# Speech Translation with Grammar Driven Probabilistic Phrasal Bilexica Extraction

*Markus Saers, Dekai Wu, Chi-kiu Lo, Karteek Addanki*

Human Language Technology Center
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{masaers|dekai|jackielo|vskaddanki}@cs.ust.hk

## Abstract

We introduce a new type of transduction grammar that allows for learning of probabilistic phrasal bilexica, leading to a significant improvement in spoken language translation accuracy. The current state-of-the-art in statistical machine translation relies on a complicated and crude pipeline to learn probabilistic phrasal bilexica—the very core of any speech translation system. In this paper, we present a more principled approach to learning probabilistic phrasal bilexica, based on stochastic transduction grammar learning applicable to speech corpora.

**Index Terms**: speech translation, transduction theory, lexicon extraction

## 1. Introduction

In this paper, we introduce a novel type of transduction grammar—the declarative form of a transducer—that improves translation on a speech translation task. The aim of the paper is to show that it is possible to replace the long and complicated pipeline that turns a parallel spoken language 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 speech translation systems of today.

When phrase-based speech translation 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 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 speech translation systems are capable of making generalizations beyond the scope of surface-based systems, it is also imperative to be able to handle chunks. 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 pairs of phrases that can be handled by generalization and the ones that cannot, is a hard problem. Naïvely enumerating all possible phrase pairs found in a parallel speech corpus and determine their probability by relative frequency, is doomed to fail because of the sheer amount of possible phrase pairs. The same is true for constructing transduction grammars with phrase pairs as terminals: the size of the grammar makes it impossible to handle. Indeed, transduction grammars (and thus transducers) are generally, but erroneously, perceived as being restricted to handle single-token terminals.

In this paper, we introduce a method for iteratively extending single token terminals to multi-token terminals. Rather than collecting all terminal pairs that could occur in the training corpus, we are collecting only those that could occur in a valid parse tree in the corpus (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 corpus using expectation-maximization (EM). When it has stabilized, we collect all adjacent emissions from the parse forest of the corpus, and add these larger emissions to the grammar. By repeating the process, larger and larger terminals can be incorporated into the grammar.

Since induction of transduction grammars is very time consuming, we opt to view the corpus as a *linear transduction* [1, 2]. 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 concept of pairs of terminals, 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 PLITG from a 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 speech translation system, allowing for easy comparison.

We will start with a review of the transduction grammar formalism of interest (Section 2), then focus on the particular stochastic phrasal formalism used in this paper (stochastic phrasal preterminalized linear inversion transduction grammars, Section 3). After the theoretical base has been established we will describe how we use these grammars to extract bilingual lexica (Section 4). To support our claims, we set up a series of experiments, described in Section 5, for which the results are given in Section 6. Finally, we offer some concluding remarks in Section 7.

## 2. Transduction Grammars

Transduction grammars are the grammar form of transducers, providing a more declarative view of the relation defined between the input and output languages of a transducer. We will refer to this pairing of two languages as a *transduction*. A transduction grammar is a grammar that can rewrite its start symbol to any and all string pairs for which the relation defined holds. As with languages, there is a hierarchy of transductions of different expressivity. In this paper we will focus on a class of transductions that lie strictly between finite-state transductions and inversion transductions [3], called linear transductions [2]. There has been two different grammar formalism described that both generate the class of linear transductions: linear inversion transduction grammars (LITGs) [1] and linear transduction grammars (LTGs) [2]. In this paper we introduce a third one termed preterminalized linear inversion transduction grammars (PLITGs). The advantage of this new type of grammars becomes clear when moving into the stochastic domain, as they allow all terminal productions to be included in a single probability distribution.

Inversion transduction grammars (ITGs) [3] model the simplest kind of relation between context-free languages that allow for reordering. Allowing for reordering is imperative for utterance level translation between natural languages.

**Definition 1.** *An* ITG *in normal form 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, $\Sigma$ is a finite nonempty set of $L_1$ symbols, $\Delta$ is a finite nonempty set of $L_2$ symbols, $S \in N$ is the start symbol, and $R$ is a finite, nonempty set of inversion transduction rules on the forms:*

$$A \to [BC], \qquad A \to \langle BC\rangle, \qquad A \to a/x$$

*where $A, B, C \in N$, $a \in \Sigma \cup \{\epsilon\}$ and $x \in \Delta \cup \{\epsilon\}$, with $\epsilon$ being the empty string.*

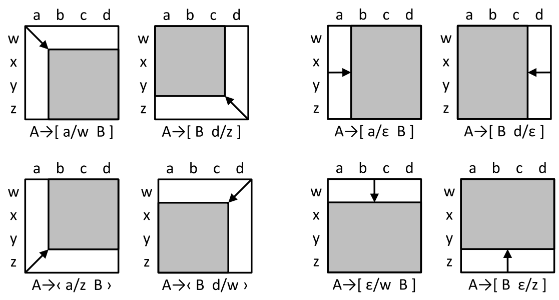
Productions enclosed in square brackets are read left-to-right in both languages, whereas productions enclosed in angled brackets are read left-to-right in $L_1$ and right-to-left in $L_2$. This simple reordering capacity is remarkably powerful, while still retaining the possibility of a two-normal form. The two-normal form is imperative for computational tractability—without it, parsing an utterance pair requires $\mathcal{O}(n^{2n+2})$ time, whereas the presence of the two-normal form allows for parsing in $\mathcal{O}(n^6)$ time. Although tractable, it is still not practical for large collections of long utterances.

An LITG is an ITG that has been subjected to a linearity constraint, so each rule may rewrite a nonterminal symbol to either a nonterminal symbol and a pair of terminal strings, or to a pair of empty strings ($\epsilon$). A pair of strings is denoted $a/x$ (where $a$ is taken from the input language and $x$ from the output language), and referred to as a *biterminal*. If either $a$ or $x$ is the empty string, the biterminal is called a *singleton*, and if both are the empty string, we call it *the empty biterminal*.

**Definition 2.** *An* LITG *over languages $L_1$ and $L_2$ is a tuple $G = \langle N, \Sigma, \Delta, S, R\rangle$ where $N$, $\Sigma$, $\Delta$ and $S$ are the same as for* ITG*s and $R$ is a set of linear inversion transduction rules on the forms:*

$$\begin{array}{lll}
A \to [a/x\ B], & A \to \langle a/x\ B\rangle, & A \to \epsilon/\epsilon, \\
A \to [B\ a/x], & A \to \langle B\ a/x\rangle, & \\
A \to [a/\epsilon\ B], & A \to \langle a/\epsilon\ B\rangle, & \\
A \to [\epsilon/x\ B], & A \to \langle B\ \epsilon/x\rangle, & \\
A \to [B\ a/\epsilon], & A \to \langle B\ a/\epsilon\rangle, & \\
A \to [B\ \epsilon/x], & A \to \langle \epsilon/x\ B\rangle &
\end{array}$$

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



(a) Without singletons (b) With singletons

Figure 1: The eight unique moves an LITG can make while producing terminal symbols. The singleton moves (b) can be made equivalently with inverting rules, instead of the displayed straight rules.

Note that the rules generating singletons are pair wise equivalent, meaning that, for every straight rule generating a singleton, there is an inverted rule generating the exact same string. Intuitively, this can be understood by noting that the position of the empty string is irrelevant—it does not matter whether it comes before or after the nonterminal. Figure 1 shows the different parsing moves that an LITG can perform, moving from one nonterminal $A$ covering a contiguous span in each sentence ($abcd$ and $wxyz$ respectively), to another nonterminal $B$ covering another span pair (gray box), by emitting some terminal symbols. When the nonterminal covers two empty spans, it can be rewritten to the empty biterminal only, eliminating the nonterminal and halting the process.

By lowering the generative capacity of the grammar in this way, we get an increase in efficiency, allowing LITGs to parse sentence pairs in $\mathcal{O}(n^4)$ time, which is further reducible to $\mathcal{O}(n)$ time by approximating the search [1].

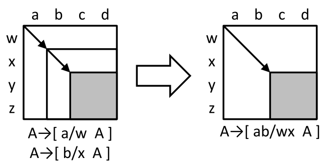
Also note that the biterminals are very much integrated into the rules, making it hard to construct a stochastic LITG that defines a probability distribution over the biterminals. In ITGs, this can be achieved by introducing a dedicated *preterminal* symbol into the set of nonterminals. The only job of the preterminal symbol is to rewrite into biterminal symbols. Since LITGs have no means of differentiating preterminals from nonterminals, introducing such a symbol would violate the linearity constraint, thereby effectively making it an ITG. To solve this, we separate the class of preterminals from the class of nonterminals, creating the class of preterminalized LITGs.

**Definition 3.** *A* PLITG *over languages $L_1$ and $L_2$ is a tuple $G = \langle N, P, \Sigma, \Delta, S, R\rangle$, where $N$, $\Sigma$, $\Delta$ and $S$ are the same as for* ITG*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:*

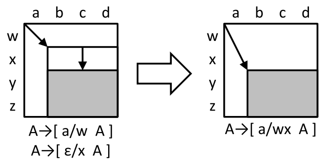
$$\begin{array}{lll}
A \to [BY], & A \to \langle BY\rangle, & A \to \epsilon/\epsilon, \\
A \to [YB], & A \to \langle YB\rangle, & X \to a/x
\end{array}$$

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

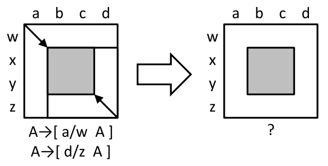
The class of PLITGs generates the same class of transductions as LTGs and LITGs, but the stochastic version (SPLITGs) allows for a single probability distribution to be defined over the biterminals, which we will exploit for bilexica construction.



(a) Valid extensions



(b) Valid extension with singleton



(c) Invalid extensions

Figure 2: Examples of valid and invalid extensions for an LITG.

## 3. Phrasal SPLITGs

Learning stochastic phrasal transduction grammars from corpora is a daunting task. Machine learning algorithms such as expectation maximization (EM) [4], which we will use in this paper, are restricted to setting the weights of the parameters of the model. This means that the parameters have to be enumerated before training begins. For a stochastic phrasal transduction grammar, every span in $L_1$ combined with every span in $L_2$ is a parameter. By only considering the ones that are observed in the training corpus, the number can be significantly reduced, but still remains intractably large.

Our solution to the problem is to start with a token based SPLITG, and iteratively add larger spans that are observed, not only in the training corpus, but also in the parse forest of the training corpus according to the grammar we have learned so far. We thus start by learning a token based grammar from the training corpus, which we stabilize with a few iterations of EM training. We then inspect the parse forest, and single out all sequences of two rules where the same results could have been achieved with one valid SPLITG rule. These hypothetical rules are then simply incorporated into the grammar. The new grammar with larger translation units is then stabilized, at which point we can either repeat the process to get even larger biterminals, or be satisfied with our grammar. Each round of enlargements doubles the maximum length of the terminal symbols in the grammar. By extending the terminal productions to produce multiple symbols rather than single symbols, the grammar becomes phrasal, making it a phrasal SPLITG.

The extension process is illustrated in Figure 2. Although the illustration includes LITG rules rather than PLITG rules, these can be deterministically recovered from the LITG rules.

## 4. Probabilistic phrasal bilexicon extraction using phrasal SPLITGs

A probabilistic bilexicon is a translation lexicon where the entries are associated with probabilities, and a phrasal bilexicon contains multi-token entries with multi-token translations. A probabilistic phrasal bilexicon is an integral part of any speech translation system, under the name *phrase table*. These phrase tables are extracted as all possible $L_1$ spans combined with all possible $L_2$ spans from a corpus that are consistent with a word alignment over that corpus. This process thus requires the training corpus to be word aligned.

The constraint that the phrases need to be consistent with a word alignment is similar to our constraint that phrasal biterminals need to be observed in the parse forest of the corpus. Since each parse tree in the parse forest assigns an alignment between the two sentences being parsed, the phrasal biterminals learned by observing rule sequences in the parse forest, encode a fragment of this alignment information. Rather than being constrained to exactly one word alignment per sentence pair, as the standard approach is, our approach integrates over all the alignments of the entire parse forest.

There is in other words a similarity between the biterminals in a phrasal SPLITG and the phrases in a phrase table. One difference is that the phrase pairs in a phrase table are scored with five different scores: $\phi(f|e)$, $\phi(e|f)$, $lex(f|e)$, $lex(e|f)$ and a constant. The letters $e$ and $f$ are used to denote the output language phrase and input language phrase respectively, $\phi$ represents the translation probability: how often was $f$ translated into $e$, and $lex$ represents the lexical probability: how well do the tokens in $f$ correspond to the tokens in $e$. The constant is there to allow the decoder to add a cost every time a phrase entry is used. This makes it prefer larger chunks, which are more likely to be correct, since they were actually observed in the training corpus.

To build a phrase table from a SPLITG, we start with the lexical rules we learned, and assume that they constitute the entries. To score the entries, we use the rule probability as a basis for the $\phi$ score, marginalizing the input and output phrase to compute the two conditional scores. To mimic the $lex$ score, we parse the entry as if it was a sentence pair. This gives us a score of how likely the string pair is given the grammar, which intuitively corresponds well to the concept of a lexical score. Again we marginalize over the two phrases to make two conditional scores rather than one joint.

This represents a much more principled approach to probabilistic phrasal bilexicon extraction that incorporates word alignment and phrase extraction into a joint process rather than a pipeline of distinct steps.

## 5. Experimental setup

To test the quality of the extracted probabilistic phrasal bilexicon, we will use a speech translation task. We selected the IWSLT07 Chinese–English translation task [5], and a standard phrase-based statistical machine translation system [6] as our baseline. To build the baseline we used the freely available Moses toolkit [7], with standard training settings, and an SRI trigram language model [8]. To compare systems we use BLEU [9].

This baseline represents the state-of-the art. Our system will be based on a phrase table extracted from a phrasal SPLITG that was learned from the same data as the baseline was trained on. To further investigate the differences between the two approaches we will create several hybrid phrase tables, where features computed for the SPLITG phrase table are added to the baseline phrase table. The reason for augmenting the baseline system rather than our own experimental system, is to minimize the model mismatch between our training regime and the Moses

| System | BLEU |
|---|---|
| Moses (baseline) | 27.36 |
| SPLITG | 21.08 |
| Moses + SPLITG indicator feature | 27.08 |
| Moses + SPLITG conditional scores | 28.03 |
| Moses + SPLITG joint scores | **28.20** |

Table 1: Results from the experiments.

decoder.

To learn the phrasal SPLITG we will use the approximate parsing described in [1] with a beam size of $b = 25$, which gives a good trade-off between efficiency and search error avoidance. We will do five iterations of EM training between the extension rounds, and a total of three extension rounds. We thus run a total of 20 iterations of EM. Three extension rounds means that the longest possible phrase length is $2^3 = 8$ tokens, which is comparable to the baseline system.

To combine the phrase tables, we will start out with the baseline phrase table, and add features from the SPLITG phrase table for phrase pairs that exist in both tables. For phrase pairs that exist only in the baseline table, we will consider the added features to have a value of zero. For phrase pairs that exist only in the SPLITG table, we do nothing.

We will compute three different combined phrase tables: one with all the computed features of the SPLITG table, one with the joint rather than conditional scores from the SPLITG table and one with an indicator feature that takes the value one if the phrase pair occurs in the SPLITG table, and zero if it does not.

## 6. Results

After running the experiment outlined in Section 5, we have five different phrase tables that we can plug into our speech translation system and test. The results of this is found in Table 1. The pure SPLITG based system performs comparatively poorly, which we would like to ascribe to the mismatch between the training and decoding models—the grammar was not constructed to be used in a surface-based speech translation system in the first place, so trying to shoe horn it into such a system is suboptimal.

Whether a phrase pair is present in the SPLITG phrase table or not (*indicator feature* in Table 1) seems to contain too little information. It is reasonable to believe that the low probability phrase pairs in the SPLITG phrase table represent a sparse collection of the actual low probability phrase pairs, the membership in the table is thus highly arbitrary for low probability phrase pairs. The fact that adding this piece of information degrades performance indicates that the feature does indeed confuse rather than aid the speech translation system.

When taking the probabilities that were learned during grammar induction—rather than just the presence of the phrase pairs discovered—into account, we see a consistent improvement. There is, in other words, some information that the SPLITG system was able to pick up which the baseline system did not. The fact that the system with fewer features (*joint scores* in Table 1) outperformed the one with more features (*conditional scores* in Table 1) seems counter intuitive at first. We believe that there are two factors that explain this: the fact that the conditional scores are synthesized from the joint scores, and the parameter tuning employed (minimum error rate training, MERT). The first explanation, that the conditional scores are synthesized rather than observed, implies that the conditional model has exactly the same information as the joint model, just phrased differently. Taking this line of reasoning to its logical conclusion, we should expect the same results with both models, which brings us to the second explanation: MERT. In general, MERT is prone to getting stuck in local optima, which is combated with random restarts. This effect becomes more severe the more features MERT has to tune, which could explain why the more feature rich model performs worse. The suboptimal tuning could have been worth it if it would allow us to use some discriminating features. In fact, adding four extra features to the baseline system does increase performance, just not as much as adding two features with the same information content.

## 7. Conclusions

In summary, we have presented a novel, grammar-based method for extracting probabilistic phrasal bilingual lexica, and successfully shown that the learned lexica contain information that the state-of-the-art baseline speech translation system failed to uncover, giving a significant boost in speech translation quality.

## 8. References

[1] M. Saers, J. Nivre, and D. Wu, "Word alignment with stochastic bracketing linear inversion transduction grammar," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California, June 2010, pp. 341–344.

[2] M. Saers, "Translation as linear transduction: Models and algorithms for efficient learning in statistical machine translation," Ph.D. dissertation, Uppsala University, Department of Linguistics and Philology, 2011.

[3] D. Wu, "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora," *Computational Linguistics*, vol. 23, no. 3, pp. 377–403, 1997.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[5] C. Fordyce, "Overview of the IWSLT 2007 evaluation campaign," in *Proceedings of the International Workshop on Spoken Language Translation*, 2007, pp. 1–12.

[6] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, Edmonton, Canada, May/June 2003.

[7] P. Koehn, 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, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic, June 2007, pp. 177–180.

[8] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado, September 2002, pp. 901–904.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, July 2002, pp. 311–318.