Principled Induction of Phrasal Bilexica

Markus Saers and **Dekai Wu**
Human Language Technology Center
Dept. of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
{masaers|dekai}@cs.ust.hk

Abstract

We aim to replace the long and complicated, pipeline employed to produce probabilistic phrasal bilexica with a theoretically principled, grammar based, approach. To this end, we introduce a learning regime to learn a phrasal grammar equivalent to linear transduction grammars. The stochastic version of this new grammar type also has the property that the set of biterminals constitute a natural probability distribution, making it similar to a probabilistic translation lexicon. Since we learn a phrasal grammar, we are, in effect, learning a probabilistic phrasal bilexicon. As a proof of concept, we show that phrasal bilexica, induced in this manner, can be used to improve the performance of a traditional phrase-based SMT system.

1 Introduction

The aim of the paper is to show that it is possible to replace the long and complicated pipeline that turns a parallel corpus into a phrasal bilexicon, with a theoretically principled, grammar based approach. The ultimate aim, towards which this paper is a step, is to eliminate the mismatch between learning and translation that plagues the statistical machine translation SMT systems of today.

When phrase-based statistical machine translation (SMT) systems replaced the token-based, a significant boost to translation quality was experienced. Rather than having to rely on broad generalizations, the system simply memorizes chunks of text and their translation. The degrees of freedom to make errors are thus severely restricted, making the system more accurate at the cost of storing huge amounts of fixed chunks. Since a surface-based system has no mechanism for generalizing in a systematic way, this is a good work-around.

Although structured SMT systems are capable of making generalizations beyond the scope of surface-based systems, it is also imperative to be able to handle chunks of text. This is the correct way to capture phenomena such as figures of speech, whose translations go beyond the valid generalizations of the language. However, distinguishing between the phrasal biterminals that can be handled by generalization and the ones that cannot, is a hard problem. Indeed, finding a candidate set of phrasal biterminals is a hard problem. Naïvely enumerating all possible phrasal entries found in a parallel corpus and determine their probability by relative frequency, is doomed to fail because of the sheer amount of possible phrasal entries. The same is true for constructing transduction grammars with phrasal biterminals: the size of the grammar makes it impossible to handle, which is why transduction grammars are usually thought of as being restricted to handle single-token terminals. Although formal proofs are often easier to construct for the normal form of a grammar, the grammar itself is not restricted to this normal form; and although the runtime complexities of parsers are tied to the normal form of the grammar being parsed, algorithms generally exists to normalize the grammar on the fly, making the *existence* of a normal form is the important part.

In this paper, we introduce a method for iteratively extending biterminals. Rather than collecting all biterminals that could occur in the training corpus, we are collecting only those that could occur in a valid parse tree in the training cor-

Mikel L. Forcada, Heidi Depraetere, Vincent Vandeghinste (eds.)
Proceedings of the 15th Conference of the European Association for Machine Translation, p. 313–320
Leuven, Belgium, May 2011

pus (according to the grammar we have induced so far). We thus start by inducing a transduction grammar based on single-token terminals directly from the parallel corpus using expectation-maximization (EM). When it has stabilized, we collect all biterminal pairs that could form larger biterminals and incorporate them into the grammar to produce a multi-token transduction grammar. The process is then repeated until large enough units are learned. Since maximum likelihood learning methods such as EM tends to favor longer chunks over shorter, we introduce a length penalty for multi-token terminals.

Since induction of transduction grammars is very time consuming, we opt to view the parallel corpus as a *linear transduction* (Saers et al., 2010b; Saers, 2011). This assumption allows us to use something that is equivalent to linear transduction grammars (LTGs), which can approximate the search for a parse forest given a sentence pair in linear time. LTGs do not, however, have an explicit biterminal concept, making it non-trivial to map the grammar to a probabilistic bilexicon. To fix this, we introduce preterminalized linear inversion transduction grammars (PLITGs), which will allow the desired parameterization.

Learning a stochastic bracketing PLITG from a parallel corpus is equivalent to building a probabilistic bilexicon based on this corpus. Iteratively extending the biterminals of the grammar makes them phrasal, giving us a probabilistic phrasal bilexicon. This constitutes the key component of a standard phrase-based SMT system, making it easy to compare our approach to the standard approach.

2 Background

There has been some recent interest in bringing structure into SMT, with various approaches to how grammars can be learned and used in the translation process. Transduction grammars are grammars that generate pairs of strings—bistrings—rather than just strings. Since a transduction is a set of bistrings, just as a language is a set of strings, it can be regarded as a relation between two languages. The most well-studied and used transductions are the finite-state transductions, handled by finite-state transducers and finite-state transduction grammars. These are very efficient, but not expressive enough to generate the structural differences we see between languages on a sentence level. A more powerful class of transductions is the syntax-directed transductions, generated by syntax-directed transduction grammars (SDTGs). These were introduced by Lewis and Stearns (1968) and further developed and formalized by Aho and Ullman (1972). If the finite-state transductions can be said to be the bilingual case of finite-state languages, syntax-directed transductions can be said to be the bilingual case of context-free languages. By *bilingual case*, we mean that the transduction relates two languages of that class to each other.

The original notation for SDTGs called for conventional CFG-rules in language F augmented with rephrased E productions, either in curly brackets, or comma separated. To differentiate identical nonterminal symbols, indices were used (the bag of nonterminals for the two productions are equal by definition).

$$\begin{aligned} &A \rightarrow B^{(1)}\ a\ B^{(2)} \quad \{x\ B^{(1)}\ B^{(2)}\} \\ &= A \rightarrow B^{(1)}\ a\ B^{(2)},\ x\ B^{(1)}\ B^{(2)} \end{aligned}$$

The semantics of the rules is that one nonterminal rewrites into a bag of nonterminals that is distributed independently in the two languages, and interspersed with any number of terminal symbols in the respective languages. As with CFGs, the terminal symbols can be factored out into preterminals with the added twist that they are shared between the two languages, since preterminals are formally nonterminals. The above rule can thus be rephrased as:

$$\begin{aligned} &A \rightarrow B^{(1)}\ \left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]\ B^{(2)},\ \left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]\ B^{(1)}\ B^{(2)} \\ &\left[\begin{smallmatrix}a\\x\end{smallmatrix}\right] \rightarrow a,\ x \end{aligned}$$

where $\left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]$ is a preterminal symbol unique for the translation of the terminal a to the terminal x. In this way, rules producing nonterminals and rules producing terminals can be separated. Since, by definition, only nonterminals are allowed to move, their movement can be represented as the original sequence of nonterminals and a permutation vector as follows:

$$\begin{aligned} &A \rightarrow B\ \left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]\ B\ ;\ 1,0,2 \\ &\left[\begin{smallmatrix}a\\x\end{smallmatrix}\right] \rightarrow a,\ x \end{aligned}$$

To keep the reordering as monotone as possible, the terminals a and x can be produced separately, but doing so eliminates any possibility of parameterizing their lexical relationship. Instead, the individual terminals are paired up with the empty

string (ϵ):

$$A \rightarrow \left[\begin{smallmatrix}\epsilon\\ x\end{smallmatrix}\right]\ B\ \left[\begin{smallmatrix}a\\ \epsilon\end{smallmatrix}\right]\ B\ ;\ 0,1,2,3$$
$$\left[\begin{smallmatrix}a\\ \epsilon\end{smallmatrix}\right] \rightarrow a,\ \epsilon$$
$$\left[\begin{smallmatrix}\epsilon\\ x\end{smallmatrix}\right] \rightarrow \epsilon,\ x$$

Lexical rules involving the empty string are referred to as *singletons*. Whenever a preterminal is used to pair up two terminal symbols, we refer to that pair of terminals as a *biterminal*.

The computational complexity of translating with an existing SDTG (between the two related languages) is comparable to parsing with context-free grammars. This makes them a desirable alternative to surface-based translation methods, which represent an NP-complete search problem that needs to be heavily pruned to be practically useful. The problem with SDTGs is that they are very time consuming to learn from parallel corpora. Whereas all context-free grammars can be reduced to an equivalent grammar in a two-normal form, SDTGs cannot. With an arbitrary-rank SDTG, parsing a sentence pair requires $\mathcal{O}(n^{2n+2})$ time, which makes induction from parallel corpora intractable.

One way to make induction tractable is to restrict the search to a subset of the syntax-directed transductions. This approach was pioneered in a series of papers by Wu (1995a; 1995b; 1996; 1997), where inversion transductions and the corresponding inversion transduction grammars (ITGs) were introduced. This restricted set of grammars can parse a sentence pair in $\mathcal{O}(n^6)$ time, making it tractable but not practical to induce grammars from data. Several attempts have been made to approximate biparsing, see for example Zhang et al. (2008), Saers et al. (2009) and Haghighi et al. (2009).

An even more aggressive restriction to SDTGs was proposed in Saers et al. (2010b). Where the original ITGs allow branching, the introduced linear ITGs (LITGs) do not, allowing for biparsing in $\mathcal{O}(n^4)$ time. By approximating the search, linear time complexity is attainable. Saers (2011) later introduced the more general class of linear transduction grammars (LTG), which are equivalent to LITGs in terms of generative capacity.

Searching for linear transductions rather than inversion transductions or full syntax-directed transductions makes the grammar induction tractable. It does, however, also make the biterminals – the building blocks of our bilexicon – intricately entangled with the nonterminal symbols of the grammar. To address this, we introduce the class of preterminalized LITGs (Section 4), which has the property that each biterminal is associated with exactly one parameter. It does not, however, mean that we can induce a *phrasal* bilexicon. To do that, we would need a grammar with phrasal biterminals. Learning such a grammar poses a different kind of challenge: rather than time-constraints, we are faced with space-constraints. Induction with EM requires the explicit storage of every rule that could be used anywhere in the training corpus. Naturally, the number of rules explodes when we need to consider all the possible phrasal translations that could have occurred somewhere in the training corpus as individual rules in our grammar.

Our solution to this problem is to start with a token based grammar, and then extract larger biterminals that occur in the parse forest of the simpler grammar. This allows us to avoid enumerating phrasal biterminals that would likely be weeded out by EM anyway, and thus saves considerable amounts of space (Section 5). Another approach to this problem is reported in Blunsom and Cohn (2010), involving sampling. Up until the point that a rule is actually sampled, it can be implicitly represented with the probability distribution it is to be drawn from.

3 Linear Transduction Grammars

Linear transduction grammars constitute the natural bilingualization of linear grammars (LGs). This class of grammar has received relatively little attention, mostly because there is no obvious use for it. LGs lie between finite-state grammars and context-free grammars in computational complexity, but all you get is palindromes. Another way to look at LGs is that they relate the beginning of a string to the end of it. In this sense, a linear language defines a transduction between beginnings and ends, and they are indeed equivalent to finite-state transduction grammars as noted early on by Rosenberg (1967) and Ginsburg and Spanier (1966) independently.

A linear transduction grammar (LTG) relates two linear languages to each other. As linear languages relate two finite-state languages to each other, a linear transduction can be said to relate four finite-state languages to each other, with the caveat that two of them are also related to each other.

Definition 1. *An* LTG *over languages* L_1 *and* L_2

is a tuple $G = \langle N, \Sigma, \Delta, S, R\rangle$ where N is a finite, nonempty set of nonterminal symbols, Σ is a finite, nonempty set of L_1 symbols, Δ is a finite, nonempty set of L_2 symbols, $S \in N$ is the designated start symbol, and R is a finite, nonempty set of linear transduction rules on the forms:

$$A \rightarrow a/x\ B\ b/y,$$
$$A \rightarrow a/x$$

where $A, B \in N$, $a, b \in \Sigma^*$ and $x, y \in \Delta^*$.

Linear inversion transduction grammars (LITGs), on the other hand, are inversion transduction grammars, ITGs (Wu, 1997) that have been subjected to a linearity constraint. An ITG is a transduction grammar restricted to have only context-free rules, and only monotonic permutations (specifically identity or inversion permutation). By subjecting an ITG in normal form to a linearity constraint, we replace one of the nonterminal symbols with biterminal symbols, thereby significantly reducing the expressive power of the grammar, for a significant efficiency boost. LITGs were introduced in Saers et al. (2010b), and subsequently compared to full ITGs in Saers et al. (2010a). The rules in an LITG take the following forms:

$$\begin{aligned}
&A \rightarrow [a/x\ B],\\
&A \rightarrow \langle a/x\ B\rangle,\\
&A \rightarrow [B\ a/x],\\
&A \rightarrow \langle B\ a/x\rangle,\\
&A \rightarrow [a/\epsilon\ B], && (= A \rightarrow \langle a/\epsilon\ B\rangle)\\
&A \rightarrow [\epsilon/x\ B], && (= A \rightarrow \langle B\ \epsilon/x\rangle)\\
&A \rightarrow [B\ a/\epsilon], && (= A \rightarrow \langle B\ a/\epsilon\rangle)\\
&A \rightarrow [B\ \epsilon/x], && (= A \rightarrow \langle \epsilon/x\ B\rangle)\\
&A \rightarrow \epsilon/\epsilon
\end{aligned}$$

Where productions enclosed in angled brackets are read left-to-right in L_1, and right-to-left in L_2. Since it makes no different whether the empty string is to the left or right of the nonterminal, any inverting rule producing a singleton can be equivalently expressed as a straight rule.

4 Preterminalized LITGs

LTGs and LITGs have the unusual property that they are not closed under the addition of preterminals to the grammar, unlike ITGs or SDTGs, both of which allow adding preterminals while remaining ITGs or SDTGs. In contrast, in an LTG or LITG, replacing a biterminal with a preterminal violates the condition that the right hand side can only contain a single nonterminal.

This becomes problematic because to induce a bilexicon from a transduction grammar, we would like to have a natural probability distribution over the lexical entries present in the grammar, representing the lexical translation probabilities of word or phrase translations. This is not available for LTGs or LITGs, and to remedy the situation we introduce preterminalized LITGs (PLITGs).

A preterminal is to be understood as a nonterminal symbol that can only rewrite into terminal symbols. This retains the computational complexity of terminals, as identifying preterminals is comparable to scanning for terminals, while at the same time allowing for the desired parameterization.

Definition 2. *A* PLITG *over languages L_1 and L_2 is a tuple $G = \langle N, P, \Sigma, \Delta, S, R\rangle$, where N, Σ, Δ and S are the same as for* LTG*s, P is a finite, nonempty set of preterminal symbols and R is a finite, nonempty set of preterminal linear inversion transduction rules on the forms:*

$$\begin{aligned}
&A \rightarrow [BY], && A \rightarrow [YB],\\
&A \rightarrow \langle BY\rangle, && A \rightarrow \langle YB\rangle,\\
&A \rightarrow \epsilon/\epsilon, && X \rightarrow a/x
\end{aligned}$$

where $A, B \in N$, $X, Y \in P$, $a \in \Sigma^*$ and $x \in \Delta^*$.

Lemma 1. *For every* PLITG *we can construct an* LITG *that generates the same transduction.*

Proof. Let $G = \langle N, P, \Sigma, \Delta, S, R\rangle$ be a PLITG and let $G' = \langle N, \Sigma, \Delta, S, R'\rangle$ be the corresponding LITG where:

$$\begin{aligned}
R' = &\{A \rightarrow [a/xB] | A \rightarrow [XB], X \rightarrow a/x \in R\}\\
&\cup \{A \rightarrow [Ba/x] | A \rightarrow [BX], X \rightarrow a/x \in R\}\\
&\cup \{A \rightarrow \langle a/xB\rangle | A \rightarrow \langle XB\rangle, X \rightarrow a/x \in R\}\\
&\cup \{A \rightarrow \langle Ba/x\rangle | A \rightarrow \langle BX\rangle, X \rightarrow a/x \in R\}\\
&\cup \{A \rightarrow \epsilon/\epsilon | A \rightarrow \epsilon/\epsilon \in R\}
\end{aligned}$$

The grammars G and G' clearly generate the same transduction. □

Lemma 2. *For every* LITG *we can construct a* PLITG *that generates the same transduction.*

Proof. Let $G = \langle N, \Sigma, \Delta, S, R\rangle$ be an LITG and $G' = \langle N, P, \Sigma, \Delta, S, R'\rangle$ be the corresponding

PLITG where:

$$
\begin{aligned}
R' = &\{A \to [B\,[{}^{a}_{x}]], [{}^{a}_{x}] \to a/x | A \to [Ba/x] \in R\} \\
&\cup \{A \to [[{}^{a}_{x}]\,B], [{}^{a}_{x}] \to a/x | A \to [a/xB] \in R\} \\
&\cup \{A \to \langle B\,[{}^{a}_{x}]\rangle, [{}^{a}_{x}] \to a/x | A \to \langle Ba/x\rangle \in R\} \\
&\cup \{A \to \langle [{}^{a}_{x}]\,B\rangle, [{}^{a}_{x}] \to a/x | A \to \langle a/xB\rangle \in R\}
\end{aligned}
$$

where $[{}^{a}_{x}]$ is a preterminal symbol unique for the biterminal a/x, and P is the set of all unique preterminals introduced when building R'. The grammars G and G' clearly generate the same transduction. □

Theorem 3. PLITG*s and* LITG*s generate the same class of transductions.*

Proof. Follows from Lemmas 1 and 2. □

Corollary. *Since* PLITG*s are equivalent to* LITG*s (Theorem 3), and since* LITG*s are equivalent to* LTG*s (Saers, 2011),* PLITG*s are equivalent to* LTG*s.*

Adding preterminals as a dedicated class of symbols in the grammar is potentially controversial, but we believe that it is imperative in this specific case. For finite-state transduction grammars, preterminals are meaningless, as they merely change the label of a terminal symbol. For ITGs/SDTGs, there is no reason to separate the preterminals from the nonterminals, as they are merely a special case, and the formalism is already capable of handling the general case. For LTGs and LITGs, on the other hand, preterminals serve a very specific purpose. Since they only rewrite to biterminals, they are terminal in a computational sense (the length of their yield is constant with respect to the length of the string), while still grouping a number of biterminals into one unit that can be associated with a parameter during learning. It also has the side-effect of compacting the grammar, which is nice since we are concerned about the space complexity of adding phrasal rules to the grammar.

5 Extracting phrasal biterminals

The natural parameterization for PLITGs allows us to learn probabilistic bilexica through grammar induction on parallel corpora through expectation maximization (EM). The problem is that EM is limited to fit the given parameters to the data, and we cannot enumerate all the possible phrasal rules that could possibly be encountered in a corpus of any useful size. Our solution to this problem is to iteratively combine existing biterminals that cooccur in the data.

We start by building a grammar with all biterminals linking one token to another token (or the empty string) that could possibly be encountered in the training corpus. For efficiency reasons, we keep to a bracketing grammar, meaning that there is only one nonterminal and one preterminal symbol. Rather than setting their probability to a uniform distribution, we use relative cooccurrence frequency as a starting point. To collect counts for the singletons we assume that there is one empty string in each sentence. The structural rules share their probability mass equally. This has the effect of initializing the grammar such that:

$$
\begin{aligned}
p(A \to [AX]) &= 0.2 \\
p(A \to [XA]) &= 0.2 \\
p(A \to \langle AX\rangle) &= 0.2 \\
p(A \to \langle XA\rangle) &= 0.2 \\
p(A \to \epsilon/\epsilon) &= 0.2 \\
p(X \to a/x) &= \frac{c(a,x)}{C}
\end{aligned}
$$

where $c(\cdot,\cdot)$ is the corpus counting function and C is the corpus constant defined as:

$$
\begin{aligned}
c(a,x) &= \sum_{\langle \bar{e},\bar{f}\rangle \in \langle E,F\rangle} \sum_{\substack{e \in \bar{e} \uplus \{\epsilon\} \\ f \in \bar{f} \uplus \{\epsilon\}}} \begin{cases} 1 & \text{if } a = e, x = f \\ 0 & \text{otherwise} \end{cases}, \\
C &= \sum_{\langle \bar{e},\bar{f}\rangle \in \langle E,F\rangle} (|\bar{e}| + 1)(|\bar{f}| + 1)
\end{aligned}
$$

where $\langle E,F\rangle$ is the corpus, and $\uplus$ represents bag union.

We then run five iterations of expectation maximization with conditional grammar pruning to weed out the biterminals that are on the way out. Conditional grammar pruning is enforced by insisting that the conditional probability of a biterminal reaches over a threshold value. Given the threshold t, the pruning ensures that the following condition holds after each iteration:

$$
\frac{p(X \to a/x)}{\sum_{x'} p(X \to a/x')} \geq t \quad \text{if} \quad X \to a/x \in R
$$

and

$$
\frac{p(X \to a/x)}{\sum_{a'} p(X \to a'/x)} \geq t \quad \text{if} \quad X \to a/x \in R
$$

Even with a very low t-value (we used $t = 10^{-200}$), the set of rules is considerably reduced after only a few iterations of EM. Having pruned down the grammar, we proceed to collect new biterminals wherever we find two consecutive biterminals that can combine to a legal rule. As EM tends to favor longer biterminals, we also apply a length penalty. Consider the derivation:

$$\begin{aligned} A, A &\Rightarrow XA, AX \Rightarrow aA, Ax \\ &\Rightarrow aXA, XAx \Rightarrow abA, yAx \\ &\Rightarrow abXA, yXAx \Rightarrow abcA, yzAx \\ &\Rightarrow abc, yzx \end{aligned}$$

From this we could extract the rule $X \rightarrow bc/yz$, but not the rule $X \rightarrow ab/yx$. The reason is that the biterminal bc/yz occurs in the derivation whereas the biterminal ab/yx does not (the terminal yx is punctured by a nonterminal). To add in the penalty when collecting fractional counts, we modify each count such that, if we would ordinarily count it as occurring x times ($x \leq 1$), we instead count it as occurring x^λ times, where

$$\lambda = \begin{cases} \frac{|a|+|x|}{2} & \text{if } |a| + |x| > 1 \\ 1 & \text{otherwise} \end{cases}$$

where $|\cdot|$ is the length of a terminal defined as the number of tokens it contains. This penalty does not affect the original token-based subset of the rules, as they are not penalized.

6 Experimental setup

To test whether the phrasal bilexicon extracted from the above process learns anything useful, we use it in an existing phrase-based SMT system. We decided to stay close to the guidelines for the 2008 Workshop of Statistical Machine Translation (WMT08) with a few minor changes. Focusing on the French-English translation task, we limited the training data to sentence pairs where none of the two sentences were longer than 20 tokens (see Table 1). Our train–tune–test pipeline includes:

1. **Preprocessing tools:** tokenizer, corpus cleaner and case folder (supplied by the organizers).
2. **Language model:** 5-gram model using SRILM (Stolcke, 2002).
3. **Translation model:** Phrase-based model extracted and scored with with the Moses toolkit (Koehn et al., 2007) using the `grow-diag-final-and` heuristic (Koehn et al., 2005) on bidirectional word alignments obtained through IBM-models (Brown et al., 1993) and HMM-alignment (Vogel et al., 1996) using GIZA++ (Och and Ney, 2000).
4. **Tuning:** Minimum error-rate training (Och and Ney, 2002).
5. **Decoder:** Moses (Koehn et al., 2007).
6. **Postprocessing:** recaser (trained with the Moses toolkit) and detokenization (supplied by the organizers).
7. **Evaluation:** NIST (Doddington, 2002) and BLEU (Papineni et al., 2002).

Corpus	Size
French–English (train)	381,780 sent. pairs
French–English (tune)	2,000 sent. pairs
French–English (test)	2,000 sent. pairs
English (LM)	1,412,546 sentences

Table 1: Corpora used in the experiments.

This constitutes the baseline. Our system replaces the translation model with a phrasal bilexicon from PLITG induction. To isolate the effect of the bilexicon (which is the main focus of this paper), we refrained from using the more advanced reordering model that the Moses toolkit can build from alignments. This is a concept that is completely orthogonal to the bilexicon, and rather than simulating it when converting the PLITG to a bilexicon, we chose to leave it out.

The PLITG bilexicon was induced by combining existing biterminals three times, with five iterations of expectation maximization after each combination iteration. This means that the longest biterminal we could possibly get would have a total size of eight times eight tokens (2^3), which is fully comparable to the baseline system. We used the approximate biparsing algorithm described in Saers et al. (2010b) with $b = 50$.

Having two lexical sources could prove to be an advantage over a single source, and in testing this, we devised two strategies for phrasal bilexicon combination. The first is the cautious combination, where one bilexicon is allowed to dominate the other. The second is a weighted average

System	NIST	BLEU	Lexicon entries
Baseline	6.8439	27.13	7,340,369
PLITG	6.8025	26.71	2,926,745
CC	6.8505	27.05	9,797,754
WA	**6.9276**	**27.63**	9,797,754

Table 2: Results from the experiments. The PLITG system uses the induced phrasal bilexicon, the CC system uses a cautious combination of the baseline (dominant) and the PLITG bilexicon, and WA uses a weighted average of the baseline and the PLITG bilexicon.

based on lexicon size. In the cautious combination we use the features from the dominant lexicon whenever possible, and when there is an entry in the subordinate lexicon that is not present in the dominant, the features of that entry are down weighted. In the weighted average combination, all features are down weighted, but the feature scores are added up, giving an advantage to entries found in both lexica. The two approaches differ only in the weighting, so for every bilexicon entry l in the union of the two bilexica we have two sets of features. These are paired up as h_1 and h_2, and the combined weight is computed to be $h = \lambda_l h_1 + (1 - \lambda_l) h_2$, where λ_l is the weight for the entry defined as:

$$\lambda_l = \begin{cases} 1 & \text{if } l \in B_1 \\ \frac{|B_1|}{|B_1|+|B_2|} & \text{otherwise} \end{cases}$$

for the cautious combination approach (using B_1 to mean the dominant bilexicon and B_2 to mean the subordinate), and as:

$$\lambda_l = \frac{|B_1|}{|B_1| + |B_2|}$$

for the weighted average approach. A missing entry is considered to have feature scores zero across the board.

7 Results

The results are summarized in Table 2. As expected, the combinations seem to help. Although the cautious approach is equivalent to its dominant system (slightly higher NIST score and slightly lower BLEU score), the weighted average scores clearly higher than any of the other systems. It is also interesting to see the PLITG bilexicon do as well as it does, considering that it is less than half the size of the baseline lexicon. By combining the two bilexica, we can also compute the overlap, which is a mere 469,360 entries. This is a surprisingly low number, and the fact that they complement each other so well in terms of coverage might explain why the combination performs better than any of them in isolation.

8 Conclusions

We have presented a method for induction of probabilistic phrasal bilexica from parallel corpora. Rather than using a complex pipeline of models and heuristics, we have defined a theoretically principled stochastic transduction grammar that includes a natural probability distribution over phrasal bilexicon entries (preterminalized linear inversion transduction grammars, PLITGs), which was fitted to the observed parallel corpus. The resulting bilexicon performed competitively with the standard method when used in a phrase-based SMT system, despite being significantly smaller. The overlap between the two bilexica was minimal, which lead us to try a system using two different combinations, one of which outperformed any of the two isolated bilexica.

We have showed that the neither of the two types of bilexica is superior to the other, but that a combination outperforms them both. The combination methods presented here could, however, be improved. There rather than letting the size of the bilexicon determine its weight during combination, this could be controlled by a hyperparameter, whose value needs to be determined empirically.

Acknowledgments

This work was funded by the Defense Advanced Research Projects Agency (DARPA) under GALE Contract Nos. HR0011-06-C-0023 and HR0011-06-C-0023, and the Hong Kong Research Grants Council (RGC) under research grants GRF621008, GRF612806, DAG03/04.EG09, RGC6256/00E, and RGC6083/99E. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

References

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Halll, Englewood Cliffs, NJ.

Blunsom, Phil and Trevor Cohn. 2010. Inducing synchronous grammars with slice sampling. In *HLT/NAACL2010*, pages 238–241, Los Angeles, California, June. Association for Computational Linguistics.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Doddington, George. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *HLT-2002*, pages 138–145, San Diego, California.

Ginsburg, Seymour and Edwin H. Spanier. 1966. Finite-turn pushdown automata. *Society for Industrial and Applied Mathematics Journal on Control*, 4(3):429–453.

Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *ACL/IJCNLP2009*, pages 923–931, Suntec, Singapore, August. Association for Computational Linguistics.

Koehn, Philipp, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL2007, Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June.

Lewis, Philip M. and Richard E. Stearns. 1968. Syntax-directed transduction. *Journal of the Association for Computing Machinery*, 15(3):465–488.

Och, Franz Josef and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, October.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, July.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, July.

Rosenberg, Arnold L. 1967. A machine realization of the linear context-free languages. *Information and Control*, 10:175–188.

Saers, Markus, Joakim Nivre, and Dekai Wu. 2009. Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *IWPT'09*, pages 29–32, Paris, France, October.

Saers, Markus, Joakim Nivre, and Dekai Wu. 2010a. A systematic comparison between inversion transduction grammar and linear transduction grammar for word alignment. In *SSST4*, pages 10–18, Beijing, China, August. Coling 2010 Organizing Committee.

Saers, Markus, Joakim Nivre, and Dekai Wu. 2010b. Word alignment with stochastic bracketing linear inversion transduction grammar. In *HLT/NAACL2010*, pages 341–344, Los Angeles, California, June. Association for Computational Linguistics.

Saers, Markus. 2011. *Translation as Linear Transduction: Models and Algorithms for Efficient Learning in Statistical Machine Translation*. Ph.D. thesis, Uppsala University, Department of Linguistics and Philology.

Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, September.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, volume 1, pages 836–841, Copenhagen, Denmark, August.

Wu, Dekai. 1995a. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 244–251, Cambridge, Massachusetts, June.

Wu, Dekai. 1995b. Trainable coarse bilingual grammars for parallel text bracketing. In *Proceedings of the Third Annual Workshop on Very Large Corpora*, pages 69–81, Cambridge, Massachusetts, Jun.

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 152–158, Santa Cruz, California, June. Association for Computational Linguistics.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Zhang, Hao, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL-08: HLT*, pages 97–105, Columbus, Ohio, June. Association for Computational Linguistics.