Statistical Bistratal Dependency Parsing

Richard Johansson
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy
johansson@disi.unitn.it

Abstract

We present an inexact search algorithm for the problem of predicting a two-layered dependency graph. The algorithm is based on a k-best version of the standard cubic-time search algorithm for projective dependency parsing, which is used as the backbone of a beam search procedure. This allows us to handle the complex non-local feature dependencies occurring in bistratal parsing if we model the interdependency between the two layers.

We apply the algorithm to the syntactic–semantic dependency parsing task of the CoNLL-2008 Shared Task, and we obtain a competitive result equal to the highest published for a system that jointly learns syntactic and semantic structure.

1 Introduction

Numerous linguistic theories assume a multistratal model of linguistic structure, such as a layer of surface syntax, deep syntax, and shallow semantics. Examples include Meaning–Text Theory (Mel'čuk, 1988), Discontinuous Grammar (Buch-Kromann, 2006), Extensible Dependency Grammar (Debusmann et al., 2004), and the Functional Generative Description (Sgall et al., 1986) which forms the theoretical foundation of the Prague Dependency Treebank (Hajič, 1998).

In the statistical NLP community, the most widely used grammatical resource is the Penn Treebank (Marcus et al., 1993). This is a purely syntactic resource, but we can also include this treebank in the category of multistratal resources since the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) projects have annotated shallow semantic structures on top of it. Dependency-converted versions of the Penn Treebank, PropBank and NomBank were used in the CoNLL-2008 Shared Task (Surdeanu et al., 2008), in which the task of the participants was to produce a bistratal dependency structure consisting of surface syntax and shallow semantics.

Producing a consistent multistratal structure is a conceptually and computationally complex task, and most previous methods have employed a purely pipeline-based decomposition of the task. This includes the majority of work on shallow semantic analysis (Gildea and Jurafsky, 2002, *inter alia*). Nevertheless, since it is obvious that syntax and semantics are highly interdependent, it has repeatedly been suggested that the problems of syntactic and semantic analysis should be carried out *simultaneously* rather than in a pipeline, and that modeling the interdependency between syntax and semantics would improve the quality of all the substructures.

The purpose of the CoNLL-2008 Shared Task was to study the feasibility of a joint analysis of syntax and semantics, and while most participating systems used a pipeline-based approach to the problem, there were a number of contributions that attempted to take the interdependence between syntax and semantics into account. The top-performing system in the task (Johansson and Nugues, 2008) applied a very simple reranking scheme by means of a k-best syntactic output, similar to previous attempts (Gildea and Jurafsky, 2002; Toutanova et al., 2005) to improve semantic role labeling performance by using mul-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 561–569,
Singapore, 6-7 August 2009.

tiple parses. The system by Henderson et al. (2008) extended previous stack-based algorithms for dependency parsing by using two separate stacks to build the syntactic and semantic graphs. Lluís and Màrquez (2008) proposed a model that simultaneously predicts syntactic and semantic links, but since its search algorithm could not take the syntactic–semantic interdependencies into account, a pre-parsing step was still needed. In addition, before the CoNLL-2008 shared task there have been a few attempts to jointly learn syntactic and semantic structure; for instance, Merlo and Musillo (2008) appended semantic role labels to the phrase tags in a constituent treebank and applied a conventional constituent parser to predict constituent structure and semantic roles.

In this paper, we propose a new approximate search method for bistratal dependency analysis. The search method is based on a beam search procedure that extends a k-best version of the standard cubic-time search algorithm for projective dependency parsing. This is similar to the search method for constituent parsing used by Huang (2008), who referred to it as *cube pruning*, inspired by an idea from machine translation decoding (Chiang, 2007). The cube pruning approach, which is normally used to solve the $\arg\max$ problem, was also recently extended to summing problems, which is needed in some learning algorithms (Gimpel and Smith, 2009).

We apply the algorithm on the CoNLL-2008 Shared Task data, and obtain the same evaluation score as the best previously published system that simultaneously learns syntactic and semantic structure (Titov et al., 2009).

2 Bistratal Dependency Parsing

In the tradition of dependency representation of sentence structure, starting from Tesnière (1959), the linguistic structure of the sentence is represented as a directed graph of relations between words. In most theories, certain constraints are imposed on this graph; the most common constraint on dependency graphs in syntax, for instance, is that the graph should form a tree (i.e. it should be connected, acyclic, and every node should have at most one incoming edge). This assumption underlies almost all dependency parsing, although there are also a few parsers based on slightly more general problem formulations (Sagae and Tsuji, 2008).

In this paper, we assume a different type of constraint: that the graph can be partitioned into two subgraphs that we will refer to as *strata* or *layers*, where the first of the layers forms a tree. For the second layer, the only assumption we make is that there is at most one link between any two words. However, we believe that for any interesting linguistic structure, the second layer will be highly dependent on the structure of the first layer.

Figure 1 shows an example of a bistratal dependency graph such as in the CoNLL-2008 Shared Task on syntactic and semantic dependency parsing. The figure shows the representation of the sentence *We were expecting prices to fall*. The primary layer represents surface-syntactic relations, shown above the sentence, and the secondary layer consists of predicate–argument links (here, we have two predicates *expecting* and *fall*).

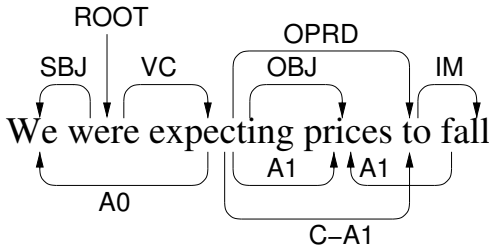

Figure 1: Example of a bistratal dependency graph.

We now give a formal model of the statistical parsing problem of prediction of a bistratal dependency graph. For a given input sentence $\boldsymbol{x}$, the task of our algorithm is to predict a structure $\hat{y}$ consisting of a *primary layer* $\hat{y}_p$ and a *secondary layer* $\hat{y}_s$. In a discriminative modeling framework, we model this prediction problem as the search for the highest-scoring output from the candidate space $\mathcal{Y}$ under a scoring function F:

$$\langle \hat{y}_p, \hat{y}_s \rangle = \arg\max_{\langle y_p, y_s \rangle \in \mathcal{Y}} F(\boldsymbol{x}, y_p, y_s)$$

The learning problem consists of searching in the model space for a scoring function F that minimizes the cost of predictions on unseen examples according to a given *cost function* ρ. In this work, we consider linear scoring functions of the following form:

$$F(\boldsymbol{x}, y_p, y_s) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_p, y_s)$$

where $\boldsymbol{\Phi}(\boldsymbol{x}, y)$ is a numeric feature representation of the tuple $(\boldsymbol{x}, y_p, y_s)$ and $\boldsymbol{w}$ a high-dimensional vector of feature weights.

Based on the structural assumptions made above, we now decompose the feature representation into three parts:

$$\mathbf{\Phi} = \mathbf{\Phi}_p + \mathbf{\Phi}_i + \mathbf{\Phi}_s$$

Here, $\mathbf{\Phi}_p$ represents the primary layer, assumed to be a tree, $\mathbf{\Phi}_s$ the secondary layer, and finally $\mathbf{\Phi}_i$ is the representation of the interdependency between the layers. For the feature representations of the primary and secondary layers, we employ *edge factorization*, a decomposition widely used in statistical dependency parsing, and assume that all edges can be scored independently:

$$\mathbf{\Phi}_p(\boldsymbol{x}, y_p) = \sum_{f \in y_p} \phi_p(\boldsymbol{x}, f)$$

The representation of the interdependency between the layers assumes that each secondary link is dependent on the primary layer, but independent of other secondary links.

$$\mathbf{\Phi}_i(\boldsymbol{x}, y_p, y_s) = \sum_{f \in y_s} \phi_i(\boldsymbol{x}, f, y_p)$$

The interdependency between layers is the bottleneck for the search algorithm that we will present in Section 3. For semantic role analysis, this involves all features that rely on a syntactic representation, most importantly the PATH feature that represents the grammatical relation between predicate and argument words. For instance, in Figure 1, we can represent the surface-syntactic relation between the tokens *fall* and *prices* as the string IM↑OPRD↑OBJ↓. In this work, all interdependency features will be based on paths in the primary layer.

3 A Bistratal Search Algorithm

This section presents an algorithm to approximately solve the $\arg\max$ problem for prediction of bistratal dependency structures. We present the algorithm in two steps: first, we review a k-best version of the standard search algorithm for projective monostratal dependency parsing, based on the work by Huang and Chiang (2005).[1] In the second step, starting from the k-best monostratal search, we devise a search method for the bistratal problem.

[1]Huang and Chiang (2005) described an even more efficient k-best algorithm based on lazy evaluation, which we will not use here since it is not obviously adaptable to the situation where the search is inexact.

3.1 Review of k-Best Dependency Parsing

The search method commonly used in dependency parsers is a chart-based dynamic programming algorithm that finds the highest-scoring projective dependency tree under an edge-factored scoring function. It runs in cubic time with respect to the sentence length. In a slightly more general formulation, it was first published by Eisner (1996). Starting from McDonald et al. (2005), it has been widely used in recent statistical dependency parsing frameworks.

The algorithm works by creating *open structures*, which consist of a dependency link and the set of links that it spans, and *closed structures*, consisting of the left or right half of a complete subtree. An open structure is created by a procedure LINK that adds a dependency link to connect a right-pointing and a left-pointing closed structure, and a closed structure by a procedure JOIN that joins an open structure with a closed structure. Figure 2 shows schematic illustrations: a LINK operation connects the right-pointing closed structure between s and j with the left-pointing closed structure between $j+1$ and e, and a JOIN operation connects an open structure between s and j with a closed structure between j and e.

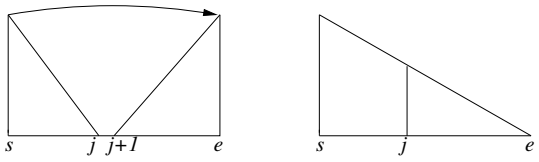

Figure 2: Illustrations of the LINK and JOIN operations.

The search algorithm can easily be extended to find the k best parses, not only the best one. In k-best parsing, we maintain a k-best list in every cell in the dynamic programming table. To create the k-best list of derivations for an open structure between the positions s and e, for instance, there are up to $|L| \cdot (e-s) \cdot k^2$ possible combinations to consider if the set of allowed labels is L. The key observation by Huang and Chiang (2005) is to make use of the fact that the lists are sorted. For every position between s and e, we add the best combination to a priority queue, from which we then repeatedly remove the front item. For every item we remove, we add three successors: an item with a next-best left part, an item with a next-best right part, and finally an item with a next-best edge

label.

The pseudocode of the search algorithm for k-best dependency parsing is given in Algorithms 1 and 2. For brevity, we omitted the code for ADVANCE-LEFT and ADVANCE-RIGHT, which are similar to ADVANCE-EDGE, as well as ADVANCE-LOWER, which resembles ADVANCE-UPPER. The FST function used in the pseudocode returns the first element of a tuple.

The algorithm uses a priority queue with standard operations ENQUEUE, which enqueues an element, and DEQUEUE, which removes the highest-scoring item from the queue. With a standard binary heap implementation of the priority queue, these two operations execute in logarithmic time. To build the queue, we use a constant-time TOSS operation, which appends an item to the queue without enforcing the priority queue constraint, and a HEAPIFY operation that constructs a consistent priority queue in linear time.

3.2 Extension to Bistratal Dependency Parsing

The k-best algorithm forms the core of the inexact bistratal search algorithm. Our method is similar to the forest reranking method by Huang (2008), although there is no forest pruning or reranking involved here. Crucially, we divide the features into *local* features, which can be computed "offline", and *nonlocal* features, which must be computed during search. In our case, the local features are $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$, while the nonlocal features are the interdependent features $\mathbf{\Phi}_i$.

Algorithm 3 shows pseudocode for the main part of the bistratal search algorithm, and Algorithm 4 for its support functions. The algorithm works as follows: for every span $\langle s, e \rangle$, the algorithm first uses the LINK procedure from the k-best monostratal search to construct a k-best list of open structures without semantic links. In the next step, secondary links are added in the procedure LINK-SECONDARY. For brevity, we show only the procedures that create open structures; they are very similar to their closed-structure counterparts.

The LINK-SECONDARY procedure starts by creating an initial candidate (FIRST-SEC-OPEN) based on the best open structure for the primary layer. FIRST-SEC-OPEN creates the candidate space for secondary links for a single primary open structure. To reduce search complexity, it makes use of a problem-specific function SCOPE that defines which secondary links are possible from a given token, given a primary-layer context.

An important insight by Huang (2008) is that nonlocal features should be computed as early as possible during search. In our case, we assume that the interdependency features are based on *tree paths* in the primary layer. This means that secondary links between two tokens can be added when there is a complete path in the primary layer between the tokens. When we create an open

Algorithm 1 k-best search algorithm for dependency parsing.

```
function k-BEST-SEARCH(k)
  n ← length of the sentence
  initialize the table O of open structures
  initialize the table C of closed structures
  for m ∈ [1, ..., n]
    for s ∈ [0, ..., n − m]
      LINK(s, s + m, →, k)
      LINK(s, s + m, ←, k)
      JOIN(s, s + m, →, k)
      JOIN(s, s + m, ←, k)
  return C[0, n, →]

procedure LINK(s, e, dir, k)
  E ← CREATE-EDGES(s, e, dir, k)
  q ← empty priority queue
  for j ∈ [s, ..., e − 1]
    l ← C[s, j, →]
    r ← C[j + 1, e, ←]
    o ← CREATE-OPEN(E, l, r, 1, 1, 1)
    TOSS(q, o)
  HEAPIFY(q)
  while |O[s, e, dir]| < k and |q| > 0
    o ← DEQUEUE(q)
    if o ∉ O[s, e, dir]
      APPEND(O[s, e, dir], o)
      ENQUEUE(q, ADVANCE-EDGE(o))
      ENQUEUE(q, ADVANCE-LEFT(o))
      ENQUEUE(q, ADVANCE-RIGHT(o))

procedure JOIN(s, e, dir, k)
  q ← empty priority queue
  if dir =→
    for j ∈ [s + 1, ..., e]
      u ← O[s, j, →]
      l ← C[j, e, →]
      c ← CREATE-CLOSED(u, l, 1, 1)
      TOSS(q, c)
  else
    for j ∈ [s, ..., e − 1]
      u ← O[j, e, ←]
      l ← C[s, j, ←]
      c ← CREATE-CLOSED(u, l, 1, 1)
      TOSS(q, c)
  HEAPIFY(q)
  while |C[s, e, dir]| < k and |q| > 0
    c ← DEQUEUE(q)
    if c ∉ C[s, e, dir]
      APPEND(C[s, e, dir], c)
      ENQUEUE(q, ADVANCE-UPPER(c))
      ENQUEUE(q, ADVANCE-LOWER(c))
```

Algorithm 2 Support operations for the k-best search.

function CREATE-EDGES(s, e, dir, k)
  $E \leftarrow \emptyset$
  for $l \in$ ALLOWED-LABELS(s, e, dir)
    $score_L \leftarrow \boldsymbol{w} \cdot \phi_p(s, e, dir, l)$
    $edge \leftarrow \langle score_L, s, e, dir, l \rangle$
    APPEND$(E, edge)$
  return the top k edges in E

function CREATE-OPEN(E, l, r, i_e, i_l, i_r)
  $score_L \leftarrow$ FST$(E[i_e])$ + FST$(l[i_l])$ + FST$(r[i_r])$
  return $\langle score_L + score_N, E, l, r, i_e, i_l, i_r \rangle$

function CREATE-CLOSED(u, l, i_u, i_r)
  $score_L \leftarrow$ FST$(u[i_u])$ + FST$(l[i_l])$
  return $\langle score_L + score_N, u, l, i_u, i_l \rangle$

function ADVANCE-EDGE(o)
where $o = (score, E, l, r, i_e, i_l, i_r)$
  if $i_e =$ LENGTH(E)
    return $\emptyset$
  else
    return CREATE-OPEN$(E, l, r, i_e + 1, i_l, i_r)$

function ADVANCE-UPPER(c)
where $c = (u, l, i_u, i_l)$
  if $i_u =$ LENGTH(u)
    return $\emptyset$
  else
    return CREATE-CLOSED$(u, l, i_u + 1, i_l)$

structure by adding a link between two substructures, a complete path is created between the tokens in the substructures. We thus search for possible secondary links only between the two substructures that are joined.

Figure 3 illustrates this process. A primary open structure between s and e has been created by adding a link from the right-pointing closed structure between s and j to the left-pointing closed structure between $j + 1$ and e. We now try to add secondary links between the two substructures. For instance, in the semantic role parsing task described in subsection 3.3, if we know that there is a predicate between s and j, then we look for arguments between $j + 1$ and e, i.e. we apply the SCOPE function to the right substructure.

When computing the scores for secondary links, note that for efficiency only the interdependent part $\boldsymbol{\Phi}_i$ should be computed in CREATE-SECEDGES; the part of the score that does not depend on the primary layer can be computed before entering the search procedure.

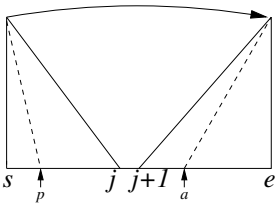

Figure 3: Illustration of the secondary linking process: When two substructures are connected, we can compute the path between a predicate in the left substructure and an argument in the right substructure.

Algorithm 3 Search algorithm for bistratal dependency parsing.

function BISTRATAL-SEARCH(k)
  $n \leftarrow$ length of the sentence
  initialize the table O of open structures
  initialize the table C of closed structures
  using ϕ_s, compute a table $scores_s$ for all
    possible secondary edges $\langle h, d, l \rangle$
  for $m \in [1, \ldots, n]$
    for $s \in [0, \ldots, n - m]$
      LINK$(s, s + m, \rightarrow, k)$
      LINK-SECONDARY$(s, s + m, \rightarrow, k)$
      LINK$(s, s + m, \leftarrow, k)$
      LINK-SECONDARY$(s, s + m, \leftarrow, k)$
      JOIN$(s, s + m, \rightarrow, k)$
      JOIN-SECONDARY$(s, s + m, \rightarrow, k)$
      JOIN$(s, s + m, \leftarrow, k)$
      JOIN-SECONDARY$(s, s + m, \leftarrow, k)$
  return FIRST$(C[0, n, \rightarrow])$

procedure LINK-SECONDARY(s, e, dir, k)
  $q \leftarrow$ empty priority queue
  $o \leftarrow$ FIRST-SEC-OPEN$(O[s, e, dir], 1, k)$
  ENQUEUE(q, o)
  $buf \leftarrow$ empty list
  while $|buf| < k$ and $|q| > 0$
    $o \leftarrow$ DEQUEUE(q)
    if $o \notin buf$
      APPEND(buf, o)
      for $o' \in$ ADVANCE-SEC-OPEN(o, k)
        ENQUEUE(q, o')
  SORT(buf) to $O[s, e, dir]$

3.3 Application on the CoNLL-2008 Shared Task Treebank

We applied the bistratal search method in Algorithm 3 on the data from the CoNLL-2008 Shared Task (Surdeanu et al., 2008). Here, the primary layer is the tree of surface-syntactic relations such as subject and object, and the secondary layer contains the links between the predicate words in the sentence and their respective logical arguments, such as agent and patient. The training corpus consists of sections 02 – 21 of the Penn Treebank, and contains roughly 1 million words.

Algorithm 4 Support operations in bistratal search.

function FIRST-SEC-OPEN(L, i_L, k)
  if $i = \text{LENGTH}(L)$
    return $\emptyset$
  $l \leftarrow$GET-LEFT($L[i_L]$), $r \leftarrow$GET-RIGHT($L[i_L]$)
  for $h \in [\text{START}(l), \ldots, \text{END}(l)]$
    for $d \in \text{SCOPE}(r, h)]$
      $E[h][d] \leftarrow \text{CREATE-SEC-EDGES}(h, d, L[i_L], k)]$
      $I_E[h][d] \leftarrow 1$
  for $h \in [\text{START}(r), \ldots, \text{END}(r)]$
    for $d \in \text{SCOPE}(l, h)]$
      $E[h][d] \leftarrow \text{CREATE-SEC-EDGES}(h, d, L[i_L], k)]$
      $I_E[h][d] \leftarrow 1$
  return CREATE-SEC-OPEN(L, i_L, E, I)

function CREATE-SEC-EDGES(h, d, o, k)
  $E \leftarrow \emptyset$
  for $l \in$ ALLOWED-SEC-LABELS(h, d)
    $score \leftarrow \boldsymbol{w} \cdot \phi_i(h, d, l, o) + scores_s[h, d, l]$
    $edge \leftarrow \langle score, h, d, l \rangle$
    APPEND($E, edge$)
  return the top k edges in E

function CREATE-SEC-OPEN(L, i_L, E, I)
  $score \leftarrow \text{FST}(L[i_L]) + \sum_{h,d} \text{FST}(E[h, d, I_E[h, d]])$
  return $\langle score, L, i_L, E, I_E \rangle$

function ADVANCE-SEC-OPEN(o, k)
where $o = \langle score, L, i_L, E, I_E \rangle$
  $buf \leftarrow \emptyset$
  if $i_L <$ LENGTH(L) and $I_E = [1, \ldots, 1]$
    APPEND(buf, FIRST-SEC-OPEN($L, i_L + 1, k$))
  for h, d
    if $I_E[h, d] <$ LENGTH($E[h, d]$)
      $I'_E \leftarrow \text{COPY}(I_E)$
      $I'_E[h, d] \leftarrow I'_E[h, d] + 1$
      APPEND(buf, CREATE-SEC-OPEN(L, i_L, E, I'_E))
  return buf

To apply the bistratal search algorithm to the problem of syntactic–semantic parsing, a problem-specific implementation of the SCOPE function is needed. In this case, we made two assumptions. First, we assumed that the identities of the predicate words are known a priori[2]. Secondly, we assumed that every argument of a given predicate word is either a direct dependent of the predicate, one of its ancestors, or a direct dependent of one of its ancestors. This assumption is a simple adaptation of the pruning algorithm by Xue and Palmer (2004), and it holds for the vast majority of arguments in the CoNLL-2008 data; in the training set, we measured that this covers 99.04% of the arguments of verbs and 97.55% of the arguments of nouns.

Figure 4 shows an example of how the SCOPE function works in our case. If a predicate is contained in the right substructure, we find two potential arguments: one at the start of the left substructure, and one more by recursively searching the left structure.

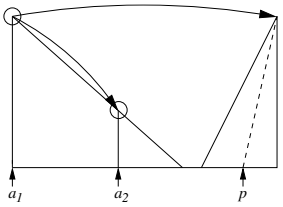

Figure 4: Illustration of the SCOPE function for predicate–argument links. If the right substructure contains a predicate, we can find potential arguments in the left substructure.

While the primary layer is assumed to be projective in Algorithm 3, the syntactic trees in the CoNLL-2008 data have a small number of nonprojective links. We used a pseudo-projective edge label encoding to handle nonprojectivity (Nivre and Nilsson, 2005).

To implement the model, we constructed feature representations $\boldsymbol{\Phi}_p$, $\boldsymbol{\Phi}_s$, and $\boldsymbol{\Phi}_i$. The surface-syntactic representation $\boldsymbol{\Phi}_p$ was a standard first-order edge factorization using the same features as McDonald et al. (2005). The features in $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_i$ are shown in Table 1 and are standard features in statistical semantic role classification.

$\boldsymbol{\Phi}_s$	$\boldsymbol{\Phi}_i$
Predicate word	Path
Predicate POS	Path + arg. POS
Argument word	Path + pred. POS
Argument POS	Path + arg. word
Pred. + arg. words	Path + pred. word
Predicate word + label	Path + label
Predicate POS + label	Path + arg. POS + label
Argument word + label	Path + pred. POS + label
Argument POS + label	Path + arg. word + label
Pred. + arg. words + label	Path + pred. word + label

Table 1: Feature representation for secondary links.

We trained the discriminative model using the Online Passive–aggressive algorithm (Crammer et al., 2006), which is an efficient online learning method that can be used to train models for learning problems with structured output spaces. A cost function ρ is needed in the learning algorithm; we decomposed it into a pri-

[2]Since our algorithm needs to know the positions of the predicates, we trained a separate classifier using the LIBLINEAR toolkit (Fan et al., 2008) to identify the predicate words. As features for the classifier, we used the words and part-of-speech tags in a ± 3 window around the word under consideration.

mary part ρ_p and a secondary part ρ_s. We computed the primary part as the sum of link errors: $\rho_p(y_p, \hat{y}_p) = \sum_{l \in \hat{y}_p} c_p(l, y_p)$, where

$$c_p(l, y_p) = \begin{cases} 0 & \text{if } l \in y_p \text{ and its label is correct} \\ 0.5 & \text{if } l \in y_p \text{ but its label is incorrect} \\ 1 & \text{if } l \notin y_p \end{cases}$$

In a similar vein, we computed the secondary part ρ_s of the cost function as $\#fp + \#fn + 0.5 \cdot \#fl$, where $\#fp$ is the number of false positive secondary links, $\#fn$ the number of false negative links, and $\#fl$ the number of links with correct endpoints but incorrect label.

The training procedure took roughly 24 hours on an 2.3 GHz AMD Athlon processor. The memory consumption was about 1 GB during training.

4 Experiments

We evaluated the performance of our system on the test set from the CoNLL-2008 shared task, which consists of section 23 of the WSJ part of the Penn Treebank, as well as a small part of the Brown corpus. A beam width k of 4 was used in this experiment. Table 2 shows the results of the evaluation. The table shows the three most important scores computed by the official evaluation script: labeled syntactic dependency accuracy (LAS), labeled semantic dependency F_1-measure (Sem. F1), and the macro-averaged F_1-measure, a weighted combination of the syntactic and semantic scores (M. F1). Our result is competitive; we obtain the same macro F1 as the newly published result by Titov et al. (2009), which is the highest published figure for a joint syntactic–semantic parser so far. Importantly, our system clearly outperforms the system by Lluís and Màrquez (2008), which is the most similar system in problem modeling, but which uses a different search strategy.

System	LAS	Sem. F1	M. F1
This paper	86.6	77.1	81.8
Titov et al. (2009)	87.5	76.1	81.8
H. et al (2008)	87.6	73.1	80.5
L. & M. (2008)	85.8	70.3	78.1

Table 2: Results of published joint syntactic–semantic parsers on the CoNLL-2008 test set.

Since the search procedure is inexact, it is important to quantify roughly how much of a detrimental impact the approximation has on the parsing quality. We studied the influence of the beam width parameter k on the performance of the parser. The results on the development set can be seen in Table 3. As can be seen, a modest increase in performance can be obtained by increasing the beam width, at the cost of increased parsing time.

k	LAS	Sem. F1	M. F1	Time
1	85.14	77.05	81.10	242
2	85.43	77.17	81.30	369
4	85.49	77.20	81.35	625
8	85.58	77.20	81.40	1178

Table 3: Influence of beam width on parsing accuracy.

In addition, to have a rough indication of the impact of search errors on the quality of the parses, we computed the fraction of sentences where the gold-standard parse had a higher score according to the model than the parse returned by the search[3]. Table 4 shows the results of this experiment. This suggests that the search errors, although they clearly have an impact, are not the major source of errors, even with small beam widths.

k	Fraction
1	0.121
2	0.104
4	0.096
8	0.090

Table 4: Fraction of sentences in the development set where the gold-standard parse has a higher score than the parse returned by the search procedure.

To investigate where future optimization efforts should be spent, we used the built-in `hprof` profiling tool of Java to locate the bottlenecks. Once again, we ran the program on the development set with a beam width of 4, and Table 5 shows the three types of operations where the algorithm spent most of its time. It turns out that 74% of the time was spent on the computation and scoring of interdependency features. To make our algorithm truly useful in practice, we thus need to devise a way to speed up or cache these computations.

[3]To be able to compare the scores of the gold-standard and predicted parses, we disabled the automatic classifier for predicate identification and provided the parser with gold-standard predicates in this experiment.

Operation	Fraction
$\boldsymbol{w} \cdot \Phi_i$	0.64
Queue operations	0.15
Computation of Φ_i	0.10

Table 5: The three most significant bottlenecks and their fraction of the total runtime.

5 Discussion

In this paper, we have presented a new approximate search method to solve the problem of jointly predicting the two layers in a bistratal dependency graph. The algorithm shows competitive performance on the treebank used in the CoNLL-2008 Shared Task, a bistratal treebank consisting of a surface-syntactic and a shallow semantic layer. In addition to the syntactic–semantic task that we have described in this paper, we believe that our method can be used in other types of multistratal syntactic frameworks, such as a representation of surface and deep syntax as in Meaning–Text Theory (Mel'čuk, 1988).

The optimization problem that we set out to solve is intractable, but we have shown that reasonable performance can be achieved with an inexact, beam search-based search method. This is not obvious: it has previously been shown that using an inexact search procedure when the learning algorithm assumes that the search is exact may lead to slow convergence or even divergence (Kulesza and Pereira, 2008), but this does not seem to be a problem in our case.

While we used a beam search method as the method of approximation, other methods are certainly possible. An interesting example is the recent system by Smith and Eisner (2008), which used loopy belief propagation in a dependency parser using highly complex features, while still maintaining cubic-time search complexity.

An obvious drawback of our approach compared to traditional pipeline-based semantic role labeling methods is that the speed of the algorithm is highly dependent on the size of the interdependency feature representation $\boldsymbol{\Phi}_i$. Also, extracting these features is fairly complex, and it is of critical importance to implement the feature extraction procedure efficiently since it is one of the bottlenecks of the algorithm. It is plausible that our performance suffers from the absence of other frequently used syntax-based features such as dependent-of-dependent and voice.

It is thus highly dubious that a joint modeling of syntactic and semantic structure is worth the additional implementational effort. So far, no system using tightly integrated syntactic and semantic processing has been competitive with the best systems, which have been either completely pipeline-based (Che et al., 2008; Ciaramita et al., 2008) or employed only a loose syntactic–semantic coupling (Johansson and Nugues, 2008). It has been conjectured that modeling the semantics of the sentence would also help in syntactic disambiguation; however, it is likely that this is already implicitly taken into account by the lexical features present in virtually all modern parsers.

In addition, a problem that our beam search method has in common with the constituent parsing method by Huang (2008) is that highly non-local features must be computed late. In our case, this means that if there is a long distance between a predicate and an argument, the secondary link between them will be unlikely to influence the final search result.

Acknowledgements

The author is grateful for the helpful comments by the reviewers. This work has been funded by the LivingKnowledge project under the seventh EU framework program.

References

Matthias Buch-Kromann. 2006. *Discontinuous Grammar. A dependency-based model of human parsing and language learning*. Ph.D. thesis, Copenhagen Business School.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. DeSRL: A linear-time semantic role labeling system. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater.

2004. A relational syntax-semantics interface based on dependency grammar. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Kevin Gimpel and Noah A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proceedings of the Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Alex Kulesza and Fernando Pereira. 2008. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems 20*.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York.

Paola Merlo and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL–2008)*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kenji Sagae and Jun'ichi Tsuji. 2008. Shift–reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht:Reidel Publishing Company and Prague:Academia.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, United States.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL–2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL–2008*.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 589–596.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.