Investigations in Exact Inference for Hierarchical Translation

Wilker Aziz[†], **Marc Dymetman**[‡], **Sriram Venkatapathy**[‡]
[†]University of Wolverhampton, Wolverhampton, UK
[‡]Xerox Research Centre Europe, Grenoble, France
[†]w.aziz@wlv.ac.uk, [‡]{*first.last*}@xrce.xerox.com

Abstract

We present a method for inference in hierarchical phrase-based translation, where both *optimisation* and *sampling* are performed in a common exact inference framework related to adaptive rejection sampling. We also present a first implementation of that method along with experimental results shedding light on some fundamental issues. In hierarchical translation, inference needs to be performed over a high-complexity distribution defined by the intersection of a translation hypergraph and a target language model. We replace this intractable distribution by a sequence of tractable upper-bounds for which exact optimisers and samplers are easy to obtain. Our experiments show that exact inference is then feasible using only a fraction of the time and space that would be required by the full intersection, without recourse to pruning techniques that only provide approximate solutions. While the current implementation is limited in the size of inputs it can handle in reasonable time, our experiments provide insights towards obtaining future speedups, while staying in the same general framework.

1 Introduction

In statistical machine translation (SMT), *optimisation* — the task of searching for an optimum translation — is performed over a high-complexity distribution defined by the intersection between a translation hypergraph and a target language model (LM). This distribution is too complex to be represented exactly and one typically resorts to approximation techniques such as beam-search (Koehn et al., 2003) and cube-pruning (Chiang, 2007), where maximisation is performed over a pruned representation of the full distribution.

Often, rather than finding a single optimum, one is really interested in obtaining a set of probabilistic *samples* from the distribution. This is the case for minimum error rate training (Och, 2003; Watanabe et al., 2007), minimum risk training (Smith and Eisner, 2006) and minimum risk decoding (Kumar and Byrne, 2004). Due to the additional computational challenges posed by sampling, n-best lists, a by-product of optimisation, are typically used as approximation to true probabilistic samples. A known issue with n-best lists is that they tend to be clustered around only one mode of the distribution. A more direct procedure is to attempt to directly draw samples from the underlying distribution rather than rely on n-best list approximations (Arun et al., 2009; Blunsom and Osborne, 2008).

OS* (Dymetman et al., 2012a) is a recent approach that stresses a unified view between the two types of inference, optimisation and sampling. In this view, rather than resorting to pruning in order to cope with the tractability issues, one upper-bounds the complex goal distribution with a simpler "proposal" distribution for which dynamic programming is feasible. This proposal is incrementally refined to be closer to the goal until the maximum is found, or until the sampling performance exceeds a certain level.

This paper applies the OS* approach to the problem of inference in hierarchical SMT (Chiang, 2007). In a nutshell, the idea is to replace the intractable problem of intersecting a context-free grammar with a full language model by the tractable problem of intersecting it with a simplified, optimistic version of this LM which "forgets" parts of n-gram contexts, and to incrementally add more context based on evidence of the need to do so. Evidence is gathered by optimising or sampling from the tractable proxy distribution and focussing on the most serious over-optimistic estimates relative to the goal distribution.

Our main contribution is to provide an exact optimiser/sampler for hierarchical SMT that is efficient in exploring only a small fraction of the space of n-grams involved in a full intersection. Although at this stage our experiments are limited to short sentences, they provide insights on the behavior of the technique and indicate directions towards a more efficient implementation within the same paradigm.

The paper is organized as follows: §2 provides background on OS* and hierarchical translation; §3 describes our approach to exact inference in SMT; in §4 the experimental setup is presented and findings are discussed; §5 discusses related work, and §6 concludes.

2 Background

2.1 OS*

The OS* approach (Dymetman et al., 2012a; Dymetman et al., 2012b) proposes a unified view of exact inference in sampling and optimisation, where the two modalities are seen as extremes in a continuum of inference tasks in L^p spaces (Rudin, 1987), with sampling associated with the L_1 norm, and optimisation with the L^∞ norm.

The objective function p, over which inference needs to be performed, is a complex non-negative function over a discrete or continuous space X, which defines an unnormalised distribution over X. The goal is to optimise or sample relative to p — where sampling is interpreted in terms of the normalised distribution $\bar{p}(.) = p(.)/\int_X p(x)dx$.

Directly optimising or sampling from p is unfeasible; however, it is possible to define an (unnormalized) distribution q of lower complexity than p, which upper-bounds p everywhere (ie. $p(x) \leq q(x), \forall x \in X$), and from which it is feasible to optimise or sample directly.

Sampling is performed through *rejection sampling*: first a sample x is drawn from q, and then x is accepted or rejected with probability given by the ratio $r = p(x)/q(x)$, which is less than 1 by construction. Accepted x's can be shown to produce an exact sample from p (Robert and Casella, 2004). When the sample x from q is rejected, it is used as a basis for "refining" q into a slightly more complex q', where $p \leq q' \leq q$ is still an upper-bound to p. This "adaptive rejection sampling" technique incrementally improves the rate of acceptance, and is pursued until some rate above a given threshold is obtained, at which point one stops refining and uses the current proposal to obtain further exact samples from p.

In the case of **optimisation**, one finds the maximum x relative to q, and again computes the ratio $r = p(x)/q(x)$. If this ratio equals 1, then it is easy to show that x is the actual maximum from p.[1] Otherwise we refine the proposal in a similar way to the sampling case, continuing until we find a ratio equal to 1 (or very close to 1 if we are willing to accept an approximation to the maximum). For finite spaces X, this optimisation technique is argued to be a generalisation of A^*.

An application of the OS^* technique to sampling/optimisation with High-Order HMM's is described in Carter et al. (2012) and provides background for this paper. In that work, while the high-order HMM corresponds to an intractable goal distribution, it can be upper-bounded by a sequence of tractable distributions for which optimisers and samplers can be obtained through standard dynamic programming techniques.

2.2 Hierarchical Translation

An abstract formulation of the decoding process for hierarchical translation models such as that of Chiang (2007) can be expressed as a sequence of three steps. In a first step, a translation model $\mathcal{G}$, represented as a weighted synchronous context-free grammar (SCFG) (Chiang, 2005), is applied to (in other words, intersected with) the source sentence f to produce a weighted context-free grammar $G(f)$ over the target language.[2] In a second step, $G(f)$ is intersected with a weighted finite-state automaton A representing the target language model, resulting in a weighted context-free grammar $G'(f) = G(f) \cap A$. In a final step, a dynamic programming procedure (see §2.4) is applied to find the maximum derivation x in $G'(f)$, and the sequence of leaves of yield(x) is the result translation.

While this formulation gives the general principle, already mentioned in Chiang (2007), most implementations do not exactly follow these steps or use this terminology. In practice, the closest approach to this abstract formulation is that of Dyer (2010) and the related system cdec (Dyer et al., 2010); we follow a similar approach here.

[1]This is because if x' was such that $p(x') > p(x)$, then $q(x') \geq p(x') > p(x) = q(x)$, and hence x would not be a maximum for q, a contradiction.

[2]$G(f)$ is thus a compact representation of a forest over target sequences, and is equivalent to a *hypergraph*, using different terminology.

Whatever the actual implementation chosen, all approaches face a common problem: the complexity of the intersection $G'(f) = G(f) \cap A$ increases rapidly with the order of the language model, and can become unwieldy for moderate-length input sentences even with a bigram model. In order to address this problem, most implementations employ variants of a technique called *cube-pruning* (Chiang, 2007; Huang and Chiang, 2007), where the cells constructed during the intersection process retain only a k-best list of promising candidates. This is an approximation technique, related to beam-search, which performs well in practice, but is not guaranteed to find the actual optimum.

In the approach presented here — described in detail in §3 — we do not prune the search space. While we do construct the full initial grammar $G(f)$, we proceed by incrementally intersecting it with simple automata associated with upper-bounds of A, for which the intersection is tractable.

2.3 Earley Intersection

In their classical paper Bar-Hillel et al. (1961) showed that the intersection of a CFG with a FSA is a CFG, and Billot and Lang (1989) were possibly the first to notice the connection of this construct with chart-parsing. In general, parsing with a CFG can be seen as a special case of intersection, with the input sequence represented as a "flat" (linear chain) automaton, and this insight allows to generalise various parsing algorithms to corresponding intersection algorithms. One such algorithm, for weighted context-free grammars and automata, inspired by the CKY parsing algorithm, is presented in Nederhof and Satta (2008). The algorithm that we are using is different; it is inspired by Earley parsing, and was introduced in chapter 2 of Dyer (2010). The advantage of Dyer's "Earley Intersection" algorithm is that it combines top-down predictions with bottom-up completions. The algorithm thus avoids constructing many non-terminals that may be justified from the bottom-up perspective, but can never be "requested" by a top-down derivation, and would need to be pruned in a second pass. Our early experiments showed an important gain in intermediary storage and in overall time by using this Earley-based technique as opposed to a CKY-based technique.

We do not describe the Earley Intersection algorithm in detail here, but refer to Dyer (2010), which we follow closely.

2.4 Optimisation and Sampling from a WCFG

Optimisation in a weighted CFG (WCFG)[3], that is, finding the maximum derivation, is well studied and involves a dynamic programming procedure that assigns in turn to each nonterminal, according to a bottom-up traversal regime, a maximum derivation along with its weight, up to the point where a maximum derivation is found for the initial nonterminal in the grammar. This can be seen as working in the max-times semiring, where the weight of a derivation is obtained through the product of the weights of its sub-derivations, and where the weight associated with a nonterminal is obtained by maximising over the different derivations rooted in that nonterminal.

The case of sampling can be handled in a very similar way, by working in the sum-times instead of the max-times semiring. Here, instead of maximising over the weights of the competing derivations rooted in the same nonterminal, one sums over these weights. By proceeding in the same bottom-up way, one ends with an accumulation of all the weights on the initial nonterminal (this can also be seen as the *partition function* associated with the grammar). An efficient exact sampler is then obtained by starting at the root nonterminal, randomly selecting an expansion proportionally to the weight of this expansion, and iterating in a top-down way. This process is described in more detail in section 4 of Johnson et al. (2007), for instance.

3 Approach

The complexity of building the full intersection $G(f) \cap A$, when A represents a language model of order n, is related to the fact that the number of states of A grows exponentially with n, and that each nonterminal N in $G(f)$ tends to generate in the grammar $G'(f)$ many indexed nonterminals of the form (i, N, j), where i, j are states of A and the nonterminal (i, N, j) can be interpreted as an N connecting an i state to a j state.

In our approach, instead of explicitly constructing the full intersection $G(f) \cap A$, which, using the notation of §2.1, is identified with the unnormalised goal distribution $p(x)$, we incrementally produce a sequence of "proposal" grammars $q^{(t)}$, which all upper-bound p, where $q^{(0)} = G(f) \cap A^{(0)}$, ..., $q^{(t+1)} = q^{(t)} \cap A^{(t)}$, etc. Here $A^{(0)}$ is

[3]Here the CFG is assumed to be acyclic, which is typically the case in translation applications.

an optimistic, low complexity, "unigram" version of the automaton A, and each increment $A^{(t)}$ is a small automaton that refines $q^{(t)}$ relative to some specific k-gram context (i.e., sequence of k words) not yet made explicit in the previous increments, where k takes some value between 1 and n. This process produces a sequence of grammars $q^{(t)}$ such that $q^{(0)}(.) \geq q^{(1)}(.) \geq q^{(2)}(.) \geq ... \geq p(.)$.

In the limit $\bigcap_{t=0}^{M} A^{(t)} = A$ for some large M, so that we are in principle able to reconstruct the full intersection $p(.) = q^{(M)} = G(f) \cap A^{(0)} \cap ... \cap A^{(M)}$ in finite time. In practice our actual process stops much earlier: in optimisation, when the value of the maximum derivation x_t^* relative to $q^{(t)}$ becomes equal to its value according to the full language model, in sampling when the acceptance rate of samples from $q^{(t)}$ exceeds a certain threshold. The process is detailed in what follows.

3.1 OS* for Hierarchical Translation

Our application of OS* to hierarchical translation is illustrated in Algorithm 1, with the two modes, optimisation and sampling, made explicit and shown side-by-side to stress the parallelism.

On line 1, we initialise the time step to 0, and for sampling we also initialise the current acceptance rate (AR) to 0. On line 2, we initialise the initial proposal grammar $q^{(0)}$, where $A^{(0)}$ is detailed in §3.2. On line 3, we start a loop: in optimisation we stop when we have found an x that is accepted, meaning that the maximum has been found; in sampling, we stop when the estimated acceptance rate (AR) of the current proposal $q^{(t)}$ exceeds a certain threshold (e.g. 20%) — this AR can be roughly estimated by observing how many of the last (say) one hundred samples from the proposal have been accepted, and tends to reflect the actual acceptance rate obtained by using $q^{(t)}$ without further refinements. On line 4, in optimisation, we compute the argmax x from the proposal, and in sampling we draw a sample x from the proposal.[4] On line 5, we compute the ratio $r = p(x)/q^{(t)}(x)$; by construction $q^{(t)}$ is an optimistic version of p, thus $r \leq 1$.

On line 6, in optimisation we accept x if the ratio is equal to 1, in which case we have found the maximum, and in sampling we accept x with probability r, which is a form of *adaptive rejection sampling* and guarantees that accepted samples form exact samples from p; see (Dymetman et al., 2012a).

If x was rejected (line 7), we then (lines 8, 9) refine $q^{(t)}$ into a $q^{(t+1)}$ such that $p(.) \leq q^{(t+1)}(.) \leq q^{(t)}(.)$ everywhere. This is done by defining the incremental automaton $A^{(t+1)}$ on the basis of x and $q^{(t)}$, as will be detailed below, and by intersecting this automaton with $q^{(t)}$

Finally, on line 11, in optimisation we return the x which has been accepted, namely the maximum of p, and in sampling we return the list of already accepted x's, which form an exact sample from p, along with the current $q^{(t)}$, which can be used as a sampler to produce further exact samples with an acceptance rate performance above the predefined threshold.

3.2 Incremental refinements

Initial automaton $\boldsymbol{A^{(0)}}$ This deterministic automaton is an "optimistic" version of A which only records unigram information. $A^{(0)}$ has only one state q_0, which is both initial and final. For each word a of the target language it has a transition (q_0, a, q_0) whose weight is denoted by $w_1(a)$. This weight is called the "max-backoff unigram weight" (Carter et al., 2012) and it is defined as:

$$w_1(a) \equiv \max_h p_{lm}(a|h),$$

where $p_{lm}(a|h)$ is the conditional language model probability of a relative to the history h, and where the maximum is taken over all possible histories, that is, over all possible sequence of target words that might precede a.

Max-backoffs Following Carter et al. (2012), for any language model of finite order, the unigram max-backoff weights $w_1(a)$ can be precomputed in a "Max-ARPA" table, an extension of the ARPA format (Jurafsky and Martin, 2000) for the target language model, which can be precomputed on the basis of the standard ARPA table.

From the Max-ARPA table one can also directly compute the following "max-backoff weights": $w_2(a|a_{-1})$, $w_3(a|a_{-2}\, a_{-1})$, ..., which are defined by:

$$\begin{aligned} w_2(a|a_{-1}) &\equiv \max_h p_{lm}(a|h, a_{-1}) \\ w_3(a|a_{-2}\, a_{-1}) &\equiv \max_h p_{lm}(a|h, a_{-2}\, a_{-1}) \\ &\;\;\ldots \end{aligned}$$

where the maximum is taken over the part of the history which is not explicitely indicated.

[4] Following the OS* approach, taking an argmax is actually assimilated to an extreme form of sampling, with an L_∞ space taking the place of an L_1 space.

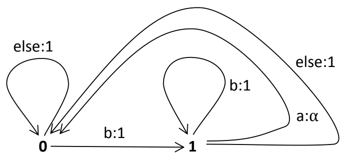
Algorithm 1 OS* for Hierarchical Translation: Optimisation (left) and Sampling (right).

```
1: $t \leftarrow 0$
2: $q^{(0)} \leftarrow G(f) \cap A^{(0)}$
3: while not an $x$ has been accepted do
4:     Find maximum $x$ in $q^{(t)}$
5:     $r \leftarrow p(x)/q^{(t)}(x)$
6:     Accept-or-Reject $x$ according to $r$
7:     if Rejected($x$) then
8:         define $A^{(t+1)}$ based on $x$ and $q^{(t)}$
9:         $q^{(t+1)} \leftarrow q^{(t)} \cap A^{(t+1)}$
10:        $t \leftarrow t+1$
11: return $x$
```

```
1: $t \leftarrow 0, AR \leftarrow 0$
2: $q^{(0)} \leftarrow G(f) \cap A^{(0)}$
3: while not $AR > threshold$ do
4:     Sample $x \sim q^{(t)}$
5:     $r \leftarrow p(x)/q^{(t)}(x)$
6:     Accept-or-Reject $x$ according to $r$
7:     if Rejected($x$) then
8:         define $A^{(t+1)}$ based on $x$ and $q^{(t)}$
9:         $q^{(t+1)} \leftarrow q^{(t)} \cap A^{(t+1)}$
10:        $t \leftarrow t+1$
11: return already accepted $x$'s along with $q^{(t)}$
```

Note that: (i) if the underlying language model is, say, a trigram model, then $w_3(a|a_{-2}\,a_{-1})$ is simply $p_{lm}(a|a_{-2}\,a_{-1})$, and similarly for an underlying model of order k in general, and (ii) $w_2(a|a_{-1}) = \max_{a_{-2}} w_3(a|a_{-2}\,a_{-1})$ and $w_1(a) = \max_{a_{-1}} w_2(a|a_{-1})$.

Incremental automata $A^{(t)}$ The weight assigned to any target sentence by $A^{(0)}$ is larger or equal to its weight according to A. Therefore, the initial grammar $q^{(0)} = G(f) \cap A^{(0)}$ is *optimistic* relative to the actual grammar $p = G(f) \cap A$: for any derivation x in p, we have $p(x) \leq q^{(0)}(x)$. We can then apply the OS* technique with $q^{(0)}$. In the case of **optimisation**, this means that we find the maximum derivation x from $q^{(0)}$. By construction, with y = yield(x), we have $A^{(0)}(y) \geq A(y)$. If the two values are equal, we have found the maximum,[5] otherwise there must be a word y_i in the sequence $y_1^m = y$ for which $p_{lm}(y_i|y_1^{i-1})$ is strictly smaller than $w_1(y_i)$. Let us take among such words the one for which the ratio $\alpha = w_2(y_i|y_{i-1})/w_1(y_i) \leq 1$ is the smallest, and for convenience let us rename $b = y_{i-1}, a = y_i$. We then define the (deterministic) automaton $A^{(1)}$ as illustrated in the following figure:



Here the state 0 is both initial and final, and the state 1 is final; all edges carry a (multiplicative) weight equal to 1, except edge $(1, a, 0)$, which carries the weight α. We use the abbreviation "else" to refer to any label other than b when starting from 0, and other than b or a when starting from 1.

It is easy to check that this automaton assigns to any word sequence y a weight equal to α^k, where k is the number of occurrences of $b\,a$ in y. In particular, if y is such that $y_{i-1} = b, y_i = a$, then the transition in (the deterministic automaton) $A^{(0)} \cap A^{(1)}$ that consumes y_i carries the weight $\alpha\ w_1(a)$, in other words, the weight $w_2(a|b)$. Thus the new proposal grammar $q^{(1)} = q^{(0)} \cap A^{(1)}$ has now "incorporated" knowledge of the bigram a-in-the-context-b, at the cost of some increase in its complexity.[6]

The general procedure for choosing $A^{(t+1)}$ follows the same pattern. We find the max derivation x in $q^{(t)}$ along with its yield y; if $p(x) = q^{(t)}(x)$, we stop and output x; otherwise we find some subsequence $y_{i-m-1}, y_{i-m}, ..., y_i$ such that the knowledge of the n-gram $y_{i-m}, ..., y_i$ has already been registered in $q^{(t)}$, but not that of the n-gram $y_{i-m-1}, y_{i-m}, ..., y_i$, and we define an automaton $A^{(t+1)}$ which assign to a sequence a weight α^k, where

$$\alpha = \frac{w_{m+1}(y_i|y_{i-m-1}, y_{i-m}, ..., y_{i-1})}{w_m(y_i|y_{i-m}, ..., y_{i-1})},$$

and where k is the number of occurrences of $y_{i-m-1}, y_{i-m}, ..., y_i$ in the sequence.[7]

We note that we have $p \leq q^{(t+1)} \leq q^{(t)}$ everywhere, and also that the number of possible refinement operations is bounded, because at some point we would have expanded all contexts to their maximum order, at which point we would have reproduced $p(.)$ on the whole space X of possible

[5]This case is very unlikely with $A^{(0)}$, but helps introduce the general case.

[6]Note that without further increasing $q^{(1)}$'s complexity one can incorporate knowledge about all bigrams sharing the prefix b. This is because $A^{(1)}$ does not need additional states to account for different continuations of the context b, all we need is to update the weights of the transitions leaving state 1 appropriately. More generally, it is not more costly to account for all n-grams prefixed by the same context of $n-1$ words than it is to account for only one of them.

[7]Building $A^{(t+1)}$ is a variant of the standard construction for a "substring-searching" automaton (Cormen et al., 2001) and produces an automaton with n states (the order of the n-gram). This construction is omitted for the sake of space.

derivations exactly. However, we typically stop much earlier than that, without expanding contexts in the regions of X which are not promising even on optimistic assessments based on limited contexts.

Following the OS* methodology, the situation with **sampling** is completely parallel to that of optimisation, the only difference being that, instead of finding the maximum derivation x from $q^{(t)}(.)$, we draw a sample x from the distribution associated with $q^{(t)}(.)$, then accept it with probability given by the ratio $r = p(x)/q^{(t)}(x) \leq 1$. In the case of a reject, we identify a subsequence $y_{i-m-1}, y_{i-m}, ..., y_i$ in yield(x) as in the optimisation case, and similarly refine $q^{(t)}$ into $q^{(t+1)} = q^{(t)} \cap A^{(t+1)}$. The acceptance rate gradually increases because $q^{(t)}$ comes closer and closer to p. We stop the process at a point where the current acceptance rate, estimated on the basis of, say, the last one hundred trials, exceeds a predefined threshold, perhaps 20%.

3.3 Illustration

In this section, we present a small running example of our approach. Consider the lowercased German source sentence: *eine letzte beobachtung .*

Table 1 shows the translation associated with the optimum derivation from each proposal $q^{(i)}$. The n-gram whose cost, if extended by one word to the left, would be increased by the largest factor is underlined. The extended context selected for refinement is highlighted in bold.

i	Rules	Optimum
0	311	**<s>** $\underline{\text{one}}$ last observation . </s>
1	454	<s> **one** $\underline{\text{last}}$ observation . </s>
2	628	<s> one **last** $\underline{\text{observation}}$. </s>
3	839	<s> one final **observation** $\underline{\text{.}}$ </s>
4	1212	<s> one **final** $\underline{\textbf{observation .}}$ </s>
		...
12	3000	<s> **a** $\underline{\textbf{final}\text{ observation}}$. </s>
13	3128	<s> one final observation . </s>

Table 1: Optimisation steps showing the iteration (i), the number of rules in the grammar and the translation associated to the *optimum* derivation.

Consider the very first iteration ($i = 0$), at which point only unigram costs have been incorporated. The sequence *<s> one last observation . </s>* represents the translation associated to the best derivation x in $q^{(0)}$. We proceed by choosing from it one sequence to be the base for a refinement that will lower $q^{(0)}$ bringing it closer to p. Amongst all possible one-word (to the left) extensions, extending the unigram 'one' to the bigram '<s> one' is the operation that lowers $q^{(0)}(x)$ the most. It might be helpful to understand it as the bigram '<s> one' being associated to the largest LM gap observed in x. Therefore the context '<s>' is selected for refinement, which means that an automaton $A^{(1)}$ is designed to down-weight derivations compatible with bigrams prefixed by '<s>'. The proposal $q^{(0)}$ is intersected with $A^{(1)}$ producing $q^{(1)}$. We proceed like this iteratively, always selecting a context not yet accounted for until $q^{(i)}(x) = p(x)$ for the best derivation (13th iteration in our example), when the true optimum is found with a certificate of optimality.

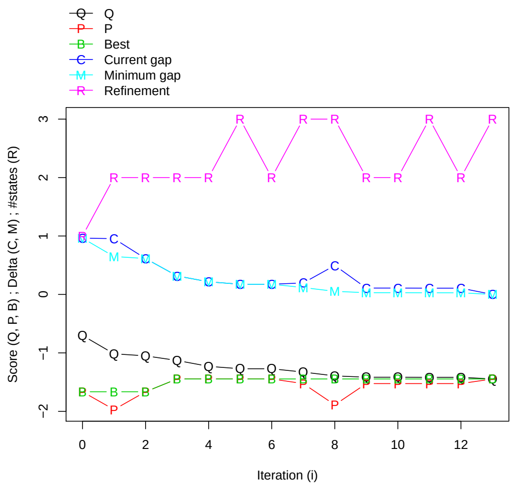


Figure 1: Certificate of optimality.

Figure 1 displays the progression of Q (score of the best derivation) and P (that derivation's true score). As guaranteed by construction, Q is always above P. B represents the score of the best derivation so far according to the true scoring function, that is, B is a lower-bound on the true optimum[8]. The optimal solution is achieved when $P = Q$.

Curve B in Figure 1 shows that the best scoring solution was found quite early in the search ($i = 3$). However, optimality could only be proven 10 iterations later. Another way of stating the convergence criterion $Q = P$ is observing a zero gap (in the *log* domain) between Q and P (see curve C – current gap), or a zero gap between Q and B (see curve M – minimum gap). Observe how M drops quickly from 1 to nearly 0, followed by a long tail where M

[8]This observation allows for error-safe pruning in *optimisation*: if x is a lower-bound on the true optimum, derivations in $q^{(i)}$ that score lower than $p(x)$ could be safely removed. We have left that possibility for future work.

decreases much slower. Note that if we were willing to accept an approximate solution, we could already stop the search if B remained unchanged for a predetermined number of iterations or if changes in B were smaller than some threshold, at the cost of giving up on the optimality certificate.

Finally, curve R shows the number of states in the automaton $A^{(i)}$ that refines the proposal at iteration i. Note how lower order n-grams (2-grams in fact) are responsible for the largest drop in the first iterations and higher-order n-grams (in fact 3-grams) are refined later in the long tail.

Figure 2 illustrates the progression of the sampler for the same German sentence. At each iteration a batch of 500 samples is drawn from $q^{(i)}$. The rejected samples in the batch are used to collect statistics about overoptimistic n-grams and to heuristically choose one context to be refined for the next iteration, similar to the optimisation mode. We start with a low acceptance rate which grows up to 30% after 15 different contexts were incorporated. Note how the L_1 norm of q (its partition function) decreases after each refinement, that is, q is gradually brought closer to p, resulting in the increased number of exact samples and better acceptance rate.

Note that, starting from iteration one, all refinements here correspond to 2-grams (i.e. one-word contexts). This can be explained by the fact that, in sampling, lower-order refinements are those that mostly increase acceptance rate (rationale: high-order n-grams are compatible with fewer grammar rules).

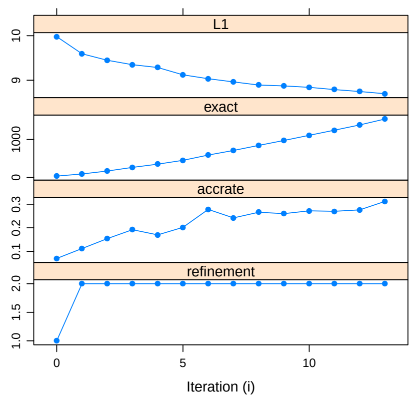


Figure 2: L_1 norm of q, the number of exact samples drawn, the acceptance rate and the refinement type at each iteration.

4 Experiments

We used the Moses toolkit (Koehn et al., 2007) to extract a SCFG following Chiang (2005) from the 6th version of the Europarl collection (Koehn, 2005) (German-English portion). We trained language models using lmplz (Heafield et al., 2013) and interpolated the models trained on the English monolingual data made available by the WMT (Callison-Burch et al., 2012) (i.e. *Europarl*, *newscommentaries*, *news-2012* and *commoncrawl*). Tuning was performed via MERT using *newstest2010* as development set; test sentences were extracted from *newstest2011*. Finally, we restricted our SCFGs to having at most 10 target productions for a given source production.

Figure 3 shows some properties of the initial grammar $G(f)$ as a function of the input sentence length (the quantities are averages over 20 sentences for each class of input length). The number of unigrams grows linearly with the input length, while the number of unique bigrams compatible with strings generated by $G(f)$ appears to grow quadratically[9] and the size of the grammar in number of rules appears to be cubic — a consequence of having up to two nonterminals on the right-hand side of a rule.

Figure 4 shows the number of refinement operations until convergence in optimisation and sampling, as well as the total duration, as a function of the input length.[10] The plots will be discussed in detail below.

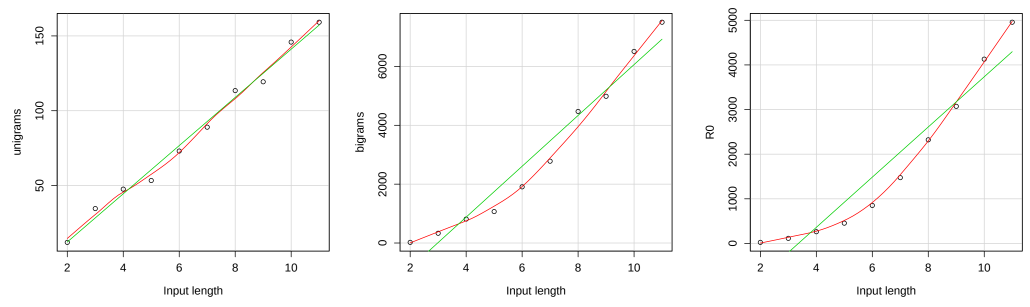
4.1 Optimisation

In optimisation (Figures 4a and 4b), the number of refinements up to convergence appears to be linear with the input length, while the total duration grows much quicker. These findings are further discussed in what follows.

Table 2 shows some important quantities regarding optimisation with OS* using a 4-gram LM. The first column shows how many sentences we are considering, the second column shows the sentence length, the third column m is the average number of refinements up to convergence. Column $|A|$ refers to the refinement type, which is the number of states in the automaton A, that is, the order of

[9]The number of unique bigrams is an estimate obtained by combining the terminals at the boundary of nonterminals that may be adjacent in a derivation.

[10]The current implementation faces timeouts depending on the length of the input sentence and the order of the language model, explaining why certain curves are interrupted earlier than others in Figure 4.



(a) Unigrams in $G(f)$ (b) Bigrams compatible with $G(f)$ (c) Number of rules in $G(f)$

Figure 3: Properties of the initial grammar as function of input length

S	Length	m	$\|A\|$	count	$\frac{\|R_f\|}{\|R_0\|}$
9	4	45.0	2	20.3	74.6 ± 53.9
			3	19.2	
			4	5.4	
10	5	62.3	2	21.9	145.4 ± 162.6
			3	32.9	
			4	7.5	
9	6	102.8	2	34.7	535.8 ± 480.0
			3	54.9	
			4	13.2	

Table 2: Optimisation with a 4-gram LM.

S	Input	m	$\|A\|$	count	$\frac{\|R_f\|}{\|R_0\|}$
10	5	1.0	2	1.0	1.9 ± 1.0
10	6	6.6	2	6.3	17.6 ± 13.6
			3	0.3	
10	7	14.5	2	12.9	93.8 ± 68.9
			3	1.5	
			4	0.1	

Table 3: Sampling with a 4-gram LM and reaching a 5% acceptance rate.

the n-grams being re-weighted (e.g. $|A| = 2$ when refining bigrams sharing a one-word context). Column *count* refers to the average number of refinements that are due to each refinement type. Finally, the last column compares the number of rules in the final proposal to that of the initial one.

The first positive result concerns how much context OS* needs to take into account for finding the optimum derivation. Table 2 (column m) shows that OS* explores a very reduced space of n-gram contexts up to convergence. To illustrate that, consider the last row in Table 2 (sentences with 6 words). On average, convergence requires incorporating only about 103 contexts of variable order, of which 55 are bigram (2-word) contexts (remember that $|A| = 3$ when accounting for a 2-word context). According to Figure 3b, in sentences with 6 words, about 2,000 bigrams are compatible with strings generated by $G(f)$. This means that only 2.75% of these bigrams (55 out of 2,000) need to be explicitly accounted for, illustrating how wasteful a full intersection would be.

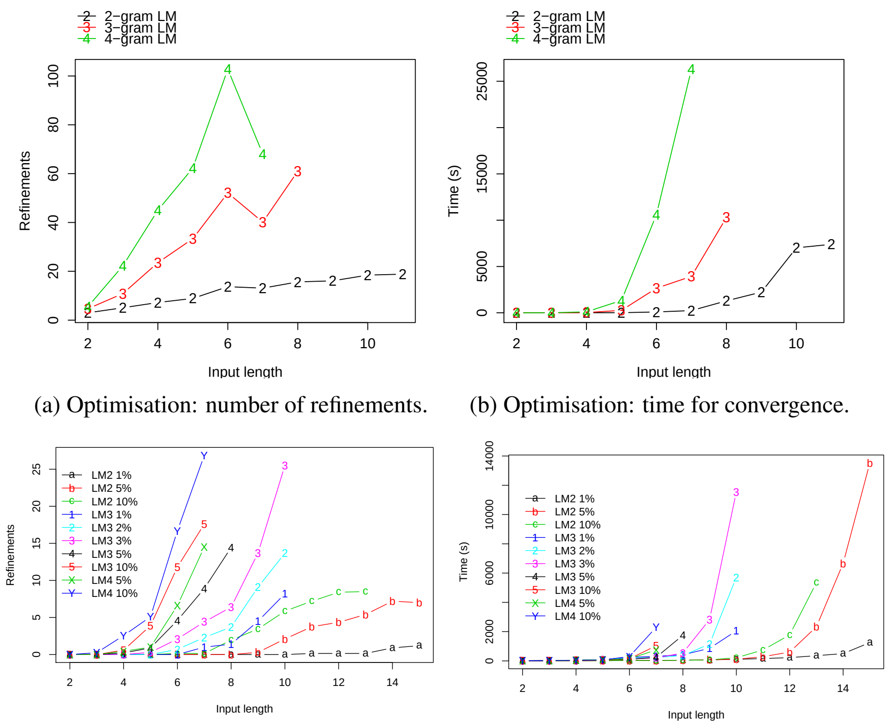
A problem, however, is that the time until convergence grows quickly with the length of the input (Figure 4b). This can be explained as follows. At each iteration the grammar is refined to account for n-grams sharing a context of $(n-1)$ words. That operation typically results in a larger grammar: most rules are preserved, some rules are deleted, but more importantly, some rules are added to account for the portion of the current grammar that involves the selected n-grams. Enlarging the grammar at each iteration means that successive refinements become incrementally slower.

The histogram of refinement types of Table 2 highlights how efficient OS* is w.r.t. the space of n-grams it needs to explore before convergence. The problem is clearly not the number of refinements, but rather the relation between the growth of the grammar and the successive intersections. Controlling for this growth and optimising the intersection as to partially reuse previously computed charts may be the key for a more generally tractable solution.

4.2 Sampling

Figure 4c shows that sampling is more economical than optimisation in that it explicitly incorporates even fewer contexts. Note how OS* converges to acceptance rates from 1% to 10% in much fewer iterations than are necessary to find an optimum[11]. Although the convergence in sampling is

[11]Currently we use MERT to train the model's weight vector — which is normalised by its L_1 norm in the Moses implementation. While optimisation is not sensitive to the scale of the weights, in sampling the scale determines how flat or



(a) Optimisation: number of refinements. (b) Optimisation: time for convergence.

(c) Sampling: number of refinements. (d) Sampling: time for convergence.

Figure 4: Convergence for different LM order as function of the input length in optimisation (top) and sampling (bottom). We show the number of refinements up to convergence on the left, and the convergence time on the right. In optimisation we stop when the true optimum is found. In sampling we stop at different acceptance rate levels: (a, b and c) use a 2-gram LM to reach 1, 5 and 10% AR; (1-4) use a 3-gram LM to reach 2, 3, 5 and 10% AR; and (X, Y) use a 4-gram LM to reach 5 and 10% AR.

faster than in optimisation, the total duration is still an issue (Figure 4b).

Table 3 shows the same quantities as Table 2, but now for sampling. It is worth highlighting that even though we are using an upper-bound over a 4-gram LM (and aiming at a 5% acceptance rate), very few contexts are selected for refinement, most of them lower-order ones (one-word contexts — rows with $|A| = 2$).

Observe that an improved acceptance rate always leads to faster acquisition of exact samples after we stop refining our proxy distribution. However, Figure 4d shows for example that moving from 5% to 10% acceptance rate using a 4-gram LM (curves X and Y) is time-consuming. Thus there is a trade-off between how much time one spends improving the acceptance rate and how many exact samples one intends do draw. Figure 5 shows the average time to draw batches between one and one million samples from two exact samplers that were refined up to 5% and 10% acceptance rate respectively. The sampler at 5% AR (which is faster to obtain) turns out to be more efficient if we aim at producing less than 10K samples.

Finally, note that samples are independently

peaked the distribution is. Arun et al. (2010) experiment with scaling MERT-trained weights as to maximise BLEU on held-out data, as well as with MBR training. A more adequate training algorithm along similar lines is reserved for future work.

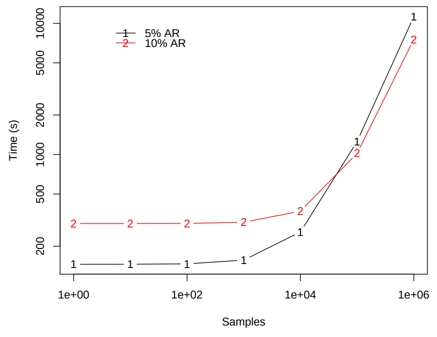


Figure 5: Average time to draw 1 to 1 million samples, for input sentences of length 6, using a 4-gram LM at 5% (curve 1) and 10% (curve 2) acceptance rate (including the time to produce the sampler).

drawn from the final proposal, making the approach an appealing candidate to parallelism in order to increase the effective acceptance rate.

5 Related Work

Rush and Collins (2011) do not consider sampling, but they address exact decoding for hierarchical translation. They use a Dual Decomposition approach (a special case of Lagrangian Relaxation), where the target CFG (hypergraph in their terminology) component and the target language model component "trade-off" their weights so as to ensure agreement on what each component believes to be the maximum. In many cases, this technique is able to detect the actual true maximum derivation. When this is not the case, they use a finite-state-based intersection mechanism to "tighten" the first component so that some constraints not satisfied by the current solution are enforced, and iterate until the true maximum is found or a time-out is met, which results in a high proportion of finding the true maximum.

Arun et al. (2009, 2010) address the question of sampling in a standard phrase-based translation model (Koehn et al., 2003). Contrarily to our use of rejection sampling (a Monte-Carlo method), they use a Gibbs sampler (a Markov-Chain Monte-Carlo (MCMC) method). Samples are obtained by iteratively re-sampling groups of well-designed variables in such a way that (i) the sampler does not tend to be trapped locally by high correlations between conditioning and conditioned variables, and (ii) the combinatorial space of possibilities for the next step is small enough so that conditional probabilities can be computed explicitly. By contrast to our exact approach, the samples obtained by Gibbs sampling are not independent, but form a Markov chain that only converges to the target distribution in the limit, with convergence properties difficult to assess. Also by contrast to us, these papers do not address the question of finding the maximum derivation directly, but only through finding a maximum among the derivations sampled so far, which in principle can be quite different.

Blunsom and Osborne (2008) address probabilistic inference, this time, as we do, in the context of hierarchical translation, where sampling is used both for the purposes of decoding and training the model. When decoding in the presence of a language model, an approximate sampling procedure is performed in two stages. First, cube-pruning is employed to construct a WCFG which generates a subset of all the possible derivations that would correspond to a full intersection with the target language model. In a second step this grammar is sampled through the same dynamic programming procedure that we have described in §2.4. By contrast to our approach, the paper does not attempt to perform exact inference. However it does not only address the question of decoding, but also that of training the model, which requires, in addition to sampling, an estimate of the model's partition function. In common with Arun et al. (2010), the authors stress the fact that a sampler of derivations is also a sampler of translations as strings, while a maximiser over derivations *cannot* be used to find the maximum translation string.

6 Conclusions

The approach we have presented is, to our knowledge, the first one to address the problem of exact sampling for hierarchical translation and to do that in a framework that also handles exact optimisation. Our experiments show that only a fraction of the language model n-grams need to be incorporated in the target grammar in order to perform exact inference in this approach. However, in the current implementation, we experience time-outs for sentences of even moderate length. We are working on improving this situation along three dimensions: (i) our implementation of the Earley Intersection rebuilds a grammar from scratch at each intersection, while it could capitalise on the charts built during the previous steps; (ii) the unigram-level max-backoffs are not as tight as they could be if one took into account more precisely the set of contexts in which each word can appear relative to the grammar; (iii) most importantly, while our refinements are "local" in the sense of addressing one n-gram context at a time, they still affect a large portion of the rules in the current grammar, even rules that have very low probability of being ever sampled by this grammar; by preventing refinement of such rules during the intersection process, we may be able to make the intersection more local and to produce much smaller grammars, without losing the exactness properties of the approach.

Acknowledgements

The first author wishes to thank the PASCAL-2 Visit to Industry programme for partially funding his visit to Xerox Research Centre Europe last Fall, which initiated this collaboration.

References

Abhishek Arun, Chris Dyer, Barry Haddow, Phil Blunsom, Adam Lopez, and Philipp Koehn. 2009. Monte carlo inference and maximization for phrase-based translation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 102–110, Stroudsburg, PA, USA. Association for Computational Linguistics.

Abhishek Arun, Barry Haddow, Philipp Koehn, Adam Lopez, Chris Dyer, and Phil Blunsom. 2010. Monte carlo techniques for phrase-based translation. *Machine Translation*, 24(2):103–121, June.

Yehoshua Bar-Hillel, Micha A. Perles, and Eli Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, (14):143–172.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver, British Columbia, Canada, June. Association for Computational Linguistics.

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 215–223, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

Simon Carter, Marc Dymetman, and Guillaume Bouchard. 2012. Exact Sampling and Decoding in High-Order Hidden Markov Models. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1125–1134, Jeju Island, Korea, July. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33:201–228.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: a decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher Dyer. 2010. *A Formal Model of Ambiguity and its Applications in Machine Translation*. Ph.D. thesis, University of Maryland.

M. Dymetman, G. Bouchard, and S. Carter. 2012a. The OS* Algorithm: a Joint Approach to Exact Optimization and Sampling. *ArXiv e-prints*, July.

Marc Dymetman, Guillaume Bouchard, and Simon Carter. 2012b. Optimization and sampling for nlp from a unified viewpoint. In *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*, pages 79–94, Mumbai, India, December. The COLING 2012 Organizing Committee.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 1 edition.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit*, pages 79–86.

Shankar Kumar and William Byrne. 2004. Minimum Bayes Risk Decoding for Statistical Machine Translation. In *Joint Conference of Human Language Technologies and the North American chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*.

Mark-Jan Nederhof and Giorgio Satta. 2008. Probabilistic parsing. In M. Dolores Jimnez-Lpez G. Bel-Enguix and C. Martn-Vide, editors, *New Developments in Formal Languages and Applications, Studies in Computational Intelligence*, volume 113, pages 229–258. Springer.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1 of *ACL '03*, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Walter Rudin. 1987. *Real and Complex Analysis*. McGraw-Hill.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 72–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 787–794, Stroudsburg, PA, USA. Association for Computational Linguistics.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June. Association for Computational Linguistics.