An alignment algorithm using belief propagation and a structure-based distortion model

Fabien Cromieres, Sadao Kurohashi

An Alignment Algorithm using Belief Propagation and a Structure-Based Distortion Model

Fabien Cromières
Graduate school of informatics
Kyoto University
Kyoto, Japan
fabien@nlp.kuee.kyoto-u.ac.jp

Sadao Kurohashi
Graduate school of informatics
Kyoto University
Kyoto, Japan
kuro@i.kyoto-u.ac.jp

Abstract

In this paper, we first demonstrate the interest of the Loopy Belief Propagation algorithm to train and use a simple alignment model where the expected marginal values needed for an efficient EM-training are not easily computable. We then improve this model with a distortion model based on structure conservation.

1 Introduction and Related Work

Automatic word alignment of parallel corpora is an important step for data-oriented Machine translation (whether Statistical or Example-Based) as well as for automatic lexicon acquisition. Many algorithms have been proposed in the last twenty years to tackle this problem. One of the most successfull alignment procedure so far seems to be the so-called "IBM model 4" described in (Brown et al., 1993). It involves a very complex distortion model (here and in subsequent usages "distortion" will be a generic term for the reordering of the words occurring in the translation process) with many parameters that make it very complex to train.

By contrast, the first alignment model we are going to propose is fairly simple. But this simplicity will allow us to try and experiment different ideas for making a better use of the sentence structures in the alignment process. This model (and even more so its subsequents variations), although simple, do not have a computationally efficient procedure for an exact EM-based training. However, we will give some theoretical and empirical evidences that Loopy Belief Propagation can give us a good approximation procedure.

Although we do not have the space to review the many alignment systems that have already been proposed, we will shortly refer to works that share some similarities with our approach. In particular, the first alignment model we will present has already been described in (Melamed, 2000). We differ however in the training and decoding procedure we propose. The problem of making use of syntactic trees for alignment (and translation), which is the object of our second alignment model has already received some attention, notably by (Yamada and Knight, 2001) and (Gildea, 2003) .

2 Factor Graphs and Belief Propagation

In this paper, we will make several use of Factor Graphs. A Factor Graph is a graphical model, much like a Bayesian Network. The three most common types of graphical models (Factor Graphs, Bayesian Network and Markov Network) share the same purpose: intuitively, they allow to represent the dependencies among random variables; mathematically, they represent a factorization of the joint probability of these variables.

Formally, a factor graph is a bipartite graph with 2 kinds of nodes. On one side, the Variable Nodes (abbreviated as V-Node from here on), and on the other side, the Factor Nodes (abbreviated as F-Node). If a Factor Graph represents a given joint distribution, there will be one V-Node for every random variable in this joint distribution. Each F-Node is associated with a function of the V-Nodes to which it is connected (more precisely, a function of the values of the random variables associated with the V-Nodes, but for brevity, we will frequently mix the notions of V-Node, Random Variables and their values). The joint distribution is then the product of these functions (and of a normalizing constant). Therefore, each F-Node actually represent a factor in the factorization of the joint distribution.

As a short example, let us consider a problem classically used to introduce Bayesian Network. We want to model the joint probability of the Weather(W) being sunny or rainy, the Sprinkle(S) being on or off, and the Lawn(L) being wet or dry. Figure 1 show the dependencies of

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 166–174,
Athens, Greece, 30 March – 3 April 2009.

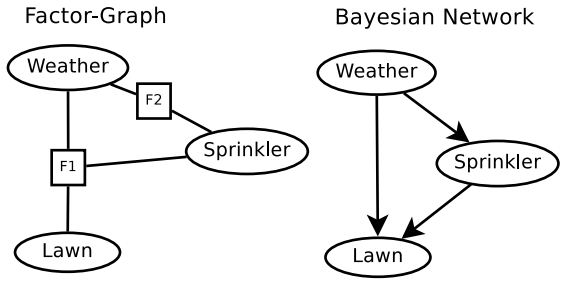

Figure 1: A classical example

the variables represented with a Factor Graph and with a Bayesian Network. Mathematically, the Bayesian Network imply that the joint probability has the following factorization: $P(W, L, S) = P(W) \cdot P(S|W) \cdot P(L|W, S)$. The Factor Graph imply there exist two functions φ_1 and φ_2 as well as a normalization constant C such that we have the factorization: $P(W, L, S) = C \cdot \varphi_2(W, S) \cdot \varphi_1(L, W, S)$. If we set $C = 1$, $\varphi_2(W, S) = P(W) \cdot P(S|W)$ and $\varphi_1(L, W, S) = P(L|W, S)$, the Factor Graph express exactly the same factorization as the Bayesian Network.

A reason to use Graphical Models is that we can use with them an algorithm called Belief Propagation (abbreviated as BP from here on) (Pearl, 1988). The BP algorithm comes in two flavors: sum-product BP and max-product BP. Each one respectively solve two problems that arise often (and are often intractable) in the use of a probabilistic model: "what are the marginal probabilities of each individual variable?" and "what is the set of values with the highest probability?". More precisely, the BP algorithm will give the correct answer to these questions if the graph representing the distribution is a forest. If it is not the case, the BP algorithm is not even guaranteed to converge. It has been shown, however, that the BP algorithm do converge in many practical cases, and that the results it produces are often surprisingly good approximations (see, for example, (Murphy et al., 1999) or (Weiss and Freeman, 2001)).

(Yedidia et al., 2003) gives a very good presentation of the sum-product BP algorithm, as well as some theoretical justifications for its success. We will just give an outline of the algorithm. The BP algorithm is a message-passing algorithm. Messages are sent during several iterations until convergence. At each iteration, each V-Node sends to its neighboring F-Nodes a message representing an estimation of its own marginal values. The message sent by the V-Node V_i to the F-Node F_j estimating the marginal probability of V_i to take the value x is :

$$m_{Vi \to Fj}(x) = \prod_{Fk \in N(Vi) \setminus Fj} m_{Fk \to Vi}(x)$$

(N(Vi) represent the set of the neighbours of V_i)

Also, every F-Node send a message to its neighboring V-Nodes that represent its estimates of the marginal values of the V-Node:

$$m_{Fj \to Vi}(x) = \sum_{v_1, \ldots, v_n} \varphi_j(v_1, .., x, .., v_n) \cdot \cdot \prod_{Vk \in N(Fj) \setminus Vi} m_{Vk \to Fj}(v_k)$$

At any point, the *belief* of a V-Node Vi is given by

$$b_i(x) = \prod_{Fk \in N(Vi)} m_{Fk \to Vi}(x)$$

, b_i being normalized so that $\sum_x b_i(x) = 1$. The belief $b_i(x)$ is expected to converge to the marginal probability (or an approximation of it) of V_i taking the value x .

An interesting point to note is that each message can be "scaled" (that is, multiplied by a constant) by any factor at any point without changing the result of the algorithm. This is very useful both for preventing overflow and underflow during computation, and also sometimes for simplifying the algorithm (we will use this in section 3.2). Also, damping schemes such as the ones proposed in (Murphy et al., 1999) or (Heskes, 2003) are useful for decreasing the cases of non-convergence.

As for the max-product BP, it is best explained as "sum-product BP where each sum is replaced by a maximization".

3 The monolink model

We are now going to present a simple alignment model that will serve both to illustrate the efficiency of the BP algorithm and as basis for further improvement. As previously mentioned, this model is mostly identical to one already proposed in (Melamed, 2000). The training and decoding procedures we propose are however different.

3.1 Description

Following the usual convention, we will designate the two sides of a sentence pair as *French* and *English*. A sentence pair will be noted (e, f). e_i represents the word at position i in e.

In this first simple model, we will pay little attention to the structure of the sentence pair we want to align. Actually, each sentence will be reduced to a bag of words.

Intuitively, the two sides of a sentence pair express the same set of meanings. What we want to do in the alignment process is find the parts of the sentences that originate from the same meaning. We will suppose here that each meaning generate at most one word on each side, and we will name *concept* the pair of words generated by a meaning. It is possible for a meaning to be expressed in only one side of the sentence pair. In that case, we will have a *"one-sided" concept* consisting of only one word. In this view, a sentence pair appears "superficially" as a pair of bag of words, but the bag of words are themselves the visible part of an underlying bag of concepts.

We propose a simple generative model to describe the generation of a sentence pair (or rather, its underlying bag of concepts):

- First, an integer n, representing the number of concepts of the sentence is drawn from a distribution P_{size}
- Then, n concepts are drawn independently from a distribution $P_{concept}$

The probability of a bag of concepts $\mathcal{C}$ is then:

$$P(\mathcal{C}) = P_{size}(|\mathcal{C}|) \prod_{(w_1,w_2)\in\mathcal{C}} P_{concept}((w_1, w_2))$$

We can alternatively represent a bag of concepts as a pair of sentence (e, f), plus an alignment a. a is a set of links, a link being represented as a pair of positions in each side of the sentence pair (the special position -1 indicating the empty side of a one-sided concept). This alternative representation has the advantage of better separating what is observed (the sentence pair) and what is hidden (the alignment). It is not a strictly equivalent representation (it also contains information about the word positions) but this will not be relevant here. The joint distribution of e,f and a is then:

$$P(e, f, a) = P_{size}(|a|) \prod_{(i,j)\in a} P_{concept}(e_i, f_j) \quad (1)$$

This model only take into consideration one-to-one alignments. Therefore, from now on, we will call this model "monolink". Considering only one-to-one alignments can be seen as a limitation compared to others models that can often produce at least one-to-many alignments, but on the good side, this allow the monolink model to be nicely symmetric. Additionally, as already argued in (Melamed, 2000), there are ways to determine the boundaries of some multi-words phrases (Melamed, 2002), allowing to treat several words as a single token. Alternatively, a procedure similar to the one described in (Cromieres, 2006), where substrings instead of single words are aligned (thus considering every segmentation possible) could be used.

With the monolink model, we want to do two things: first, we want to find out good values for the distributions P_{size} and $P_{concept}$. Then we want to be able to find the most likely alignment a given the sentence pair (e, f).

We will consider P_{size} to be a uniform distribution over the integers up to a sufficiently big value (since it is not possible to have a uniform distribution over an infinite discrete set). We will not need to determine the exact value of P_{size} . The assumption that it is uniform is actually enough to "remove" it of the computations that follow.

In order to determine the $P_{concept}$ distribution, we can use an EM procedure. It is easy to show that, at every iteration, the EM procedure will require to set $P_{concept}(w_e, w_f)$ proportional to the sum of the expected counts of the concept (w_e, w_f) over the training corpus. This, in turn, mean we have to compute the conditional expectation:

$$E((i, j) \in a|e, f) = \sum_{a|(i,j)\in a} P(a|e, f)$$

for every sentence pair (e, f). This computation require a sum over all the possible alignments, whose numbers grow exponentially with the size of the sentences. As noted in (Melamed, 2000), it does not seem possible to compute this expectation efficiently with dynamic programming tricks like the one used in the IBM models 1 and 2 (as a passing remark, these "tricks" can actually be seen as instances of the BP algorithm).

We propose to solve this problem by applying the BP algorithm to a Factor Graph representing the conditional distribution $P(a|e, f)$. Given a sentence pair (e, f), we build this graph as follows.

We create a V-node V_i^e for every position i in the English sentence. This V-Node can take for

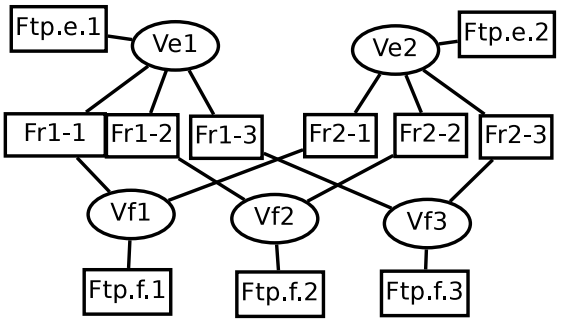

Figure 2: A Factor Graph for the monolink model in the case of a 2-words English sentence and a 3-words french sentence (F_{ij}^{rec} nodes are noted Fri-j)

value any position in the french sentence, or the special position -1 (meaning this position is not aligned, corresponding to a one-sided concept). We create symmetrically a V-node V_j^f for every position in the french sentence.

We have to enforce a "reciprocal love" condition: if a V-Node at position i choose a position j on the opposite side, the opposite V-Node at position j must choose the position i. This is done by adding a F-Node $F_{i,j}^{rec}$ between every opposite node V_i^e and V_j^f, associated with the function:

$$\varphi_{i,j}^{rec}(k,l) = \begin{cases} 1 & \text{if } (i = l \text{ and } j = k) \\ & \text{or } (i \neq l \text{ and } j \neq k) \\ 0 & \text{else} \end{cases}$$

We then connect a "translation probability" F-Node $F_i^{tp.e}$ to every V-Node V_i^e associated with the function:

$$\varphi_i^{tp.e}(j) = \begin{cases} \sqrt{P_{concept}(e_i, f_j)} & \text{if } j \neq -1 \\ P_{concept}(e_i, \emptyset) & \text{if } j = -1 \end{cases}$$

We add symmetrically on the French side F-Nodes $F_j^{tp.f}$ to the V-Nodes V_j^f.

It should be fairly easy to see that such a Factor Graph represents $P(a|e,f)$. See figure 2 for an example.

Using the sum-product BP, the beliefs of every V-Node V_i^e to take the value j and of every node V_j^f to take the value i should converge to the marginal expectation $E((i,j) \in a|e,f)$ (or rather, a hopefully good approximation of it).

We can also use max-product BP on the same graph to decode the most likely alignment. In the monolink case, decoding is actually an instance of the "assignment problem", for which efficient algorithms are known. However this will not be the case for the more complex model of the next section. Actually, (Bayati et al., 2005) has recently proved that max-product BP always give the optimal solution to the assignment problem.

3.2 Efficient BP iterations

Applying naively the BP algorithm would lead us to a complexity of $O(|e|^2 \cdot |f|^2)$ per BP iteration. While this is not intractable, it could turn out to be a bit slow. Fortunately, we found it is possible to reduce this complexity to $O(|e| \cdot |f|)$ by making two useful observations.

Let us note m_{ij}^e the resulting message from V_i^e to V_j^f (that is the message sent by $F_{i,j}^{rec}$ to V_j^f after it received its own message from V_i^e). $m_{ij}^e(x)$ has the same value for every x different from i: $m_{ij}^e(x \neq i) = \sum_{k \neq j} \frac{b_i^e(k)}{m_{ji}^f(k)}$. We can divide all the messages m_{ij}^e by $m_{ij}^e(x \neq i)$, so that $m_{ij}^e(x) = 1$ except if $x = i$; and the same can be done for the messages coming from the French side m_{ij}^f. It follows that $m_{ij}^e(x \neq i) = \sum_{k \neq j} b_i^e(k) = 1 - b_i^e(j)$ if the b_i^e are kept normalized. Therefore, at every step, we only need to compute $m_{ij}^e(j)$, not $m_{ij}^e(x \neq j)$.

Hence the following algorithm ($m_{ij}^e(j)$ will be here abbreviated to m_{ij}^e since it is the only value of the message we need to compute). We describe the process for computing the English-side messages and beliefs (m_{ij}^e and b_i^e) , but the process must also be done symmetrically for the French-side messages and beliefs (m_{ij}^f and b_i^f) at every iteration.

0- Initialize all messages and beliefs with: $m_{ij}^{e(0)} = 1$ and $b_i^{e(0)}(j) = \varphi_i^{tp.e}(j)$

Until convergence (or for a set number of iteration):

1- Compute the messages m_{ij}^e: $m_{ij}^{e(t+1)} = b_i^{e(t)}(j)/((1 - b_i^{e(t)}(j)) \cdot m_{ji}^{f(t)})$

2- Compute the beliefs $b_i^e(j)$:$b_i(j)^{e(t+1)} = \varphi_i^{tp.e}(j) \cdot m_{ji}^{f(t+1)}$

3- And then normalize the $b_i(j)^{e(t+1)}$ so that $\sum_j b_i(j)^{e(t+1)} = 1$.

A similar algorithm can be found for the max-product BP.

3.3 Experimental Results

We evaluated the monolink algorithm with two languages pairs: French-English and Japanese-English.

For the English-French Pair, we used 200,000 sentence pairs extracted from the Hansard corpus (Germann, 2001). Evaluation was done with the scripts and gold standard provided during the workshop HLT-NAACL 2003[1] (Mihalcea and Pedersen, 2003). Null links are not considered for the evaluation.

For the English-Japanese evaluation, we used 100,000 sentence pairs extracted from a corpus of English/Japanese news. We used 1000 sentence pairs extracted from pre-aligned data(Utiyama and Isahara, 2003) as a gold standard. We segmented all the Japanese data with the automatic segmenter Juman (Kurohashi and Nagao, 1994). There is a caveat to this evaluation, though. The reason is that the segmentation and alignment scheme used in our gold standard is not very fine-grained: mostly, big chunks of the Japanese sentence covering several words are aligned to big chunks of the English sentence. For the evaluation, we had to consider that when two chunks are aligned, there is a link between every pair of words belonging to each chunk. A consequence is that our gold standard will contain a lot more links than it should, some of them not relevants. This means that the recall will be largely underestimated and the precision will be overestimated.

For the BP/EM training, we used 10 BP iterations for each sentences, and 5 global EM iterations. By using a damping scheme for the BP algorithm, we never observed a problem of non-convergence (such problems do commonly appears without damping). With our python/C implementation, training time approximated 1 hour. But with a better implementation, it should be possible to reduce this time to something comparable to the model 1 training time with Giza++.

For the decoding, although the max-product BP should be the algorithm of choice, we found we could obtain slightly better results (by between 1 and 2 AER points) by using the sum-product BP, choosing links with high beliefs, and cutting-off links with very small beliefs (the cut-off was chosen roughly by manually looking at a few aligned sentences not used in the evaluation, so as not to create too much bias).

Due to space constraints, all of the results of this section and the next one are summarized in two tables (tables 1 and 2) at the end of this paper.

In order to compare the efficiency of the BP training procedure to a more simple one, we reimplemented the Competitive Link Algorithm (abbreviated as CLA from here on) that is used in (Melamed, 2000) to train an identical model. This algorithm starts with some relatively good estimates found by computing correlation score (we used the G-test score) between words based on their number of co-occurrences. A greedy Viterbi training is then applied to improve this initial guess. In contrast, our BP/EM training do not need to compute correlation scores and start the training with uniform parameters.

We only evaluated the CLA on the French/English pair. The first iteration of CLA did improve alignment quality, but subsequent ones decreased it. The reported score for CLA is therefore the one obtained during the best iteration. The BP/EM training demonstrate a clear superiority over the CLA here, since it produce almost 7 points of AER improvement over CLA.

In order to have a comparison with a well-known and state-of-the-art system, we also used the GIZA++ program (Och and Ney, 1999) to align the same data. We tried alignments in both direction and provide the results for the direction that gave the best results. The settings used were the ones used by the training scripts of the Moses system[2], which we assumed to be fairly optimal. We tried alignment with the default Moses settings (5 iterations of model 1, 5 of Hmm, 3 of model 3, 3 of model 4) and also tried with increased number of iterations for each model (up to 10 per model).

We are aware that the score we obtained for model 4 in English-French is slightly worse than what is usually reported for a similar size of training data. At the time of this paper, we did not have the time to investigate if it is a problem of non-optimal settings in GIZA++, or if the training data we used was "difficult to learn from" (it is common to extract sentences of moderate length for the training data but we didn't, and some sentences of our training corpus do have more than 200 words; also, we did not use any kind of preprocessing). In any case, Giza++ is compared here with an algorithm trained on the same data and with no possibilities for fine-tuning; therefore the comparison should be fair.

The comparison show that performance-wise, the monolink algorithm is between the model 2 and the model 3 for English/French. Considering

[1] http://www.cs.unt.edu/ rada/wpt/

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

our model has the same number of parameters as the model 1 (namely, the word translation probabilities, or concept probabilities in our model), these are pretty good results. Overall, the monolink model tend to give better precision and worse recall than the Giza++ models, which was to be expected given the different type of alignments produced (1-to-1 and 1-to-many).

For English/Japanese, monolink is at just about the level of model 1, but model 1,2 and 3 have very close performances for this language pair (interestingly, this is different from the English/French pair). Incidentally, these performances are very poor. Recall was expected to be low, due to the previously mentioned problem with the gold standard. But precision was expected to be better. It could be the algorithms are confused by the very fine-grained segmentation produced by Juman.

4 Adding distortion through structure

4.1 Description

While the simple monolink model gives interesting results, it is somehow limited in that it do not use any model of distortion. We will now try to add a distortion model; however, rather than directly modeling the movement of the positions of the words, as is the case in the IBM models, we will try to design a distortion model based on the structures of the sentences. In particular, we are interested in using the trees produced by syntactic parsers.

The intuition we want to use is that, much like there is a kind of "lexical conservation" in the translation process, meaning that a word on one side has usually an equivalent on the other side, there should also be a kind of "structure conservation", with most structures on one side having an equivalent on the other.

Before going further, we should precise the idea of "structure" we are going to use. As we said, our prime (but not only) interest will be to make use of the syntactic trees of the sentences to be aligned. However these kind of trees come in very different shapes depending on the language and the type of parser used (dependency, constituents,...). This is why we decided the only information we would keep from a syntactic tree is the set of its sub-nodes. More specifically, for every sub-node, we will only consider the set of positions it cover in the underlying sentence. We will call such a set of positions a P-set. This simplification will allow us to process dependency trees, constituents trees and other structures in a uniformized way. Figure 3 gives an example of a constituents tree and the P-sets it generates.

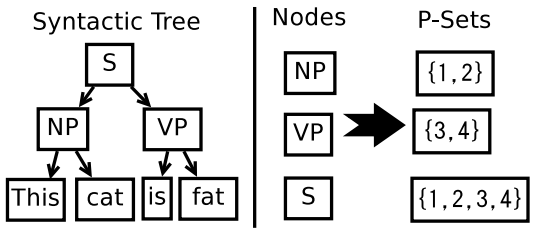

Figure 3: A small syntactic tree and the 3 P-Sets it generates

According to our intuition about the "conservation of structure", some (not all) of the P-sets on one side should have an equivalent on the other side. We can model this in a way similar to how we represented equivalence between words with *concepts*. We postulate that, in addition to a bag of concepts, sentence pairs are underlaid by a set of *P-concepts*. *P-concepts* being actually pairs of P-sets (a P-set for each side of the sentence pair). We also allow the existence of one-sided P-concepts.

In the previous model, sentence pairs where just bag of words underlaid by a or bag of concepts, and there was no modeling of the position of the words. P-concepts bring a notion of word position to the model. Intuitively, there should be coherency between P-concepts and concepts. This coherence will come from a *compatibility constraint*: if a sentence contains a two-sided P-concept (PS_e, PS_f), and if a word w_e covered by PS_e come from a two-sided concept (w_e, w_f), then w_f must be covered by PS_f.

Let us describe the model more formally. In the view of this model, a sentence pair is fully described by: e and f (the sentences themselves), a (the word alignment giving us the underlying bag of concept), s^e and s^f (the sets of P-sets on each side of the sentence) and a_s (the P-set alignment that give us the underlying set of P-concepts). e,f,s^e,s^f are considered to be observed (even if we will need parsing tools to observe s^e and s^f); a and a_s are hidden. The probability of a sentence pair is given by the joint probability of these variables :$P(e, f, s^e, s^f, a, a_s)$. By making some simple independence assumptions, we can write:

$$P(a, a_s, e, f, s^e, s^f) = P_{ml}(a, e, f) \cdot \cdot P(s^e, s^f|e, f) \cdot P(a_s|a, s^e, s^f)$$

$P_{ml}(a, e, f)$ is taken to be identical to the monolink model (see equation (1)). We are not interested in $P(s^e, s^f|e, f)$ (parsers will deal with it for us). In our model, $P(a_s|a, s^e, s^f)$ will be equal to:

$$P_(a_s|a, s^e, s^f) = C \cdot \prod_{(i,j)\in a_s} P_{pc}(s_i^e, s_j^f) \cdot \cdot comp(a, a_s, s^e, s^f)$$

where $comp(a, a_s, s^e, s^f)$ is equal to 1 if the *compatibility constraint* is verified, and 0 else. C is a normalizing constant. P_{pc} describe the probability of each P-concept.

Although it would be possible to learn parameters for the distribution P_{pc} depending on the characteristics of each P-concepts, we want to keep our model simple. Therefore, P_{pc} will have only two different values. One for the one-sided P-concepts, and one for the two-sided ones. Considering the constraint of normalization, we then have actually one parameter: $\alpha = \frac{P_{pc}(1-sided)}{P_{pc}(2-sided)}$. Although it would be possible to learn the parameter α during the EM-training, we choose to set it at a preset value. Intuitively, we should have $0 < \alpha < 1$, because if α is greater than 1, then the one-sided P-concepts will be favored by the model, which is not what we want. Some empirical experiments showed that all values of α in the range [0.5,0.9] were giving good results, which lead to think that α can be set mostly independently from the training corpus.

We still need to train the concepts probabilities (used in $P_{ml}(a, e, f)$), and to be able to decode the most probable alignments. This is why we are again going to represent $P(a, a_s|e, f, s_e, s_f)$ as a Factor Graph.

This Factor Graph will contain two instances of the monolink Factor Graph as subgraph: one for a, the other for a_s (see figure 4). More precisely, we create again a V-Node for every position on each side of the sentence pair. We will call these V-Nodes "Word V-Nodes", to differentiate them from the new "P-set V-Nodes". We will create a "P-set V-Node" $V_i^{ps.e}$ for every P-set in s_e, and a "P-set V-Node" $V_j^{ps.f}$ for every P-set in s_j. We inter-connect all of the Word V-Nodes so that we have a subgraph identical to the Factor Graph used in the monolink case. We also create a "monolink subgraph" for the P-set V-Nodes.

We now have 2 disconnected subgraphs. However, we need to add F-Nodes between them to enforce the *compatibility constraint* between a_s and

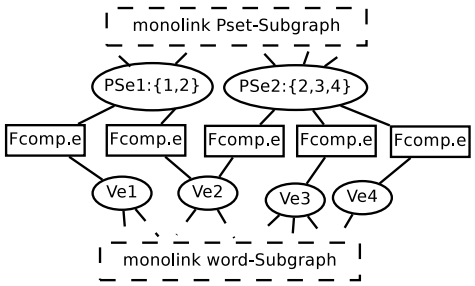

Figure 4: A part of a Factor Graph showing the connections between P-set V-Nodes and Word V-Nodes on the English side.The V-Nodes are connected to the French side through the 2 monolink subgraphs

a. On the English side, for every P-set V-Node V_k^{pse}, and for every position i that the corresponding P-set cover, we add a F-Node $F_{k,i}^{comp.e}$ between V_k^{pse} and V_i^e, associated with the function:

$$\varphi_{k,i}^{comp.e}(l, j) = \begin{cases} 1 & \text{if } j \in s_l^f \text{ or} \\ & j = -1 \text{ or } l = -1 \\ 0 & \text{else} \end{cases}$$

We proceed symmetrically on the French side.

Messages inside each monolink subgraph can still be computed with the efficient procedure described in section 3.2. We do not have the space to describe in details the messages sent between P-set V-Nodes and Word V-Nodes, but they are easily computed from the principles of the BP algorithm. Let $N_E = \sum_{ps\in s^e} |ps|$ and $N_F = \sum_{ps\in s^f} |ps|$. Then the complexity of one BP iteration will be $O(N_G \cdot N_D + |e| \cdot |f|)$.

An interesting aspect of this model is that it is flexible towards enforcing the respect of the structures by the alignment, since not every P-set need to have an equivalent in the opposite sentence. (Gildea, 2003) has shown that too strict an enforcement can easily degrade alignment quality and that good balance was difficult to find.

Another interesting aspect is the fact that we have a somehow "parameterless" distortion model. There is only one real-valued parameter to control the distortion: α. And even this parameter is actually pre-set before any training on real data. The distortion is therefore totally controlled by the two sets of P-sets on each side of the sentence.

Finally, although we introduced the P-sets as being generated from a syntactic tree, they do not need to. In particular, we found interesting to use P-sets consisting of every pair of adja-

cent positions in a sentence. For example, with a sentence of length 5, we generate the P-sets {1,2},{2,3},{3,4} and {4,5}. The underlying intuition is that "adjacency" is often preserved in translation (we can see this as another case of "conservation of structure"). Practically, using P-sets of adjacent positions create a distortion model where permutation of words are not penalized, but gaps are penalized.

4.2 Experimental Results

The evaluation setting is the same as in the previous section. We created syntactic trees for every sentences. For English,we used the Dan Bikel implementation of the Collins parser (Collins, 2003). For French, the SYGMART parser (Chauché, 1984) and for Japanese, the KNP parser (Kurohashi and Nagao, 1994).

The line *SDM:Parsing* (SDM standing for "Structure-based Distortion Monolink") shows the results obtained by using P-sets from the trees produced by these parsers. The line *SDM:Adjacency* shows results obtained by using adjacent positions P-sets ,as described at the end of the previous section (therefore, *SDM:Adjacency* do not use any parser).

Several interesting observations can be made from the results. First, our structure-based distortion model did improve the results of the monolink model. There are however some surprising results. In particular, *SDM:Adjacency* produced surprisingly good results. It comes close to the results of the IBM model 4 in both language pairs, while it actually uses exactly the same parameters as model 1. The fact that an assumption as simple as "allow permutations, penalize gaps" can produce results almost on par with the complicated distortion model of model 4 might be an indication that this model is unnecessarily complex for languages with similar structure.Another surprising result is the fact that *SDM:Adjacency* gives better results for the English-French language pair than *SDM:Parsing*, while we expected that information provided by parsers would have been more relevant for the distortion model. It might be an indication that the structure of English and French is so close that knowing it provide only moderate information for word reordering. The contrast with the English-Japanese pair is, in this respect, very interesting. For this language pair, *SDM:Adjacency* did provide a strong improvement, but significantly less so than *SDM:Parsing*. This tend to show that for language pairs that have very different structures, the information provided by syntactic tree is much more relevant.

Algorithm	**AER**	**P**	**R**
Monolink	0.197	0.881	0.731
SDM:Parsing	0.166	0.882	0.813
SDM:Adjacency	0.135	**0.887**	0.851
CLA	0.26	0.819	0.665
GIZA++ /Model 1	0.281	0.667	0.805
GIZA++ /Model 2	0.205	0.754	0.863
GIZA++ /Model 3	0.162	0.806	0.890
GIZA++ /Model 4	**0.121**	0.849	**0.927**

Table 1: Results for English/French

Algorithm	**F**	**P**	**R**
Monolink	0.263	0.594	0.169
SDM:Parsing	0.291	**0.662**	0.186
SDM:Adjacency	0.279	0.636	0.179
GIZA++ /Model 1	0.263	0.555	0.172
GIZA++ /Model 2	0.268	0.566	0.176
GIZA++ /Model 3	0.267	0.589	0.173
GIZA++ /Model 4	**0.299**	0.658	**0.193**

Table 2: Results for Japanese/English.

5 Conclusion and Future Work

We will summarize what we think are the 4 more interesti ng contributions of this paper. BP algorithm has been shown to be useful and flexible for training and decoding complex alignment models. An original mostly non-parametrical distortion model based on a simplified structure of the sentences has been described. Adjacence constraints have been shown to produce very efficient distortion model. Empirical performances differences in the task of aligning Japanese and English to French hint that considering different paradigms depending on language pairs could be an improvement on the "one-size-fits-all" approach generally used in Statistical alignment and translation.

Several interesting improvement could also be made on the model we presented. Especially, a more elaborated P_{pc}, that would take into account the nature of the nodes (NP, VP, head,..) to parametrize the P-set alignment probability, and would use the EM-algorithm to learn those parameters.

References

M. Bayati, D. Shah, and M. Sharma. 2005. Maximum weight matching via max-product belief propagation. *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1763–1767.

Peter E Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer, 1993. *The mathematics of statistical machine translation: parameter estimation*, volume 19, pages 263–311.

J. Chauché. 1984. *Un outil multidimensionnel de lanalyse du discours. Coling84*. Stanford University, California.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*.

Fabien Cromieres. 2006. Sub-sentential alignment using substring co-occurrence counts. In *Proceedings of ACL*. The Association for Computer Linguistics.

U. Germann. 2001. Aligned hansards of the 36th parliament of canada. *http://www.isi.edu/naturallanguage/download/hansard/*.

D. Gildea. 2003. Loosely tree-based alignment for machine translation. *Proceedings of ACL*, 3.

T. Heskes. 2003. Stable fixed points of loopy belief propagation are minima of the bethe free energy. *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*.

S. Kurohashi and M. Nagao. 1994. A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

I. Melamed. 2002. *Empirical Methods for Exploiting Parallel Texts*. The MIT Press.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Alberta, Canada, May 31. Association for Computational Linguistics.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. *In Proceedings of Uncertainty in AI*, pages 467—475.

Franz Josef Och and Hermann Ney. 1999. Improved alignment models for statistical machine translation. *University of Maryland, College Park, MD*, pages 20—28.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers.

M. Utiyama and H. Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 72–79.

Y. Weiss and W. T. Freeman. 2001. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. on Information Theory*, 47(2):736–744.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. *Proceedings of ACL*.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss, 2003. *Understanding belief propagation and its generalizations*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.