Computing Lattice BLEU Oracle Scores for Machine Translation

Artem Sokolov **Guillaume Wisniewski** **François Yvon**
LIMSI-CNRS & Univ. Paris Sud
BP-133, 91 403 Orsay, France
{firstname.lastname}@limsi.fr

Abstract

The search space of Phrase-Based Statistical Machine Translation (PBSMT) systems can be represented under the form of a directed acyclic graph (lattice). The quality of this search space can thus be evaluated by computing the best achievable hypothesis in the lattice, the so-called *oracle* hypothesis. For common SMT metrics, this problem is however NP-hard and can only be solved using heuristics. In this work, we present two new methods for efficiently computing BLEU oracles on lattices: the first one is based on a linear approximation of the corpus BLEU score and is solved using the FST formalism; the second one relies on integer linear programming formulation and is solved directly and using the Lagrangian relaxation framework. These new decoders are positively evaluated and compared with several alternatives from the literature for three language pairs, using lattices produced by two PBSMT systems.

1 Introduction

The search space of Phrase-Based Statistical Machine Translation (PBSMT) systems has the form of a very large directed acyclic graph. In several softwares, an approximation of this search space can be outputted, either as a n*-best list* containing the n top hypotheses found by the decoder, or as a phrase or word graph (*lattice*) which compactly encodes those hypotheses that have survived search space pruning. Lattices usually contain much more hypotheses than n-best lists and better approximate the search space.

Exploring the PBSMT search space is one of the few means to perform *diagnostic analysis* and to better understand the behavior of the system (Turchi et al., 2008; Auli et al., 2009). Useful diagnostics are, for instance, provided by looking at the best *(oracle)* hypotheses contained in the search space, *i.e*, those hypotheses that have the highest quality score with respect to one or several references. Such oracle hypotheses can be used for failure analysis and to better understand the bottlenecks of existing translation systems (Wisniewski et al., 2010). Indeed, the inability to faithfully reproduce reference translations can have many causes, such as scantiness of the translation table, insufficient expressiveness of reordering models, inadequate scoring function, non-literal references, over-pruned lattices, etc. Oracle decoding has several other applications: for instance, in (Liang et al., 2006; Chiang et al., 2008) it is used as a work-around to the problem of non-reachability of the reference in discriminative training of MT systems. Lattice reranking (Li and Khudanpur, 2009), a promising way to improve MT systems, also relies on oracle decoding to build the training data for a reranking algorithm.

For sentence level metrics, finding oracle hypotheses in n-best lists is a simple issue; however, solving this problem on lattices proves much more challenging, due to the number of embedded hypotheses, which prevents the use of brute-force approaches. When using BLEU, or rather sentence-level approximations thereof, the problem is in fact known to be NP-hard (Leusch et al., 2008). This complexity stems from the fact that the contribution of a given edge to the total *modified* n-gram precision can not be computed without looking at all other edges on the path. Similar (or worse) complexity result are expected for other metrics such as METEOR (Banerjee and Lavie, 2005) or TER (Snover et al., 2006). The exact computation of oracles under corpus level metrics, such as BLEU, poses supplementary combinatorial problems that will not be addressed in this work.

In this paper, we present two original methods for finding approximate oracle hypotheses on lattices. The first one is based on a linear approximation of the corpus BLEU, that was originally designed for efficient Minimum Bayesian Risk decoding on lattices (Tromble et al., 2008). The second one, based on Integer Linear Programming, is an extension to lattices of a recent work on failure analysis for phrase-based decoders (Wisniewski et al., 2010). In this framework, we study two decoding strategies: one based on a generic ILP solver, and one, based on Lagrangian relaxation.

Our contribution is also experimental as we compare the quality of the BLEU approximations and the time performance of these new approaches with several existing methods, for different language pairs and using the lattice generation capacities of two publicly-available state-of-the-art phrase-based decoders: Moses[1] and N-code[2].

The rest of this paper is organized as follows. In Section 2, we formally define the oracle decoding task and recall the formalism of finite state automata on semirings. We then describe (Section 3) two existing approaches for solving this task, before detailing our new proposals in sections 4 and 5. We then report evaluations of the existing and new oracles on machine translation tasks.

2 Preliminaries

2.1 Oracle Decoding Task

We assume that a *phrase-based* decoder is able to produce, for each source sentence $\mathbf{f}$, a *lattice* $L_{\mathbf{f}} = \langle Q, \Xi \rangle$, with $\#\{Q\}$ vertices (states) and $\#\{\Xi\}$ edges. Each edge carries a source phrase f_i, an associated output phrase e_i as well as a feature vector $\bar{h}_i$, the components of which encode various compatibility measures between f_i and e_i.

We further assume that $L_{\mathbf{f}}$ is a *word* lattice, meaning that each e_i carries a single word[3] and that it contains a unique initial state q_0 and a unique final state q_F. Let $\Pi_{\mathbf{f}}$ denote the set of all paths from q_0 to q_F in $L_{\mathbf{f}}$. Each path $\boldsymbol{\pi} \in \Pi_{\mathbf{f}}$ corresponds to a possible translation $\mathbf{e}_{\boldsymbol{\pi}}$. The job of a (conventional) decoder is to find the best path(s) in $L_{\mathbf{f}}$ using scores that combine the edges' feature vectors with the parameters $\bar{\lambda}$ learned during tuning.

In oracle decoding, the decoder's job is quite different, as we assume that at least a reference $\mathbf{r}_{\mathbf{f}}$ is provided to evaluate the quality of each individual hypothesis. The decoder therefore aims at finding the path $\boldsymbol{\pi}^*$ that generates the hypothesis that best matches $\mathbf{r}_{\mathbf{f}}$. For this task, only the output labels e_i will matter, the other informations can be left aside.[4]

Oracle decoding assumes the definition of a measure of the similarity between a reference and a hypothesis. In this paper we will consider sentence-level approximations of the popular BLEU score (Papineni et al., 2002). BLEU is formally defined for two parallel corpora, $\mathcal{E} = \{\mathbf{e}_j\}_{j=1}^{J}$ and $\mathcal{R} = \{\mathbf{r}_j\}_{j=1}^{J}$, each containing J sentences as:

$$n\text{-BLEU}(\mathcal{E}, \mathcal{R}) = BP \cdot \left(\prod_{m=1}^{n} p_m \right)^{1/n}, \quad (1)$$

where $BP = \min(1, e^{1 - c_1(\mathcal{R})/c_1(\mathcal{E})})$ is the brevity penalty and $p_m = c_m(\mathcal{E}, \mathcal{R})/c_m(\mathcal{E})$ are *clipped* or *modified* m-gram precisions: $c_m(\mathcal{E})$ is the total number of word m-grams in $\mathcal{E}$; $c_m(\mathcal{E}, \mathcal{R})$ accumulates over sentences the number of m-grams in $\mathbf{e}_j$ that also belong to $\mathbf{r}_j$. These counts are clipped, meaning that a m-gram that appears k times in $\mathcal{E}$ and l times in $\mathcal{R}$, with $k > l$, is only counted l times. As it is well known, BLEU performs a compromise between precision, which is directly appears in Equation (1), and recall, which is indirectly taken into account via the brevity penalty. In most cases, Equation (1) is computed with $n = 4$ and we use BLEU as a synonym for 4-BLEU.

BLEU is defined for a pair of corpora, but, as an oracle decoder is working at the sentence-level, it should rely on an approximation of BLEU that can

[1]http://www.statmt.org/moses/

[2]http://ncode.limsi.fr/

[3]Converting a *phrase* lattice to a *word* lattice is a simple matter of redistributing a compound input or output over a linear chain of arcs.

[4]The algorithms described below can be straightforwardly generalized to compute oracle hypotheses under combined metrics mixing model scores and quality measures (Chiang et al., 2008), by weighting each edge with its model score and by using these weights down the pipe.

evaluate the similarity between a single hypothesis and its reference. This approximation introduces a discrepancy as gathering sentences with the highest (local) approximation may not result in the highest possible (corpus-level) BLEU score. Let BLEU$'$ be such a sentence-level approximation of BLEU. Then lattice oracle decoding is the task of finding an optimal path $\boldsymbol{\pi}^*(\mathbf{f})$ among all paths $\Pi_{\mathbf{f}}$ for a given $\mathbf{f}$, and amounts to the following optimization problem:

$$\boldsymbol{\pi}^*(\mathbf{f}) = \arg\max_{\boldsymbol{\pi} \in \Pi_{\mathbf{f}}} \text{BLEU}'(\mathbf{e}_{\boldsymbol{\pi}}, \mathbf{r}_{\mathbf{f}}). \qquad (2)$$

2.2 Compromises of Oracle Decoding

As proved by Leusch et al. (2008), even with brevity penalty dropped, the problem of deciding whether a confusion network contains a hypothesis with clipped uni- and bigram precisions all equal to 1.0 is NP-complete (and so is the associated optimization problem of oracle decoding for 2-BLEU). The case of more general word and phrase lattices and 4-BLEU score is consequently also NP-complete. This complexity stems from chaining up of local unigram decisions that, due to the clipping constraints, have non-local effect on the bigram precision scores. It is consequently necessary to keep a possibly exponential number of non-recombinable hypotheses (characterized by counts for each n-gram in the reference) until very late states in the lattice.

These complexity results imply that any oracle decoder has to waive either the form of the objective function, replacing BLEU with better-behaved scoring functions, or the exactness of the solution, relying on approximate heuristic search algorithms.

In Table 1, we summarize different compromises that the existing (section 3), as well as our novel (sections 4 and 5) oracle decoders, have to make. The “target” and “target level” columns specify the targeted score. None of the decoders optimizes it directly: their objective function is rather the approximation of BLEU given in the “target replacement” column. Column “search” details the accuracy of the target replacement optimization. Finally, columns “clipping” and “brevity” indicate whether the corresponding properties of BLEU score are considered in the target substitute and in the search algorithm.

2.3 Finite State Acceptors

The implementations of the oracles described in the first part of this work (sections 3 and 4) use the common formalism of finite state acceptors (FSA) over different semirings and are implemented using the generic OpenFST toolbox (Allauzen et al., 2007).

A $(\oplus, \otimes)$-semiring $\mathbb{K}$ over a set K is a system $\langle K, \oplus, \otimes, \bar{0}, \bar{1}\rangle$, where $\langle K, \oplus, \bar{0}\rangle$ is a commutative monoid with identity element $\bar{0}$, and $\langle K, \otimes, \bar{1}\rangle$ is a monoid with identity element $\bar{1}$. $\otimes$ distributes over $\oplus$, so that $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ and $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$ and element $\bar{0}$ annihilates K ($a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$).

Let $A = (\Sigma, Q, I, F, E)$ be a weighted finite-state acceptor with labels in Σ and weights in $\mathbb{K}$, meaning that the transitions (q, σ, q') in A carry a weight $w \in \mathbb{K}$. Formally, E is a mapping from $(Q \times \Sigma \times Q)$ into $\mathbb{K}$; likewise, initial I and final weight F functions are mappings from Q into $\mathbb{K}$. We borrow the notations of Mohri (2009): if $\xi = (q, a, q')$ is a transition in $\mathrm{domain}(E)$, $p(\xi) = q$ (resp. $n(\xi) = q'$) denotes its origin (resp. destination) state, $w(\xi) = \sigma$ its label and $E(\xi)$ its weight. These notations extend to paths: if $\boldsymbol{\pi}$ is a path in A, $p(\boldsymbol{\pi})$ (resp. $n(\boldsymbol{\pi})$) is its initial (resp. ending) state and $w(\boldsymbol{\pi})$ is the label along the path. A finite state transducer (FST) is an FSA with output alphabet, so that each transition carries a pair of input/output symbols.

As discussed in Sections 3 and 4, several oracle decoding algorithms can be expressed as shortest-path problems, provided a suitable definition of the underlying acceptor and associated semiring. In particular, quantities such as:

$$\bigoplus_{\boldsymbol{\pi} \in \Pi(A)} E(\boldsymbol{\pi}), \qquad (3)$$

where the total weight of a successful path $\boldsymbol{\pi} = \xi_1 \dots \xi_l$ in A is computed as:

$$E(\boldsymbol{\pi}) = I(p(\xi_1)) \otimes \Big[\bigotimes_{i=1}^{l} E(\xi_i)\Big] \otimes F(n(\xi_l))$$

can be efficiently found by generic shortest distance algorithms over acyclic graphs (Mohri, 2002). For FSA-based implementations over semirings where $\oplus = \max$, the optimization problem (2) is thus reduced to Equation (3), while the oracle-specific details can be incorporated into in the definition of $\otimes$.

	oracle	target	target level	target replacement	search	clipping	brevity
existing	LM-2g/4g	2/4-BLEU	sentence	$P_2(\mathbf{e};\mathbf{r})$ or $P_4(\mathbf{e};\mathbf{r})$	exact	no	no
	PB	4-BLEU	sentence	partial $\log$ BLEU (4)	appr.	no	no
	PBℓ	4-BLEU	sentence	partial $\log$ BLEU (4)	appr.	no	yes
this paper	LB-2g/4g	2/4-BLEU	corpus	linear appr. lin BLEU (5)	exact	no	yes
	SP	1-BLEU	sentence	unigram count	exact	no	yes
	ILP	2-BLEU	sentence	uni/bi-gram counts (7)	appr.	yes	yes
	RLX	2-BLEU	sentence	uni/bi-gram counts (8)	exact	yes	yes

Table 1: Recapitulative overview of oracle decoders.

3 Existing Algorithms

In this section, we describe our reimplementation of two approximate search algorithms that have been proposed in the literature to solve the oracle decoding problem for BLEU. In addition to their approximate nature, none of them accounts for the fact that the count of each matching word has to be clipped.

3.1 Language Model Oracle (LM)

The simplest approach we consider is introduced in (Li and Khudanpur, 2009), where oracle decoding is reduced to the problem of finding the most likely hypothesis under a n-gram language model trained with the sole reference translation.

Let us suppose we have a n-gram language model that gives a probability $P(e_n|e_1 \dots e_{n-1})$ of word e_n given the $n-1$ previous words. The probability of a hypothesis $\mathbf{e}$ is then $P_n(\mathbf{e}|\mathbf{r}) = \prod_{i=1} P(e_{i+n}|e_i \dots e_{i+n-1})$. The language model can conveniently be represented as a FSA A_{LM}, with each arc carrying a negative log-probability weight and with additional ρ-type failure transitions to accommodate for back-off arcs.

If we train, for each source sentence $\mathbf{f}$, a separate language model $A_{LM}(\mathbf{r_f})$ using only the reference $\mathbf{r_f}$, oracle decoding amounts to finding a shortest (most probable) path in the weighted FSA resulting from the composition $L \circ A_{LM}(\mathbf{r_f})$ over the $(\min, +)$-semiring:

$$\boldsymbol{\pi}^*_{LM}(\mathbf{f}) = \texttt{ShortestPath}(L \circ A_{LM}(\mathbf{r_f})).$$

This approach replaces the optimization of n-BLEU with a search for the most probable path under a simplistic n-gram language model. One may expect the most probable path to select frequent n-gram from the reference, thus augmenting n-BLEU.

3.2 Partial BLEU Oracle (PB)

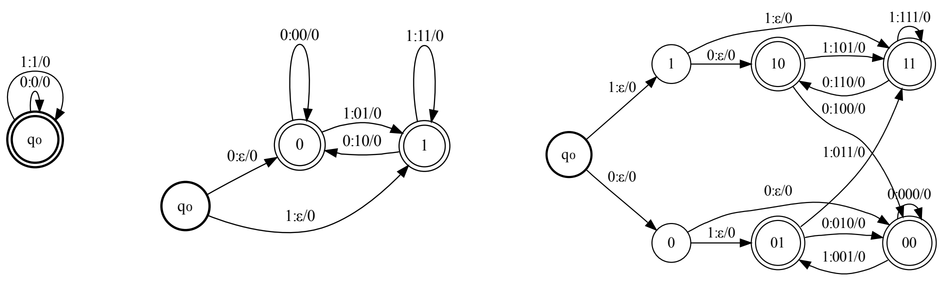
Another approach is put forward in (Dreyer et al., 2007) and used in (Li and Khudanpur, 2009): oracle translations are shortest paths in a lattice L, where the weight of each path $\boldsymbol{\pi}$ is the sentence level $\log \text{BLEU}(\boldsymbol{\pi})$ score of the corresponding complete or partial hypothesis:

$$\log \text{BLEU}(\boldsymbol{\pi}) = \frac{1}{4} \sum_{m=1\dots4} \log p_m. \qquad (4)$$

Here, the brevity penalty is ignored and n-gram precisions are offset to avoid null counts: $p_m = (c_m(\mathbf{e}_{\boldsymbol{\pi}}, \mathbf{r}) + 0.1)/(c_m(\mathbf{e}_{\boldsymbol{\pi}}) + 0.1)$.

This approach has been reimplemented using the FST formalism by defining a suitable semiring. Let each weight of the semiring keep a set of tuples accumulated up to the current state of the lattice. Each tuple contains three words of recent history, a partial hypothesis as well as current values of the length of the partial hypothesis, n-gram counts (4 numbers) and the sentence-level $\log$ BLEU score defined by Equation (4). In the beginning each arc is initialized with a singleton set containing one tuple with a single word as the partial hypothesis. For the semiring operations we define one common $\otimes$-operation and two versions of the $\oplus$-operation:
• $L_1 \otimes_{PB} L_2$ – appends a word on the edge of L_2 to L_1's hypotheses, shifts their recent histories and updates n-gram counts, lengths, and current score; • $L_1 \oplus_{PB} L_2$ – merges all sets from L_1 and L_2 and recombinates those having the same recent history; • $L_1 \oplus_{PB\ell} L_2$ – merges all sets from L_1 and L_2 and recombinates those having the same recent history *and* the same hypothesis length.

If several hypotheses have the same recent history (and length in the case of $\oplus_{PB\ell}$), recombination removes all of them, but the one



(a) Δ_1 (b) Δ_2 (c) Δ_3

Figure 1: Examples of the Δ_n automata for $\Sigma = \{0, 1\}$ and $n = 1 \ldots 3$. Initial and final states are marked, respectively, with bold and with double borders. Note that arcs between final states are weighted with 0, while in reality they will have this weight only if the corresponding n-gram does not appear in the reference.

with the largest current BLEU score. Optimal path is then found by launching the generic ShortestDistance(L) algorithm over one of the semirings above.

The $(\oplus_{PB\ell}, \otimes_{PB})$-semiring, in which the equal length requirement also implies equal brevity penalties, is more conservative in recombining hypotheses and should achieve final BLEU that is least as good as that obtained with the $(\oplus_{PB}, \otimes_{PB})$-semiring[5].

4 Linear BLEU Oracle (LB)

In this section, we propose a new oracle based on the linear approximation of the corpus BLEU introduced in (Tromble et al., 2008). While this approximation was earlier used for Minimum Bayes Risk decoding in lattices (Tromble et al., 2008; Blackwood et al., 2010), we show here how it can also be used to approximately compute an oracle translation.

Given five real parameters $\theta_{0 \ldots 4}$ and a word vocabulary Σ, Tromble et al. (2008) showed that one can approximate the *corpus*-BLEU with its first-order (linear) Taylor expansion:

$$\text{lin BLEU}(\boldsymbol{\pi}) = \theta_0 \, |\mathbf{e}_{\boldsymbol{\pi}}| + \sum_{n=1}^{4} \theta_n \sum_{u \in \Sigma^n} c_u(\mathbf{e}_{\boldsymbol{\pi}}) \delta_u(\mathbf{r}), \quad (5)$$

where $c_u(\mathbf{e})$ is the number of times the n-gram u appears in $\mathbf{e}$, and $\delta_u(\mathbf{r})$ is an indicator variable testing the presence of u in $\mathbf{r}$.

To exploit this approximation for oracle decoding, we construct four weighted FSTs Δ_n containing a (final) state for each possible $(n-1)$-gram, and all weighted transitions of the kind $(\sigma_1^{n-1}, \sigma_n : \sigma_1^n / \theta_n \times \delta_{\sigma_1^n}(\mathbf{r}), \sigma_2^n)$, where σs are in Σ, input word sequence σ_1^{n-1} and output sequence σ_2^n, are, respectively, the maximal prefix and suffix of an n-gram σ_1^n.

In supplement, we add auxiliary states corresponding to m-grams ($m < n-1$), whose functional purpose is to help reach one of the main $(n-1)$-gram states. There are $\frac{|\Sigma|^{n-1}-1}{|\Sigma|-1}, n > 1$, such supplementary states and their transitions are $(\sigma_1^k, \sigma_{k+1} : \sigma_1^{k+1}/0, \sigma_1^{k+1}), k = 1 \ldots n-2$. Apart from these auxiliary states, the rest of the graph (i.e., all final states) reproduces the structure of the well-known de Bruijn graph $B(\Sigma, n)$ (see Figure 1).

To actually compute the best hypothesis, we first weight all arcs in the input FSA L with θ_0 to obtain Δ_0. This makes each word's weight equal in a hypothesis path, and the total weight of the path in Δ_0 is proportional to the number of words in it. Then, by sequentially composing Δ_0 with other Δ_ns, we discount arcs whose output n-gram corresponds to a matching n-gram. The amount of discount is regulated by the ratio between θ_n's for $n > 0$.

With all operations performed over the $(\min, +)$-semiring, the oracle translation is then given by:

$$\boldsymbol{\pi}^*_{LB} = \texttt{ShortestPath}(\Delta_0 \circ \Delta_1 \circ \Delta_2 \circ \Delta_3 \circ \Delta_4).$$

We set parameters θ_n as in (Tromble et al., 2008): $\theta_0 = 1$, roughly corresponding to the brevity penalty (each word in a hypothesis adds up equally to the final path length) and $\theta_n = -(4p \cdot r^{n-1})^{-1}$, which are increasing discounts

[5]See, however, experiments in Section 6.

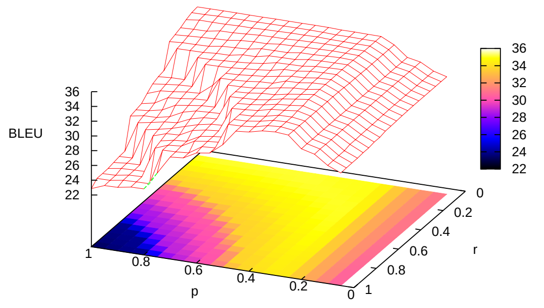


Figure 2: Performance of the LB-4g oracle for different combinations of p and r on WMT11 de2en task.

for matching n-grams. The values of p and r were found by grid search with a 0.05 step value. A typical result of the grid evaluation of the LB oracle for German to English WMT'11 task is displayed on Figure 2. The optimal values for the other pairs of languages were roughly in the same ballpark, with $p \approx 0.3$ and $r \approx 0.2$.

5 Oracles with n-gram Clipping

In this section, we describe two new oracle decoders that take n-gram clipping into account. These oracles leverage on the well-known fact that the shortest path problem, at the heart of all the oracles described so far, can be reduced straightforwardly to an Integer Linear Programming (ILP) problem (Wolsey, 1998). Once oracle decoding is formulated as an ILP problem, it is relatively easy to introduce additional constraints, for instance to enforce n-gram clipping. We will first describe the optimization problem of oracle decoding and then present several ways to efficiently solve it.

5.1 Problem Description

Throughout this section, abusing the notations, we will also think of an edge ξ_i as a binary variable describing whether the edge is "selected" or not. The set $\{0,1\}^{\#\{\Xi\}}$ of all possible edge assignments will be denoted by $\mathcal{P}$. Note that Π, the set of all paths in the lattice is a subset of $\mathcal{P}$: by enforcing some constraints on an assignment $\boldsymbol{\xi}$ in $\mathcal{P}$, it can be guaranteed that it will represent a path in the lattice. For the sake of presentation, we assume that each edge ξ_i generates a single word $w(\xi_i)$ and we focus first on finding the optimal hypothesis with respect to the sentence approximation of the 1-BLEU score.

As 1-BLEU is decomposable, it is possible to define, for every edge ξ_i, an associated reward, θ_i that describes the edge's local contribution to the hypothesis score. For instance, for the sentence approximation of the 1-BLEU score, the rewards are defined as:

$$\theta_i = \begin{cases} \Theta_1 & \text{if } w(\xi_i) \text{ is in the reference,} \\ -\Theta_2 & \text{otherwise,} \end{cases}$$

where Θ_1 and Θ_2 are two positive constants chosen to maximize the corpus BLEU score[6]. Constant Θ_1 (resp. Θ_2) is a reward (resp. a penalty) for generating a word in the reference (resp. *not* in the reference). The score of an assignment $\boldsymbol{\xi} \in \mathcal{P}$ is then defined as: $\text{score}(\boldsymbol{\xi}) = \sum_{i=1}^{\#\{\Xi\}} \xi_i \cdot \theta_i$. This score can be seen as a compromise between the number of common words in the hypothesis and the reference (accounting for recall) and the number of words of the hypothesis that do not appear in the reference (accounting for precision).

As explained in Section 2.3, finding the oracle hypothesis amounts to solving the shortest distance (or path) problem (3), which can be reformulated by a constrained optimization problem (Wolsey, 1998):

$$\begin{aligned} &\arg\max_{\boldsymbol{\xi} \in \mathcal{P}} \sum_{i=1}^{\#\{\Xi\}} \xi_i \cdot \theta_i \qquad (6) \\ &\quad \text{s.t.} \sum_{\xi \in \Xi^-(q_F)} \xi = 1, \quad \sum_{\xi \in \Xi^+(q_0)} \xi = 1 \\ &\quad \sum_{\xi \in \Xi^+(q)} \xi - \sum_{\xi \in \Xi^-(q)} \xi = 0, \ q \in \mathcal{Q} \backslash \{q_0, q_F\} \end{aligned}$$

where q_0 (resp. q_F) is the initial (resp. final) state of the lattice and $\Xi^-(q)$ (resp. $\Xi^+(q)$) denotes the set of incoming (resp. outgoing) edges of state q. These *path constraints* ensure that the solution of the problem is a valid path in the lattice.

The optimization problem in Equation (6) can be further extended to take clipping into account. Let us introduce, for each word w, a variable γ_w that denotes the number of times w appears in the hypothesis clipped to the number of times, it appears in the reference. Formally, γ_w is defined by:

$$\gamma_w = \min \left\{ \sum_{\xi \in \Omega(w)} \xi, c_w(\mathbf{r}) \right\}$$

[6]We tried several combinations of Θ_1 and Θ_2 and kept the one that had the highest corpus 4-BLEU score.

where $\Omega(w)$ is the subset of edges generating w, and $\sum_{\xi \in \Omega(w)} \xi$ is the number of occurrences of w in the solution and $c_w(\mathbf{r})$ is the number of occurrences of w in the reference $\mathbf{r}$. Using the γ variables, we define a "clipped" approximation of 1-BLEU:

$$\Theta_1 \cdot \sum_w \gamma_w - \Theta_2 \cdot \left(\sum_{i=1}^{\#\{\Xi\}} \xi_i - \sum_w \gamma_w \right)$$

Indeed, the clipped number of words in the hypothesis that appear in the reference is given by $\sum_w \gamma_w$, and $\sum_{i=1}^{\#\{\Xi\}} \xi_i - \sum_w \gamma_w$ corresponds to the number of words in the hypothesis that do not appear in the reference or that are surplus to the clipped count.

Finally, the clipped lattice oracle is defined by the following optimization problem:

$$\begin{aligned} \arg\max_{\boldsymbol{\xi} \in \mathcal{P}, \gamma_w} \quad & (\Theta_1 + \Theta_2) \cdot \sum_w \gamma_w - \Theta_2 \cdot \sum_{i=1}^{\#\{\Xi\}} \xi_i \qquad (7) \\ \text{s.t.} \quad & \gamma_w \geq 0, \gamma_w \leq c_w(\mathbf{r}), \gamma_w \leq \sum_{\xi \in \Omega(w)} \xi \\ & \sum_{\xi \in \Xi^-(q_F)} \xi = 1, \sum_{\xi \in \Xi^+(q_0)} \xi = 1 \\ & \sum_{\xi \in \Xi^+(q)} \xi - \sum_{\xi \in \Xi^-(q)} \xi = 0, \; q \in Q \setminus \{q_0, q_F\} \end{aligned}$$

where the first three sets of constraints are the linearization of the definition of γ_w, made possible by the positivity of Θ_1 and Θ_2, and the last three sets of constraints are the path constraints.

In our implementation we generalized this optimization problem to *bigram* lattices, in which each edge is labeled by the bigram it generates. Such bigram FSAs can be produced by composing the word lattice with Δ_2 from Section 4. In this case, the reward of an edge will be defined as a combination of the (clipped) number of unigram matches and bigram matches, and solving the optimization problem yields a 2-BLEU optimal hypothesis. The approach can be further generalized to higher-order BLEU or other metrics, as long as the reward of an edge can be computed locally.

The constrained optimization problem (7) can be solved efficiently using off-the-shelf ILP solvers[7].

[7]In our experiments we used Gurobi (Optimization, 2010) a commercial ILP solver that offers free academic license.

5.2 Shortest Path Oracle (SP)

As a trivial special class of the above formulation, we also define a Shortest Path Oracle (SP) that solves the optimization problem in (6). As no clipping constraints apply, it can be solved efficiently using the standard Bellman algorithm.

5.3 Oracle Decoding through Lagrangian Relaxation (RLX)

In this section, we introduce another method to solve problem (7) without relying on an external ILP solver. Following (Rush et al., 2010; Chang and Collins, 2011), we propose an original method for oracle decoding based on Lagrangian relaxation. This method relies on the idea of relaxing the clipping constraints: starting from an unconstrained problem, the counts clipping is enforced by incrementally strengthening the weight of paths satisfying the constraints.

The oracle decoding problem with clipping constraints amounts to solving:

$$\begin{aligned} \arg\min_{\boldsymbol{\xi} \in \Pi} \quad & - \sum_{i=1}^{\#\{\Xi\}} \xi_i \cdot \theta_i \qquad (8) \\ \text{s.t.} \quad & \sum_{\xi \in \Omega(w)} \xi \leq c_w(\mathbf{r}), w \in \mathbf{r} \end{aligned}$$

where, by abusing the notations, $\mathbf{r}$ also denotes the set of words in the reference. For sake of clarity, the path constraints are incorporated into the domain (the $\arg\min$ runs over Π and not over $\mathcal{P}$). To solve this optimization problem we consider its dual form and use Lagrangian relaxation to deal with clipping constraints.

Let $\boldsymbol{\lambda} = \{\lambda_w\}_{w \in \mathbf{r}}$ be positive Lagrange multipliers, one for each different word of the reference, then the Lagrangian of the problem (8) is:

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\xi}) = - \sum_{i=1}^{\#\{\Xi\}} \xi_i \theta_i + \sum_{w \in \mathbf{r}} \lambda_w \left(\sum_{\xi \in \Omega(w)} \xi - c_w(\mathbf{r}) \right)$$

The dual objective is $\mathcal{L}(\boldsymbol{\lambda}) = \min_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\xi})$ and the dual problem is: $\max_{\boldsymbol{\lambda}, \boldsymbol{\lambda} \succeq 0} \mathcal{L}(\boldsymbol{\lambda})$. To solve the latter, we first need to work out the dual objective:

$$\begin{aligned} \boldsymbol{\xi}^* &= \arg\min_{\boldsymbol{\xi} \in \Pi} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\xi}) \\ &= \arg\min_{\boldsymbol{\xi} \in \Pi} \sum_{i=1}^{\#\{\Xi\}} \xi_i \left(\lambda_{w(\xi_i)} - \theta_i \right) \end{aligned}$$

where we assume that $\lambda_{w(\xi_i)}$ is 0 when word $w(\xi_i)$ is not in the reference. In the same way as in Section 5.2, the solution of this problem can be efficiently retrieved with a shortest path algorithm.

It is possible to optimize $\mathcal{L}(\boldsymbol{\lambda})$ by noticing that it is a concave function. It can be shown (Chang and Collins, 2011) that, at convergence, the clipping constraints will be enforced in the optimal solution. In this work, we chose to use a simple gradient descent to solve the dual problem. A subgradient of the dual objective is:

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_w} = \sum_{\xi \in \Omega(w) \cap \xi^*} \xi - c_w(\mathbf{r}).$$

Each component of the gradient corresponds to the difference between the number of times the word w appears in the hypothesis and the number of times it appears in the reference. The algorithm below sums up the optimization of task (8). In the algorithm $\alpha^{(t)}$ corresponds to the step size at the t^{th} iteration. In our experiments we used a constant step size of 0.1. Compared to the usual gradient descent algorithm, there is an additional projection step of $\boldsymbol{\lambda}$ on the positive orthant, which enforces the constraint $\boldsymbol{\lambda} \succeq 0$.

```
$\forall w, \lambda_w^{(0)} \leftarrow 0$
**for** $t = 1 \rightarrow T$ **do**
    $\boldsymbol{\xi}^{*(t)} = \arg\min_{\boldsymbol{\xi}} \sum_i \xi_i \cdot \left(\lambda_{w(\xi_i)} - \theta_i\right)$
    **if** *all clipping constraints are enforced*
    **then** optimal solution found
    **else for** $w \in \mathbf{r}$ **do**
        $n_w \leftarrow$ n. of occurrences of $w$ in $\boldsymbol{\xi}^{*(t)}$
        $\lambda_w^{(t)} \leftarrow \lambda_w^{(t)} + \alpha^{(t)} \cdot (n_w - c_w(\mathbf{r}))$
        $\lambda_w^{(t)} \leftarrow \max(0, \lambda_w^{(t)})$
```

6 Experiments

For the proposed new oracles and the existing approaches, we compare the quality of oracle translations and the average time per sentence needed to compute them[8] on several datasets for 3 language pairs, using lattices generated by two open-source decoders: N-code and Moses[9] (Figures 3 and 4). Systems were trained on the data provided for the WMT'11 Evaluation task[10], tuned on the WMT'09 test data and evaluated on WMT'10 test set[11] to produce lattices. The BLEU test scores and oracle scores on 100-best lists with the approximation (4) for N-code and Moses are given in Table 2. It is not until considering 10,000-best lists that n-best oracles achieve performance comparable to the (mediocre) SP oracle.

	decoder	fr2en	de2en	en2de
test	N-code	27.88	22.05	15.83
	Moses	27.68	21.85	15.89
oracle	N-code	36.36	29.22	21.18
	Moses	35.25	29.13	22.03

Table 2: Test BLEU scores and oracle scores on 100-best lists for the evaluated systems.

To make a fair comparison with the ILP and RLX oracles which optimize 2-BLEU, we included 2-BLEU versions of the LB and LM oracles, identified below with the "-2g" suffix. The two versions of the PB oracle are respectively denoted as PB and PBℓ, by the type of the $\oplus$-operation they consider (Section 3.2). Parameters p and r for the LB-4g oracle for N-code were found with grid search and reused for Moses: $p = 0.25, r = 0.15$ (fr2en); $p = 0.175, r = 0.575$ (en2de) and $p = 0.35, r = 0.425$ (de2en). Correspondingly, for the LB-2g oracle: $p = 0.3, r = 0.15$; $p = 0.3, r = 0.175$ and $p = 0.575, r = 0.1$.

The proposed LB, ILP and RLX oracles were the best performing oracles, with the ILP and RLX oracles being considerably faster, suffering only a negligible decrease in BLEU, compared to the 4-BLEU-optimized LB oracle. We stopped RLX oracle after 20 iterations, as letting it converge had a small negative effect ($\sim$1 point of the corpus BLEU), because of the sentence/corpus discrepancy ushered by the BLEU score approximation.

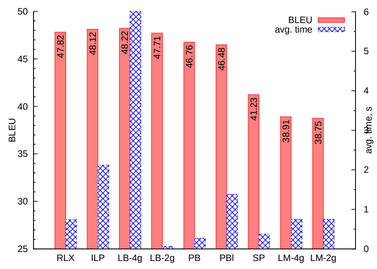
Experiments showed consistently inferior performance of the LM-oracle resulting from the optimization of the sentence probability rather than BLEU. The PB oracle often performed comparably to our new oracles, however, with sporadic resource-consumption bursts, that are difficult to

[8]Experiments were run in parallel on a server with 64G of RAM and 2 Xeon CPUs with 4 cores at 2.3 GHz.

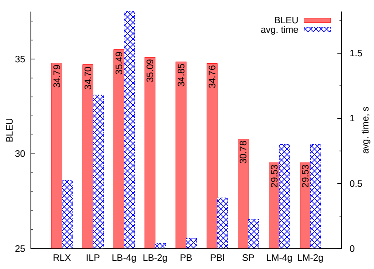
[9]As the ILP (and RLX) oracle were implemented in Python, we pruned Moses lattices to accelerate task preparation for it.

[10]http://www.statmt.org/wmt2011

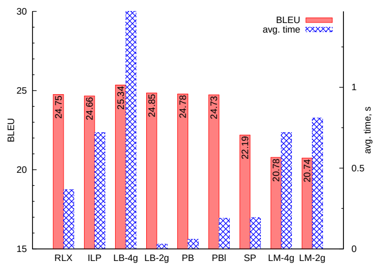
[11]All BLEU scores are reported using the multi-bleu.pl script.



(a) fr2en

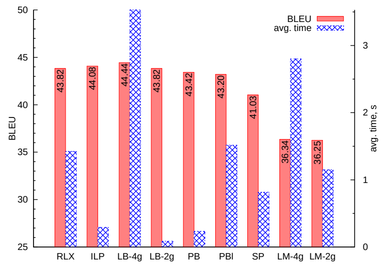


(b) de2en

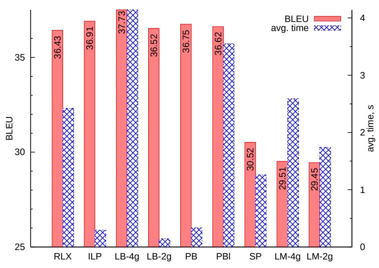


(c) en2de

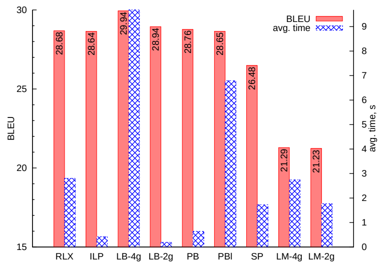
Figure 3: Oracles performance for N-code lattices.



(a) fr2en



(b) de2en



(c) en2de

Figure 4: Oracles performance for Moses lattices pruned with parameter `-b 0.5`.

avoid without more cursory hypotheses recombination strategies and the induced effect on the translations quality. The length-aware PBℓ oracle has unexpectedly poorer scores compared to its length-agnostic PB counterpart, while it should, at least, stay even, as it takes the brevity penalty into account. We attribute this fact to the complex effect of clipping coupled with the lack of control of the process of selecting one hypothesis among several having the same BLEU score, length and recent history. Anyhow, BLEU scores of both of PB oracles are only marginally different, so the PBℓ's conservative policy of pruning and, consequently, much heavier memory consumption makes it an unwanted choice.

7 Conclusion

We proposed two methods for finding oracle translations in lattices, based, respectively, on a linear approximation to the corpus-level BLEU and on integer linear programming techniques. We also proposed a variant of the latter approach based on Lagrangian relaxation that does not rely on a third-party ILP solver. All these oracles have superior performance to existing approaches, in terms of the quality of the found translations, resource consumption and, for the LB-2g oracles, in terms of speed. It is thus possible to use better approximations of BLEU than was previously done, taking the corpus-based nature of BLEU, or clipping constrainst into account, delivering better oracles without compromising speed.

Using 2-BLEU and 4-BLEU oracles yields comparable performance, which confirms the intuition that hypotheses sharing many 2-grams, would likely have many common 3- and 4-grams as well. Taking into consideration the exceptional speed of the LB-2g oracle, in practice one can safely optimize for 2-BLEU instead of 4-BLEU, saving large amounts of time for oracle decoding on long sentences.

Overall, these experiments accentuate the acuteness of scoring problems that plague modern decoders: very good hypotheses exist for most input sentences, but are poorly evaluated by a linear combination of standard features functions. Even though the tuning procedure can be held responsible for part of the problem, the comparison between lattice and n-best oracles shows that the beam search leaves good hypotheses out of the n-best list until very high value of n, that are never used in practice.

Acknowledgments

This work has been partially funded by OSEO under the Quaero program.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. of the Int. Conf. on Implementation and Application of Automata*, pages 11–23.

Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. A systematic analysis of translation model search spaces. In *Proc. of WMT*, pages 224–232, Athens, Greece.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation*, pages 65–72, Ann Arbor, MI, USA.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Efficient path counting transducers for minimum bayes-risk decoding of statistical machine translation lattices. In *Proc. of the ACL 2010 Conference Short Papers*, pages 27–32, Stroudsburg, PA, USA.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proc. of the 2011 Conf. on EMNLP*, pages 26–37, Edinburgh, UK.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of the 2008 Conf. on EMNLP*, pages 224–233, Honolulu, Hawaii.

Markus Dreyer, Keith B. Hall, and Sanjeev P. Khudanpur. 2007. Comparing reordering constraints for SMT using efficient BLEU oracle computation. In *Proc. of the Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Morristown, NJ, USA.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the BLEU-optimal hypothesis in a confusion network. In *Proc. of the 2008 Conf. on EMNLP*, pages 839–847, Honolulu, Hawaii.

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient extraction of oracle-best translations from hypergraphs. In *Proc. of Human Language Technologies: The 2009 Annual Conf. of the North American Chapter of the ACL, Companion Volume: Short Papers*, pages 9–12, Morristown, NJ, USA.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of the 21st Int. Conf. on Computational Linguistics and the 44th annual meeting of the ACL*, pages 761–768, Morristown, NJ, USA.

Mehryar Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7:321–350.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 6, pages 213–254.

Gurobi Optimization. 2010. Gurobi optimizer, April. Version 3.0.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the Annual Meeting of the ACL*, pages 311–318.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of the 2010 Conf. on EMNLP*, pages 1–11, Stroudsburg, PA, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of the Conf. of the Association for Machine Translation in the America (AMTA)*, pages 223–231.

Roy W. Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Proc. of the Conf. on EMNLP*, pages 620–629, Stroudsburg, PA, USA.

Marco Turchi, Tijl De Bie, and Nello Cristianini. 2008. Learning performance of a machine translation system: a statistical and computational analysis. In *Proc. of WMT*, pages 35–43, Columbus, Ohio.

Guillaume Wisniewski, Alexandre Allauzen, and François Yvon. 2010. Assessing phrase-based translation models with oracle decoding. In *Proc. of the 2010 Conf. on EMNLP*, pages 933–943, Stroudsburg, PA, USA.

L. Wolsey. 1998. *Integer Programming*. John Wiley & Sons, Inc.