Pre-reordering for machine translation using transition-based walks on dependency parse trees

Antonio Valerio Miceli-Barone
Dipartimento di Informatica
Largo B. Pontecorvo, 3
56127 Pisa, Italy
miceli@di.unipi.it

Giuseppe Attardi
Dipartimento di Informatica
Largo B. Pontecorvo, 3
56127 Pisa, Italy
attardi@di.unipi.it

Abstract

We propose a pre-reordering scheme to improve the quality of machine translation by permuting the words of a source sentence to a target-like order. This is accomplished as a transition-based system that walks on the dependency parse tree of the sentence and emits words in target-like order, driven by a classifier trained on a parallel corpus. Our system is capable of generating arbitrary permutations up to flexible constraints determined by the choice of the classifier algorithm and input features.

1 Introduction

The dominant paradigm in statistical machine translation consists mainly of phrase-based system such as Moses (Koehn et.al.,2007). Different languages, however, often express the same concepts in different idiomatic word orders, and while phrase-based system can deal to some extent with short-distance word swaps that are captured by short segments, they typically perform poorly on long-distance (more than four or five words apart) reordering. In fact, according to (Birch et.al., 2008), the amount of reordering between two languages is the most predictive feature of phrase-based translation accuracy.

A number of approaches to deal with long-distance reordering have been proposed. Since an extuasive search of the permutation space is unfeasible, these approaches typically constrain the search space by leveraging syntactical structure of natural languages.

In this work we consider approaches which involve reordering the words of a source sentence in a target-like order as a preprocessing step, before feeding it to a phrase-based decoder which has itself been trained with a reordered training set. These methods also try to leverage syntax, typically by applying hand-coded or automatically induced reordering rules to a constituency or dependency parse of the source sentence. (Galley and Manning, 2008; Xu et.al., 2009; Genzel, 2010; Isozaki et.al., 2010) or by treating reordering as a global optimization problem (Tromble and Eisner, 2009; Visweswariah et.al., 2011). In order to keep the training and execution processes tractable, these methods impose hard constrains on the class of permutations they can generate.

We propose a pre-reordering method based on a walk on the dependency parse tree of the source sentence driven by a classifier trained on a parallel corpus.

In principle, our system is capable of generating arbitrary permutations of the source sentence. Practical implementations will necessarily limit the available permutations, but these constraints are not intrinsic to the model, rather they depend on the specific choice of the classifier algorithm, its hyper-parameters and input features.

2 Reordering as a walk on a dependency tree

2.1 Dependency parse trees

Let a sentence be a list of words $s \equiv (w_1, w_2, \ldots, w_n)$ and its dependency parse tree be a rooted tree whose nodes are the words of the sentence. An edge of the tree represents a syntactical dependency relation between a head (parent) word and a modifier (child) word. Typical dependency relations include *verb-subject*, *verb-object*, *noun-adjective*, and so on.

We assume that in addition to its head h_i and dependency relation type d_i each word is also annotated with a part-of-speech p_i and optionally a lemma l_i and a morphology m_i (e.g. grammatical case, gender, number, tense).

Some definitions require dependency parse trees to be *projective*, meaning that any complete

Proceedings of the Eighth Workshop on Statistical Machine Translation, pages 164–169,
Sofia, Bulgaria, August 8-9, 2013

subtree must correspond to a contiguous span of words in the sentence, however, we don't place such a requirement. In practice, languages with a substantially strict word ordering like English typically have largely projective dependencies, while languages with a more free word ordering like Czech can have substantial non-projectivity.

2.2 Reordering model

Given a sentence $s \in S$ with its dependency parse tree and additional annotations, we incrementally construct a reordered sentence s' by emitting its words in a sequence of steps. We model the reordering process as a non-deterministic transition system which traverses the parse tree:

Let the state of the system be a tuple $x \equiv (i,\ r,\ a,\ ,\dots)$ containing at least the index of the current node i (initialized at the root), the list of emitted nodes r (initialized as empty) and the last transition action a (initialized as $null$). Additional information can be included in the state x, such as the list of the last K nodes that have been visited, the last K actions and a visit count for each node.

At each step we choose one of the following actions:

- $EMIT$: emit the current node. Enabled only if the current node hasn't already been emitted

$$\frac{i \notin r}{(i,\ r,\ a,\ ,\dots) \overset{EMIT}{\rightarrow} (i,\ (r \mid i),\ EMIT,\ ,\dots)}$$

- UP: move to the parent of the current node

$$\frac{h_i \neq null,\ \forall j\ a \neq DOWN_j}{(i,\ r,\ a,\ ,\dots) \overset{UP}{\rightarrow} (h_i,\ r,\ UP,\ ,\dots)}$$

- $DOWN_j$: move to the child j of the current node. Enabled if the subtree of j (including j) contains nodes that have not been emitted yet.

$$\frac{h_j = i,\ a \neq UP,\ \exists k \in subtree(i) :\ k \notin r}{(i,\ r,\ a,\ ,\dots) \overset{DOWN_j}{\rightarrow} (j,\ r,\ DOWN_j,\ ,\dots)}$$

The pre-conditions on the UP and DOWN actions prevent them from canceling each other, ensuring that progress is made at each step. The additional precondition on DOWN actions ensures that the process always halts at a final state where all the nodes have been emitted.

Let $T(s)$ be the set of legal traces of the transition system for sentence s. Each trace $\tau \in T(s)$ defines a permutation s_τ of s as the list of emitted nodes r of its final state.

We define the reordering problem as finding the trace τ^* that maximizes a scoring function Φ

$$\tau^* \equiv \arg\max_{\tau \in T(s)} \Phi(s,\ \tau) \tag{1}$$

Note that since the parse tree is connected, in principle any arbitrary permutation can be generated for a suitable choice of Φ, though the maximization problem (1) is NP-hard and APX-complete in the general case, by trivial reduction from the traveling salesman problem.

The intuition behind this model is to leverage the syntactical information provided by the dependency parse tree, as successfully done by (Xu et.al., 2009; Genzel, 2010; Isozaki et.al., 2010) without being strictly constrained by a specific type reordering rules.

2.3 Trace scores

We wish to design a scoring function Φ that captures good reorderings for machine translation and admits an efficient optimization scheme.

We chose a function that additively decomposes into local scoring functions, each depending only on a single state of the trace and the following transition action

$$\Phi(s,\ \tau) \equiv \sum_{t=1}^{|\tau|-1} \phi(s,\ x(\tau,\ t),\ x_a(\tau,\ t+1)) \tag{2}$$

We further restrict our choice to a function which is linear w.r.t. a set of elementary local feature functions $\{f_k\}$

$$\phi(s,\ x,\ a) \equiv \sum_{k=1}^{|F|} v_k f_k(s,\ x,\ a) \tag{3}$$

where $\{v_k\} \in \mathbb{R}^{|F|}$ is a vector of parameters derived from a training procedure.

While in principle each feature function could depend on the whole sentence and the whole sequence of nodes emitted so far, in practice we restrict the dependence to a fixed neighborhood of the current node and the last few emitted nodes. This reduces the space of possible permutations.

2.4 Classifier-driven action selection

Even when the permutation space has been restricted by an appropriate choice of the feature functions, computing an exact solution of the optimization problem (1) remains non-trivial, because

at each step of the reordering generation process, the set of enabled actions depends in general on nodes emitted at any previous step, and this prevents us from applying typical dynamic programming techniques. Therefore, we need to apply an heuristic procedure.

In our experiments, we apply a simple greedy procedure: at each step we choose an action according to the output a two-stage classifier:

1. A three-class one-vs-all logistic classifier chooses an action among EMIT, UP or DOWN based on a vector of features extracted from a fixed neighborhood of the current node i, the last emitted nodes and additional content of the state.

2. If a DOWN action was chosen, then a one-vs-one voting scheme is used to choose which child to descend to: For each pair $(j,\ j') :\ j < j'$ of children of i, a binary logistic classifier assigns a vote either to j or j'. The child that receives most votes is chosen. This is similar to the max-wins approach used in packages such as LIBSVM (Chang and Lin, 2011) to construct a M-class classifier from $M\,(M-1)\,/2$ binary classifiers, except that we use a single binary classifier acting on a vector of features extracted from the pair of children $(j,\ j')$ and the node i, with their respective neighborhoods.

We also experimented with different classification schemes, but we found that this one yields the best performance.

Note that we are not strictly maximizing a global linear scoring function as as defined by equations (2) and (3), although this approach is closely related to that framework.

This approach is related to transition-based dependency parsing such as (Nivre and Scholz, 2004; Attardi, 2006) or dependency tree revision(Attardi and Ciaramita, 2007).

3 Training

3.1 Dataset preparation

Following (Al-Onaizan and Papineni, 2006; Tromble and Eisner, 2009; Visweswariah et.al., 2011), we generate a source-side reference reordering of a parallel training corpus. For each sentence pair, we generate a bidirectional word alignment using GIZA++ (Och and Ney, 2000) and the "grow-diag-final-and" heuristic implemented in Moses (Koehn et.al.,2007), then we assign to each source-side word a integer index corresponding to the position of the leftmost target-side word it is aligned to (attaching unaligned words to the following aligned word) and finally we perform a stable sort of source-side words according to this index.

On language pairs where GIZA++ produces substantially accurate alignments (generally all European languages) this scheme generates a target-like reference reordering of the corpus.

In order to tune the parameters of the downstream phrase-based translation system and to test the overall translation accuracy, we need two additional small parallel corpora. We don't need a reference reordering for the tuning corpus since it is not used for training the reordering system, however we generate a reference reordering for the test corpus in order to evaluate the accuracy of the reordering system in isolation. We obtain an alignment of this corpus by appending it to the training corpus, and processing it with GIZA++ and the heuristic described above.

3.2 Reference traces generation and classifier training

For each source sentence s in the training set and its reference reordering s', we generate a minimum-length trace τ of the reordering transition system, and for each state and action pair in it we generate the following training examples:

- For the first-stage classifier we generate a single training examples mapping the local features to an EMIT, UP or DOWN action label
- For the second-stage classifier, if the action is $DOWN_j$, for each pair of children $(k,\ k') :\ k < k'$ of the current node i, we generate a positive example if $j = k$ or a negative example if $j = k'$.

Both classifiers are trained with the LIBLINEAR package (Fan et.al., 2008), using the L2-regularized logistic regression method. The regularization parameter C is chosen by two-fold cross-validation. In practice, subsampling of the training set might be required in order to keep memory usage and training time manageable.

3.3 Translation system training and testing

Once the classifiers have been trained, we run the reordering system on the source side of the

whole (non-subsampled) training corpus and the tuning corpus. For instance, if the parallel corpora are German-to-English, after the reordering step we obtain *German'*-to-English corpora, where *German'* is German in an English-like word order. These reordered corpora are used to train a standard phrase-based translation system. Finally, the reordering system is applied to source side of the test corpus, which is then translated with the downstream phrase-based system and the resulting translation is compared to the reference translation in order to obtain an accuracy measure. We also evaluate the "monolingual" reordering accuracy of upstream reordering system by comparing its output on the source side of the test corpus to the reference reordering obtained from the alignment.

4 Experiments

We performed German-to-English and Italian-to-English reordering and translation experiments.

4.1 Data

The German-to-English corpus is Europarl v7 (Koehn, 2005). We split it in a 1,881,531 sentence pairs training set, a 2,000 sentence pairs development set (used for tuning) and a 2,000 sentence pairs test set. We also used a 3,000 sentence pairs "challenge" set of newspaper articles provided by the WMT 2013 translation task organizers.

The Italian-to-English corpus has been assembled by merging Europarl v7, JRC-ACQUIS v2.2 (Steinberger et.al., 2006) and bilingual newspaper articles crawled from news websites such as Corriere.it and Asianews.it. It consists of a 3,075,777 sentence pairs training set, a 3,923 sentence pairs development set and a 2,000 sentence pairs test set.

The source sides of these corpora have been parsed with Desr (Attardi, 2006). For both language pairs, we trained a baseline Moses phrase-based translation system with the default configuration (including lexicalized reordering).

In order to keep the memory requirements and duration of classifier training manageable, we subsampled each training set to 40,000 sentences, while both the baseline and reordered Moses system are trained on the full training sets.

4.2 Features

After various experiments with feature selection, we settled for the following configuration for both German-to-English and Italian-to-English:

- First stage classifier: current node i stateful features (emitted?, left/right subtree emitted?, visit count), curent node lexical and syntactical features (surface form w_i, lemma l_i, POS p_i, morphology m_i, DEPREL d_i, and pairwise combinations between lemma, POS and DEPREL), last two actions, last two visited nodes POS, DEPREL and visit count, last two emitted nodes POS and DEPREL, bigram and syntactical trigram features for the last two emitted nodes and the current node, all lexical, syntactical and stateful features for the neighborhood of the current node (left, right, parent, parent-left, parent-right, grandparent, left-child, right-child) and pairwise combination between syntactical features of these nodes.

- Second stage classifier: stateful features for the current node i and the the children pair (j, j'), lexical and syntactical features for each of the children and pairwise combinations of these features, visit count differences and signed distances between the two children and the current node, syntactical trigram features between all combinations of the two children, the current node, the parent h_i and the two last emitted nodes and the two last visited nodes, lexical and syntactical features for the two children left and right neighbors.

All features are encoded as binary one-of-n indicator functions.

4.3 Results

For both German-to-English and Italian-to-English experiments, we prepared the data as described above and we trained the classifiers on their subsampled training sets. In order to evaluate the classifiers accuracy in isolation from the rest of the system, we performed two-fold cross validation on the same training sets, which revealed an high accuracy: The first stage classifier obtains approximately 92% accuracy on both German and Italian, while the second stage classifier obtains approximately 89% accuracy on German and 92% on Italian.

	BLEU	NIST
German	57.35	13.2553
Italian	68.78	15.3441

Table 1: Monolingual reordering scores

	BLEU	NIST
de-en baseline	**33.78**	**7.9664**
de-en reordered	32.42	7.8202
it-en baseline	**29.17**	**7.1352**
it-en reordered	28.84	7.1443

Table 2: Translation scores

We applied the reordering preprocessing system to the source side of the corpora and evaluated the monolingual BLEU and NIST score of the test sets (extracted from Europarl) against their reference reordering computed from the alignment

To evaluate translation performance, we trained a Moses phrase-based system on the reordered training and tuning corpora, and evaluated the BLEU and NIST of the (Europarl) test sets. As a baseline, we also trained and evaluated Moses system on the original unreordered corpora.

We also applied our baseline and reordered German-to-English systems to the WMT2013 translation task dataset.

5 Discussion

Unfortunately we were generally unable to improve the translation scores over the baseline, even though our monolingual BLEU for German-to-English reordering is higher than the score reported by (Tromble and Eisner, 2009) for a comparable dataset.

Accuracy on the WMT 2013 set is very low. We attribute this to the fact that it comes form a different domain than the training set.

Since classifier training set cross-validation accuracy is high, we speculate that the main problem lies with the training example generation process: training examples are generated only from optimal reordering traces. This means that once the classifiers produce an error and the system strays away from an optimal trace, it may enter in a feature space that is not well-represented in the training set, and thus suffer from unrecoverable performance degradation. Moreover, errors occurring on nodes high in the parse tree may cause incorrect placement of whole spans of words, yielding a poor BLEU score (although a cursory examination of the reordered sentences doesn't reveal this problem to be prevalent). Both these issues could be possibly addressed by switching from a classifier-based system to a structured prediction system, such as averaged structured perceptron (Collins, 2002) or MIRA (Crammer, 2003; McDonald et.al., 2005).

Another possible cause of error is the purely greedy action selection policy. This could be addressed using a search approach such as beam search.

We reserve to investigate these approaches in future work.

References

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*: Volume 2 - Volume 2 (EMNLP '09), Vol. 2. Association for Computational Linguistics, Stroudsburg, PA, USA, 1007-1016.

G. Attardi, M. Ciaramita. 2007. Tree Revision Learning for Dependency Parsing. In *Proc. of the Human Language Technology Conference 2007.*

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '09).* Association for Computational Linguistics, Stroudsburg, PA, USA, 245-253.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10).* Association for Computational Linguistics, Stroudsburg, PA, USA, 376-384.

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL-44).* Association for Computational Linguistics, Stroudsburg, PA, USA, 529-536.

Alexandra , Miles Osborne, and Philipp Koehn. 2008. Predicting success in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08).* Association for Computational Linguistics, Stroudsburg, PA, USA, 745-754.

	BLEU	BLEU (11b)	BLEU-cased	BLEU-cased (11b)	TER
de-en baseline	**18.8**	**18.8**	**17.8**	**17.8**	**0.722**
de-en reordered	18.1	18.1	17.3	17.3	0.739

Table 3: WMT2013 de-en translation scores

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)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 177-180.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head finalization: a simple reordering rule for SOV languages. *In Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR (WMT '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 244-251.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 848-856.

Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan, and Jiri Navratil. 2011. A word reordering model for improved machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 486-496.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X '06)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 166-170.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th international conference on Computational Linguistics (COLING '04)*. Association for Computational Linguistics, Stroudsburg, PA, USA, , Article 64 .

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 440-447.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 3, Article 27 (May 2011), 27 pages.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.* 9 (June 2008), 1871-1874.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *MT Summit 2005.*

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Toma Erjavec, Dan Tufiş. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006).*

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10 (EMNLP '02)*, Vol. 10. Association for Computational Linguistics, Stroudsburg, PA, USA, 1-8.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 91-98.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.* 3 (March 2003), 951-991.