Literality Based Sample Sorting for Syntax Projection

Bruno Cavestro and Nicola Cancedda
Xerox Research Centre Europe
6, chemin de Maupertuis 38240
Meylan, France
cavestro@xrce.xerox.com
cancedda@xrce.xerox.com

Abstract

We consider the problem of projecting syntax trees across different sides of a parallel corpus, without using any language dependent feature. To achieve this task we introduce a literality score and use it to sort the bi-sentences of the parallel corpus in different classes. We show how to iteratively train a parser over those classes.

1 Introduction

In the last decade we have assisted to an increasing interest on the task of automatically building linguistic resources and tools via parallel corpora. Since the introduction of SITGs (Wu, 1995), quite a few efforts have been done in this area: (Yarowsky and Ngai, 2001; Yarowsky et al., 2001) presented a method to build part-of-speech taggers and morphological analyzers projecting information from one language to the other. Two years later (Rebecca Hwa et al., 2003) introduced the Direct Correspondence Assumption and presented some experiments in the projection of syntactic dependencies. Nevertheless the problem is not solved yet.

In the present paper we consider the problem of projecting syntax trees over two languages, through a language independent process, where, for the target side language, only a POS-tagger is needed. Our approach is centered around a literality score $\mathcal{L}$: we use this weighting function to order bi-sentences from our parallel corpus and to partition the corpus accordingly. We then iteratively train the parser over a class and parse with it the susbequent one.

We believe in fact that when a literal translation of a sentence occurs in a parallel corpus, the projection of information is more robust and complete.

2 Method

We will give a brief description of the full process and then devote one separate paragraph to each step:

Preliminary Step: Data annotation; this step consists on annotating automatically both sides of the corpus. The source side corpus is annotated with both syntactic and dependency trees. The target side is annotated with POS-tags. Both task are done without any human supervision. Furthermore a subset of the parallel corpus is manually annotated with a literality score.

First Step: Alignment; 1-to-1 word alignments are identified automatically.

Second Step: Projection; according to the alignments identified in the previous step, relying on the available annotations, partial dependency trees are built on the target side.

Third Step: Scoring Function Regression; according to the projected information, to a set of features evaluated over that information and to the manual annotations available for the literality score, the parameters of the scoring function are learned

Fourth Step: Bi-sentence scoring; each bi-sentence is scored via the function obtained in the previous step. On the basis of this score, the sententences are sorted and the corpus is partitioned in classes. We expect the first class, the one with highest scored sentences, to contain fully annotated data.

Fifth Step: Parser Training; the first class is adopted as the training set of a parser; no attempt

to clean the data has been done yet.

Sixth Step: Iteration; the parser trained at the previous step is employed to parse the subsequent class; the result of the analisys of the class are compared with the data available by projection, and a cleaning process takes place; the treebank resulting from this cleaning step is added to the previously available training corpus: in such a way, we have enriched our starting training corpus, initialy inherited from the partitioning of corpus through the scored projected trees. We then iterate over all available classes, obtaining at each step a better parser.

At the end of the last iteration we obtain a parser trained over several thousand sentences. From now on we will refer to this process as the $project\ and\ merge$ iterative algorithm

2.1 Data annotation

A preliminary step is required, It consists on three different type of annotations: the first one concerns just the source side which has to be automatically parsed. We used, for this task, Bikel's [1](Bikel, 2004) implementation of Collins' parser: it was configured to output both syntactic and dependency trees. The second annotation concerns the target side which has to be POS-tagged: we used the Xerox Incremental Parser (Roux, 1999) for this task. The third annotation required, concerns a small part of the parallel corpus which has to be manually scored: a small amount of bi-sentences has to be scored with respect to literality of translation.

2.1.1 Manual annotation

We asked, for this task, to volunteers, with a generic knowledge of the concerned languages, to give a score from 1 to 5 to a set of bi-sentences, according to the following brief indications:

- score 5: Word-to-word translation, 1 English word for 1 French word,
- score 4: Articles, prepositions or adjectives missing/added,
- score 3: Minor differences in the structure like auxiliary missing/added or active sentences that become passive,
- score 2: Important differences in the structure,
- score 1: Free translation like in proverbs.

Annotators were otherwise allowed to rely on their subjective interpretation of what literality is. Each sentence was scored up to three times, to have the possibility to evaluate annotations agreement.

2.2 Alignment

The projection of syntactic information heavily relies on the accuracy of the alignment at the word level. We took a conservative position by restricting ourselves to 1-to-1 word alignments. In our experiments, we obtained such alignments by running GIZA++ (Och et al., 1999) to train IBM model 4 in both directions. The alignments obtained as a by-product were then intersected. As model 4 allows in general n-to-1 word alignments, with at most one target word aligned to each source word, the corresponding intersection is guaranteed to contain only 1-to-1 word alignments.

2.3 Projection

The projection method is done according to Hwa's Direct Correspondence Assumption (Rebecca Hwa et al., 2002):

> Given a pair of sentences E and F that are (literal) translation of each other with syntactic structures $Tree_E$ and $Tree_F$, if nodes x_E and y_E of $Tree_E$ are aligned with nodes x_F and y_F of $Tree_F$, respectively, and if syntactic relationship $R(x_E, y_E)$ holds in $Tree_E$, then $R(x_F, y_F)$ holds in $Tree_F$.

where R is in our case just $head$.

Nevertheless we apply the projection step in a peculiar way: instead of beeing a real projection step, we do a substitution one. Given a source dependency tree of the source side, we just substitute the source side words with the target side one, preserving the source side trees. We will obtain a swiss-cheese dependency tree (an example in Fig. 2.3): some nodes of the tree will be associated to the NULL word, and some target words will be excluded from the tree.

[1]Developed at the University of Pennsylvania by Dan Bikel; available from http://www.cis.upenn.edu/~dbikel/software.html

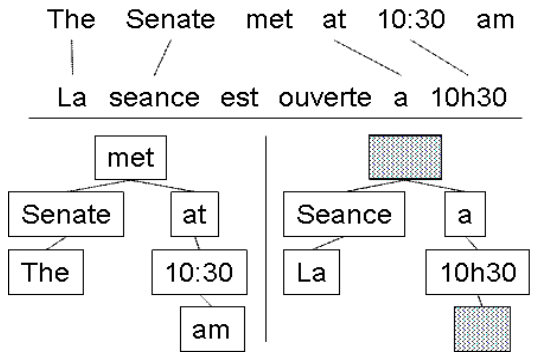

Figure 1: According to the alignment, the source side words of the source side tree are substituted with the corresponding target side words to obtain the target side tree

2.4 Scoring Function Regression

The intuition behind this work is that syntactic information can be projected more effectively when two parallel sentences are one the literal translation of the other. To turn this intuition into an operational ranking criterion, however, we need to identify an appropriate function based on the linguistic knowledge of which we can dispose. Rather than identifying a small number of features and crafting a literality function by hand, we decided to follow a data-driven approach: we identified a relatively large number of features and proceeded to fit a linear combination to the judgments provided by human annotators through regression. We thus seek a literality score function in the form $\mathcal{L} = \lambda \cdot \mathbf{f}$, where $\mathbf{f}$ is a vector of language-independent features and λ is a vector of weights.

As a number of phenomena occuring in trees can be indicative of a more or less literal translation, for each bi-sentence:

- we consider the fraction of source side words and target side words aligned and not aligned, with respect to the length of the sentence;
- we consider the average difference between the maximum translation probability for any word in the translation and the translation probability of the actually aligned word;
- we take into consideration the type/token ratio in both the source side sentece and the target sentence: IBM models get more easily confused when a word appears more than once in a sentence;
- we evaluate the average deviation across POS-tags *k* between the number of null-aligned source tokens with pos *k* and its expected value, estimated as the ratio between sentence length and the number of possible parts of speech. An equivalent feature is applied to target tokens;
- for each possible couples of pos $(pos_k, pos_{k'})$, we consider the fraction of words in the source side sentence with tag pos_k aligned to a word with tag $pos_{k'}$ in the target side sentence with respect to the number of alignments over the bisentence;
- we consider then the average relative depth of each empty node in the swiss cheese dependency tree with respect to the branch it belongs to;
- and finally, we consider how many left children of a node become right ones; we weight this with respect to the relative depth of the node and divide by the number of tree nodes. An equivalent feature is applied to right children becoming left ones.

These features are computed for each bi-sentence annotated by projection.

A linear regression method is applied to determine the coefficients λ to be used for the scoring function $\mathcal{L} = \sum \lambda_i f_i$ where f denotes the features. As the number of features is relatively large (quadratic in the size of the POS tagsets), an appropriately penalised method is required to avoid overfitting. In our experiments we used the Lasso algorithm implemented in the R package "LARS"[2] (Efron et al., 2002). The Lasso algorithm is similar to Least Squares regression, but penalises the objective function by a term proportional to the sum of the absolute values of the coefficients.

2.5 Bi-sentences scoring

Once we have obtained the coefficients of our linear combination, we use the weighting function $\mathcal{L}$

[2] Available from http://www-stat.stanford.edu/ hastie/Papers/LARS/.

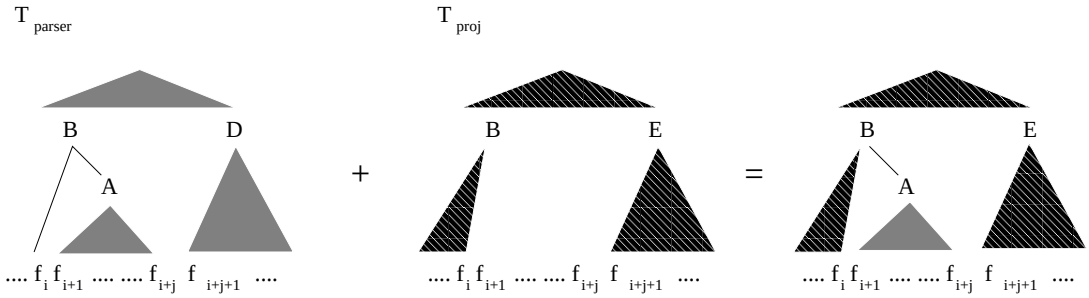

Figure 2: An example of application of the merging operator when a subtree from the tree generated by the trained parser can be appended into the tree obtained by projection. A similar operation is done when the subtree can be attached to its right.

to evaluate the literality of the bi-sentences belonging to the parallel corpus and decompose the target side corpus into several classes. We expect the class corresponding to highly scored sentences to contain bi-sentences obtained by word-by-word translation. We expect those sentences to be more easily alignable and the corresponding syntactic annotations to be more reliably projected. On the contrary, the lower the score, the more holes we expect to have in our tree, and errors in the alignments.

2.6 Parser Training

Given the partition of the bi-sentences based on the Literality Score $\mathcal{L}$, the first class, which includes only bi-sentences scoring 5 or more, is selected. The syntactic trees - and not the dependency ones- of the sentences belonging to this class are selected to train the parser for the target side language. Once again Bikel's implementation of the Collins' parser is used.

2.7 Iteration

At this point an iterative step takes place: the subsequent class, including bi-sentences with scores larger than 4, is parsed with the current parser. Given the guidelines that were provided to the annotators, we expect that projected dependency trees in this class will present empty leaves, but major structural differences will be infrequent.

The fully annotated dependency trees produced by the parser are compared to the swiss cheese dependency trees built by projection: the result is a new set of fully annotated syntax trees to add to the previous training corpus. A new parser is trained over this corpus, and the procedure is repeated for all sentence classes.

The *merge* operation that takes a swiss-cheese tree T_{proj} obtained by projection and the tree T_{parse} obtained through parsing is asymmetric, and relies more heavily on the former. Whenever a span of words $f_{i+1} \dots f_{i+j}$ is not covered by any nonterminal in T_{proj}, T_{parse} is checked to see whether there is a single nonterminal node A covering it. If this is not the case, then the bi-sentence is discarded. If it is the case, and if the parent of A spans $f_i \dots f_{i+j}$ and has the same grammatical label as the node B spanning f_i in T_{proj}, then the subtree rooted in A is appended under B (Fig. 2). Otherwise, if the parent of A spans $f_{i+1} \dots f_{i+j+1}$ and has the same grammatical label as the node C spanning f_{i+j+1} in T_{proj}, then the subtree rooted in A is appended under C. If none of this events happens, then the bi-sentence is discarded.

3 Experiment

We used approx 800.000 sentences of the Canadian Hansard parallel corpus for our experiment. Source side language was English, and target side was French.

The manual annotation was done over 918 sentences. It was asked to annotators to give a score between 1 and 5: 5 being perfectly literal. We collected in the average 2.2 annotations per sentence. The overall annotator agreement was above 75%: given a sentence S, annotated three times, with scores, f_1, f_2 and f_3, we consider those annotations to agree if at worst one of the scores differs from the others and if the gap is equal to ± 1. Annotators where volun-

teers fluent in both languages; no special linguistic background were required, just fluency in both languages.

The parser to be trained is the Bikel's implementation of the Collins parser. The same parser, trained over the standard Wall Street Journal corpus, has been used to label the English side of the Canadian Hansard corpus. GIZA++ was used for the Step 1 of the $project\ and\ merge$ iterative algorithm: the tool was modified to handle all sentences of the corpus, no matter their length and fertility ratio. In order to obtain 1-1 alignment we ran GIZA++ on both direction, english to french and french to english, and we took the intersection of the alignments. The second step, the projection of trees, was realised according to the description in paragraph 2.3 . For the third step, the implementation of the features as described in paragraph 2.4 yielded an initial set of 792 features; more than 50% of the features were originated from the 5th feature subset, related to POS-tags. Given the resulting features, we have trained the LASSO algorithm obtaining the best-matching model. We have then used it to score all the sentences: the scores distribution resulting from this 4th step is depicted in Fig. 3.

A slight modification had to be applied to the 5th step: despite we expected sentences with score equal or larger than 5 to be fully annotated, only 258 out of 1663 happened to be fully complete trees. In order to recover as much information as possible, we decided to parse the incomplete sentences with the Bikel's parser trained over the complete 258 sentences. Given the restricted size of the training set, the parser wasn't able to analyse many sentence, essentially because of lack of vocabulary. The properly parsed sentences were then submitted to the *merge* process and the resulting treebank were then promoted into the training set, to train a new parser. The resulting parser was then adopted to try to recover the sentences discarded by the previous version of the parser. The iterations were repeated until no further sentences were added to the corpus. After two iterations we obtained a 314 trees Treebank. Given this Treebank as a result of the fifth step of the algorithm, the iteration -6th step- over non perfectly literal sentences was performed. Given the observed distribution (Fig. 3 and Fig. 4) we decided to feed the loops with classes of size 1000 of

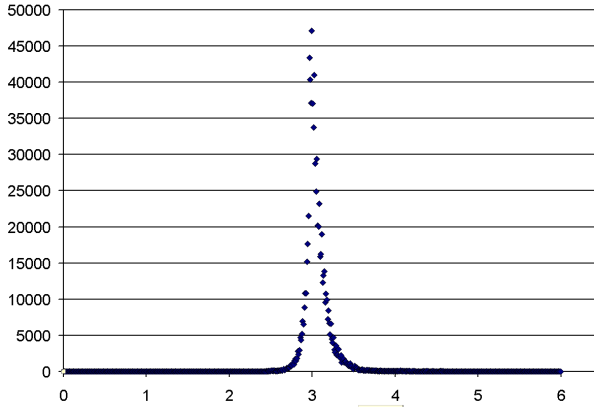

Figure 3: Distribution of Documents According to the Literality Score

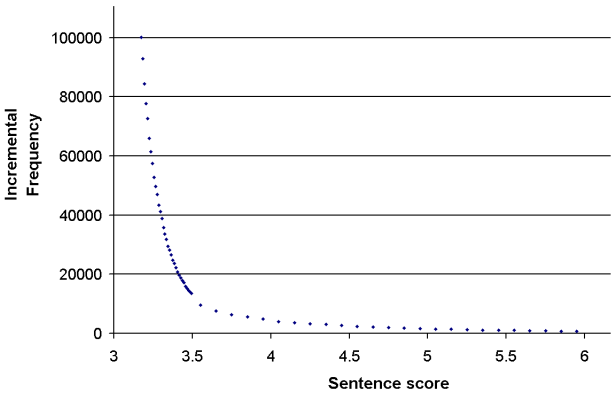

Figure 4: $f(x)$ = Sentences with scores larger than x, with $x > 3.2$

decreasing score.

A simplified version of the *merge* operation was adopted: we decided to limit the merging of trees,to size one spans (Fig. 2), j being fixed to1, in $f_{i+1} \cdots f_{i+j+1}$.

4 Evaluation

4.1 The Literality Score

The quality of the data used to train the Literality Score $\mathcal{L}$ is depicted in Fig. 5 and Fig. 6. The comparison of the two graphs highlight which scores are more affected by annotation disagreement; we recall that we collected 2.2 annotation per document: the number of document scored 5 seems to be the most affected by annotator disagreement.

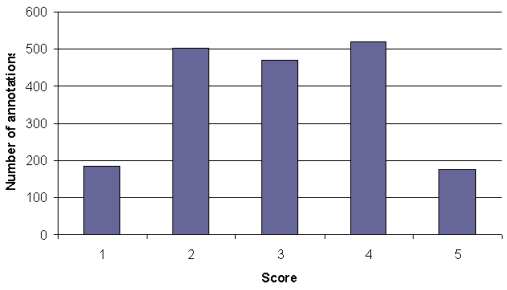

Figure 5: $f(x)$ = Number of document with an annotation x.

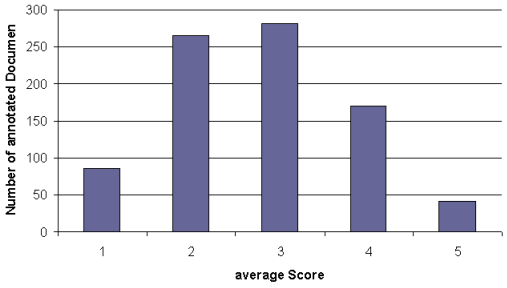

Figure 6: $f(x)$ = Number of document with average annotation equal to x.

```
Score 1.00329:
Nos jeunes doivent franchir de
 nombreux obstacles pour faire
 des etudes superieurs.
There  are  many  barries that
 stand in the way of our youth
 attaining higher education.
Score 5.34012:
Nous pouvons etre tres fier de
 notre pays.
We  can  be  very proud of our
 country.
```

Example 1: Sentences scored with Literality Score $\mathcal{L}$

Figures 3 and 4 depict the distribution of the literality score $\mathcal{L}$, over the 800.000 sentences of the Canadian Hansard corpus. This distribution doesn't respect the distribution observed over the manually annotated data: too many documents get a score around three. Nevertheless Fig. 4 highlights that we are still able to get a corpus large enough using documents with a score slightly above three: we can easily get an intial corpus of hundred thousand sentences. We notice, however, that this is not the size of the corpus used to train the parser, given that during the iterations a number of sentences will be removed. Finally although small the size of the class used to bootstrap the algorithm is still reasonable: in (Rebecca Hwa, 2001) a Collins Parser trained over a comparable subset of the WSJ performs at 75%, using the combined label precision and label recall score as metric.

4.2 The Treebank

Unfortunately, no manually annotated French Treebank was available to us to check performance progress through iterations. In the impossibility of measuring the evolution of the accuracy of the French parsers, we limited our attention to the convergence rate of the parser series trained on the French side relative to the rate of convergence on the English side.

We first isolated a test set S_{test} of X bi-sentences.

Let's call $T_{f,i}$ the training treebank used to train the French parser $p_{f,i}$ at iteration i, and let $T_{e,i}$ be the corresponding treebank on the English side. We train a new English parser $p_{e,i}$ on $T_{e,i}$. Let $T_{e,test,i}$

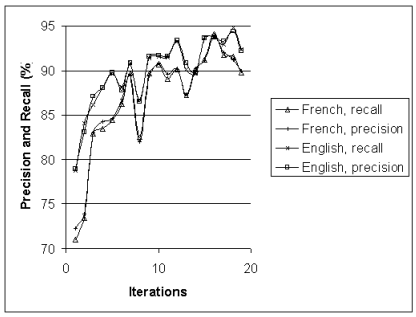

Figure 7: Precision and Recall for both parser

and $T_{f,test,i}$ be the parses resulting from parsing the two sides of S_{test} with $p_{e,i}$ and $p_{f,i}$ respectively.

At each iteration, we score $T_{f,test,i-1}$ and $T_{e,test,i-1}$ against $T_{f,test,i}$ and $T_{e,test,i}$ respectively. If the learning process converges, performance indicators converge to "perfect" values: for large enough i, $T_{f,test,i-1} \approx T_{f,test,i}$ and $T_{e,test,i-1} \approx T_{e,test,i}$. The rapidity of such convergence on the French side compared to the rapidity of convergence on the English side provides an indirect indication of the quality of the French training material obtained by projection and merging. Figure 7 shows the curves we obtained for some of the performance measures output by EVALB [3].

5 Conclusions and further work

As the literality score didn't fully respected our expectations, several tests has still to be done in order to identify which combinations of features set, between those mentioned in paragraph 2.4, yield to better predictions.

In order to fully understand the efficacy of the Literality Score $\mathcal{L}$, we have to compare the performances of the parser trained on a corpus sorted accordingly, with a parser boostrapped in the same way but trained over randomly choosen sentences.

Finally, we still need to compare a parser trained over the corpus we have built with a parser trained over a real Treebank.

6 Acknowledgments

The authors would like to thank the members of the LCA group at XRCE for fruitful discussion. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

[3] Satoshi Sekine and Michael John Collins, available at http://cs.nyu.edu/cs/projects/proteus/evalb

References

Daniel M. Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani, 2002. *A new method for variable subset selection, with the lasso and "epsilon" forward stagewise methods as special cases. LARS Software for R and Splus.* Stanford University.

F. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. EMNLP/WVLC.*

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 2002. Breaking the Resource Bottleneck for Multilingual Parsing. Technical Report LAMP-TR-086,CS-TR-4355,UMIACS-TR-2002-35, University of Maryland, College Park, April.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2003. Evaluating Translational Correspondence using Annotation Projection. Technical Report LAMP-TR-100,CS-TR-4455,UMIACS-TR-2003-25, University of Maryland, College Park, February.

Rebecca Hwa. 2001. On Minimizing Training Corpus for Parser Acquisition. Technical Report LAMP-TR-073,CS-TR-4258,UMIACS-TR-2001-40, University of Maryland, College Park, July.

Claude Roux. 1999. Phrase-driven parser. In *Conference Venezia per il Trattamento Automatico delle lingue (VEXTAL)*, Venezia, Italy, November. Unipress.

Dekai Wu. 1995. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *IJCAI*, pages 1328–1337.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of NAACL-2001*, pages 377–404.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001: First International Conference on Human Language Technology Research*.