s = seconds).

record 23,955 rules to disk was decreased from 144 seconds to 137.9 seconds, and from 8 hours 43 minutes to 8 hours 20 minutes for *All Rules Estimate*. The *All Rules Estimate* for the total number of rules is 5,214,189, which is close to the actual no. of induced rules mentioned above.

Our experimental evaluation clearly shows using the packed representation of transfer rules has two major advantages. Both the required disk space and time needed to load the rules to memory are reduced by more than a factor of 10. This is achieved with very little trade-off in the time taken by the unpacking algorithm that retrieves the rules from memory, as the estimate increase in time taken to retrieve 23,955 rules from memory is less than a second, and the *All Rules Estimate* shows an increase of less than three minutes to retrieve over five million rules from memory.

6 Conclusions

We presented a new method of encoding dependency-based transfer rules in an efficient packed representation. The method is a straight-forward approach that uses contextualized constraints and achieves the ability to encode $O(2^n)$ transfer rules in a $O(n)$ size data structure. Our experimental evaluation shows an impressive reduction in the amount of disk space required to store the transfer rules as well as a great reduction in the time needed to load a large number of rules to memory.

This method of packing transfer rules is currently used at the stage of transfer rule induction. However, we believe the packing scheme could be used for packing rules in the transfer chart. This could provide a means of reducing the memory needed for decoding and allow a larger beam size for beam search decoding, which could result in improved translation quality. In addition, the method could be applied to constrain factoring of linguistic features contained in transfer rules. We plan to carry out this research in the future.

7 Acknowledgements

The work presented in this paper was partly funded by a Science Foundation Ireland PhD studentship P07077-60101. We would like to thank Mary Hearne and John Maxwell for their assistance and comments.

References

Ondřej Bojar and Martin Čmejrek. 2007. Mathematical Model of Tree Transformations. *Project Euromatrix Deliverable 3.2*, Ufal, Charles University, Prague.

Ondřej Bojar and Jan Hajič. 2008. Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In *Proceedings of the third Workshop on Statistical Machine Translation*, Columbus, Ohio, June 2008.

Joan Bresnan. 2001. *Lexical-Functional Syntax.*, Blackweelm Oxford, 2001.

Miriam Butt, Helge Dyvik, Tracy H. King, Hiroshi Masuichi and Christian Rohrer. 2002. The Parallel Grammar Project. (grammar version 2005) In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), Workshop on Grammar Engineering and Evaluation*, pages 1-7. Tapei, ROC.

Eugene Charniak, Kevin Knight and Kenji Yamada. 2003. Syntax-based Language Models for Statistical Machine Translation. In *Proceedings of the Machine Translation Summit IX 2003*

Ilyas Cicekli and Altay Gvenir. 2003. Learning Translation Templates from Bilingual Translation Examples. In *Recent Advances in Example-based Machine Translation*, pages 255-286, M. Carl and A. Way (eds.)

Mary Dalrymple. *L*exical-Functional Grammar, Academic Press, San Diego, CA; London. 2001.

Steve DeNeefe, Kevin Knight, Wei Wang and Daniel Marcu. 2007. What Can Syntax-based MT Learn from Phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational*

Yuan Ding and Martha Palmer. 2005. Machine Translation Using Probabilistic Synchronous Dependency Insertion Grammars. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 541-548, Ann Arbor, June 2005.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 205-208, Sappora, July 2003.

Jan Hajič, Martin Čmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev and Owen Rambow. 2002. Natural Language Generation in the Context of Machine Translation. Technical Report. *NLP WS'02*, final report.

Ronald M. Kaplan, Tracy H. King and John T. Maxwell. 2002. Adapting existing grammars: the XLE experience. In *Proceedings of COLING 2002*, Taipei, Taiwan.

Ronald Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar, a Formal System for Grammatical Represenation. In Bresnan, J. editor, *The Mental Representation of Grammatical Relations*, pages 173-281, MIT Press, Cambridge, MA.

Philipp Koehn, Franz Josef Och and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the HLT-NAACL 2003*, pages 48-54, Edmonton, May/June 2003.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868876, Prague, June 2007.

Alon Lavie. 2008. . Stat-XFER: A General Search-Based Syntax-Driven Framework for Machine Translation. In *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics*, pages 362-375, Haifa, Israel, 2008.

John T. Maxwell III and Ronald M. Kaplan. 1991. A Method for Disjunctive Constraint Satisfaction. In *Current Issues in Parsing Technology*, Masaru Tomita editor, pages 173-190, Kluwer Academic Publishers.

Franz Josef Och, Christoph Tillmann Hermann and Ney. 2000. Improved alignment modesl for statistical machine translation. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing* (EMNLP'99). College Park, MD, pages 20-28.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 440-447.

Chris Quirk, Arul Menezes and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics (ACL)*, Ann Arbor, June 2005, pages 271-279.

Franz Richter. 2007. The German-English word list. *http://ftp.tu-chemnitz.de/pub/Local/urz/ding/de-en* Copyright (c) Frank Richter 1995-2007.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using Lexical Functional Grammar and discriminitive estimation techniques . (grammar version 2005) In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL)*, Philadelphia, July 2002.

Stefan Riezler and John T. Maxwell III. 2006. Grammatical Machine Translation. In *Proceedings of HLT-ACL*, pages 248-255, New York.

Petr Sgall, Eva Hajicova and Jarmilla Panevova. 1986. *The Meaning of the Sentence and its Semantic and Pragmatic Aspects*. Dordrecht: Reidel and Prague: Academia 1986.