Structural Correspondence Learning for Parse Disambiguation

Barbara Plank
Alfa-informatica
University of Groningen, The Netherlands
b.plank@rug.nl

Abstract

The paper presents an application of Structural Correspondence Learning (SCL) (Blitzer et al., 2006) for domain adaptation of a stochastic attribute-value grammar (SAVG). So far, SCL has been applied successfully in NLP for Part-of-Speech tagging and Sentiment Analysis (Blitzer et al., 2006; Blitzer et al., 2007). An attempt was made in the CoNLL 2007 shared task to apply SCL to non-projective dependency parsing (Shimizu and Nakagawa, 2007), however, without any clear conclusions. We report on our exploration of applying SCL to adapt a syntactic disambiguation model and show promising initial results on Wikipedia domains.

1 Introduction

Many current, effective natural language processing systems are based on supervised Machine Learning techniques. The parameters of such systems are estimated to best reflect the characteristics of the training data, at the cost of portability: a system will be successful only as long as the training material resembles the input that the model gets. Therefore, whenever we have access to a large amount of labeled data from some "source" (out-of-domain), but we would like a model that performs well on some new "target" domain (Gildea, 2001; Daumé III, 2007), we face the problem of *domain adaptation*.

The need for domain adaptation arises in many NLP tasks: Part-of-Speech tagging, Sentiment Analysis, Semantic Role Labeling or Statistical Parsing, to name but a few. For example, the performance of a statistical parsing system drops in an appalling way when a model trained on the Wall Street Journal is applied to the more varied Brown corpus (Gildea, 2001).

The problem itself has started to get attention only recently (Roark and Bacchiani, 2003; Hara et al., 2005; Daumé III and Marcu, 2006; Daumé III, 2007; Blitzer et al., 2006; McClosky et al., 2006; Dredze et al., 2007). We distinguish two main approaches to domain adaptation that have been addressed in the literature (Daumé III, 2007): *supervised* and *semi-supervised*.

In *supervised domain adaptation* (Gildea, 2001; Roark and Bacchiani, 2003; Hara et al., 2005; Daumé III, 2007), besides the labeled source data, we have access to a comparably small, but labeled amount of target data. In contrast, *semi-supervised domain adaptation* (Blitzer et al., 2006; McClosky et al., 2006; Dredze et al., 2007) is the scenario in which, in addition to the labeled source data, we only have *unlabeled* and no labeled target domain data. Semi-supervised adaptation is a much more realistic situation, while at the same time also considerably more difficult.

Studies on the supervised task have shown that straightforward baselines (e.g. models based on source only, target only, or the union of the data) achieve a relatively high performance level and are "surprisingly difficult to beat" (Daumé III, 2007). Thus, one conclusion from that line of work is that as soon as there is a reasonable (often even small) amount of labeled target data, it is often more fruitful to either just use that, or to apply simple adaptation techniques (Daumé III, 2007; Plank and van Noord, 2008).

2 Motivation and Prior Work

While several authors have looked at the supervised adaptation case, there are less (and especially less successful) studies on semi-supervised domain adaptation (McClosky et al., 2006; Blitzer et al., 2006; Dredze et al., 2007). Of these, McClosky et al. (2006) deal specifically with self-training for data-driven statistical parsing. They show that together with a re-ranker, improvements

Proceedings of the EACL 2009 Student Research Workshop, pages 37–45,
Athens, Greece, 2 April 2009. ©2009 Association for Computational Linguistics

are obtained. Similarly, Structural Correspondence Learning (Blitzer et al., 2006; Blitzer et al., 2007; Blitzer, 2008) has proven to be successful for the two tasks examined, PoS tagging and Sentiment Classification. In contrast, Dredze et al. (2007) report on “frustrating” results on the CoNLL 2007 semi-supervised adaptation task for dependency parsing, i.e. ”no team was able to improve target domain performance substantially over a state of the art baseline”. In the same shared task, an attempt was made to apply SCL to domain adaptation for data-driven dependency parsing (Shimizu and Nakagawa, 2007). The system just ended up at rank 7 out of 8 teams. However, based on annotation differences in the datasets (Dredze et al., 2007) and a bug in their system (Shimizu and Nakagawa, 2007), their results are inconclusive.[1] Thus, the effectiveness of SCL is rather unexplored for parsing.

So far, most previous work on domain adaptation for parsing has focused on *data-driven* systems (Gildea, 2001; Roark and Bacchiani, 2003; McClosky et al., 2006; Shimizu and Nakagawa, 2007), i.e. systems employing (constituent or dependency based) *treebank grammars* (Charniak, 1996). Parse selection constitutes an important part of many parsing systems (Johnson et al., 1999; Hara et al., 2005; van Noord and Malouf, 2005; McClosky et al., 2006). Yet, the adaptation of parse selection models to novel domains is a far less studied area. This may be motivated by the fact that potential gains for this task are inherently bounded by the underlying grammar. The few studies on adapting disambiguation models (Hara et al., 2005; Plank and van Noord, 2008) have focused exclusively on the supervised scenario.

Therefore, the direction we explore in this study is semi-supervised domain adaptation for parse disambiguation. We examine the effectiveness of *Structural Correspondence Learning* (SCL) (Blitzer et al., 2006) for this task, a recently proposed adaptation technique shown to be effective for PoS tagging and Sentiment Analysis. The system used in this study is Alpino, a wide-coverage Stochastic Attribute Value Grammar (SAVG) for Dutch (van Noord and Malouf, 2005; van Noord, 2006). For our empirical evaluation we explore Wikipedia as primary test and training collection.

In the sequel, we first introduce the parsing system. Section 4 reviews Structural Correspondence Learning and shows our application of SCL to parse selection, including all our design choices. In Section 5 we present the datasets, introduce the process of constructing target domain data from Wikipedia, and discuss interesting initial empirical results of this ongoing study.

3 Background: Alpino parser

Alpino (van Noord and Malouf, 2005; van Noord, 2006) is a robust computational analyzer for Dutch that implements the conceptual two-stage parsing approach. The system consists of approximately 800 grammar rules in the tradition of HPSG, and a large hand-crafted lexicon, that together with a left-corner parser constitutes the generation component. For parse selection, Alpino employs a discriminative approach based on Maximum Entropy (MaxEnt). The output of the parser is dependency structure based on the guidelines of CGN (Oostdijk, 2000).

The Maximum Entropy model (Berger et al., 1996; Ratnaparkhi, 1997; Abney, 1997) is a conditional model that assigns a probability to every possible parse ω for a given sentence s. The model consists of a set of m feature functions $f_j(\omega)$ that describe properties of parses, together with their associated weights θ_j. The denominator is a normalization term where $Y(s)$ is the set of parses with yield s:

$$p_\theta(\omega|s;\theta) = \frac{\exp(\sum_{j=1}^{m} \theta_j f_j(\omega))}{\sum_{y \in Y(s)} \exp(\sum_{j=1}^{m} \theta_j f_j(y)))} \quad (1)$$

The parameters (weights) θ_j can be estimated efficiently by maximizing the regularized conditional likelihood of a training corpus (Johnson et al., 1999; van Noord and Malouf, 2005):

$$\hat{\theta} = \arg\max_{\theta} \log L(\theta) - \frac{\sum_{j=1}^{m} \theta_j^2}{2\sigma^2} \quad (2)$$

where $L(\theta)$ is the likelihood of the training data. The second term is a regularization term (Gaussian prior on the feature weights with mean zero and variance σ). The estimated weights determine the contribution of each feature. Features appearing in correct parses are given increasing (positive) weight, while features in incorrect parses are

[1] As shown in Dredze et al. (2007), the biggest problem for the shared task was that the provided datasets were annotated with different annotation guidelines, thus the general conclusion was that the task was ill-defined (Nobuyuki Shimizu, personal communication).

given decreasing (negative) weight. Once a model is trained, it can be applied to choose the parse with the highest sum of feature weights.

The MaxEnt model consists of a large set of features, corresponding to instantiations of feature templates that model various properties of parses. For instance, Part-of-Speech tags, dependency relations, grammar rule applications, etc. The current standard model uses about 11,000 features. We will refer to this set of features as original features. They are used to train the baseline model on the given labeled source data.

4 Structural Correspondence Learning

SCL (Structural Correspondence Learning) (Blitzer et al., 2006; Blitzer et al., 2007; Blitzer, 2008) is a recently proposed domain adaptation technique which uses unlabeled data from *both* source and target domain to learn correspondences between features from different domains.

Before describing the algorithm in detail, let us illustrate the intuition behind SCL with an example, borrowed from Blitzer et al. (2007). Suppose we have a Sentiment Analysis system trained on book reviews (domain A), and we would like to adapt it to kitchen appliances (domain B). Features such as “boring” and “repetitive” are common ways to express negative sentiment in A, while “not working” or “defective” are specific to B. If there are features across the domains, e.g. “don’t buy”, with which the domain specific features are highly correlated with, then we might tentatively align those features.

Therefore, the key idea of SCL is to identify automatically correspondences among features from different domains by modeling their correlations with *pivot features*. Pivots are features occurring frequently and behaving similarly in both domains (Blitzer et al., 2006). They are inspired by auxiliary problems from Ando and Zhang (2005). Non-pivot features that correspond with many of the same pivot-features are assumed to correspond. Intuitively, if we are able to find good correspondences among features, then the augmented labeled source domain data should transfer better to a target domain (where no labeled data is available) (Blitzer et al., 2006).

```
Input:  - labeled source data $\{(x_s, y_s)_{s=1}^{N_s}\}$
        - unlabeled data from both source and
        target domain $x_{ul} = x_s, x_t$

1. Select $m$ pivot features
2. Train $m$ binary classifiers (pivot predictors)
3. Create matrix $W_{n \times m}$ of binary predictor
   weight vectors $W = [w_1, .., w_m]$, where $n$
   is the number of nonpivot features in $x_{ul}$
4. Apply SVD to $W$: $W_{n \times m} = U_{n \times n} D_{n \times m} V^T_{m \times m}$ where $\theta = U^T_{[1:h,:]}$
   are the $h$ top left singular vectors of $W$.
5. Apply projection $x_s \theta$ and train a predictor
   on the original and new features obtained
   through the projection.
```

Figure 1: SCL algorithm (Blitzer et al., 2006).

The outline of the algorithm is given in Figure 1. The first step is to identify m pivot features occurring frequently in the unlabeled data of both domains. Then, a binary classifier is trained for each pivot feature (pivot predictor) of the form: “Does pivot feature l occur in this instance?”. The pivots are masked in the unlabeled data and the aim is to predict them using non-pivot features. In this way, we obtain a weight vector w for each pivot predictor. Positive entries in the weight vector indicate that a non-pivot is highly correlated with the respective pivot feature. Step 3 is to arrange the m weight vectors in a matrix W, where a column corresponds to a pivot predictor weight vector. Applying the projection $W^T x$ (where x is a training instance) would give us m new features, however, for “both computational and statistical reasons” (Blitzer et al., 2006; Ando and Zhang, 2005) a low-dimensional approximation of the original feature space is computed by applying Singular Value Decomposition (SVD) on W (step 4). Let $\theta = U^T_{h \times n}$ be the top h left singular vectors of W (with h a dimension parameter and n the number of non-pivot features). The resulting θ is a projection onto a lower dimensional space $\mathbb{R}^h$, parameterized by h.

The final step of SCL is to train a linear predictor on the augmented labeled source data $\langle x, \theta x \rangle$. In more detail, the original feature space x is augmented with h new features obtained by applying the projection θx. In this way, we can learn weights for domain-specific features, which otherwise would not have been observed. If θ contains meaningful correspondences, then the predictor trained on the augmented data should transfer well to the new domain.

4.1 SCL for Parse Disambiguation

A property of the pivot predictors is that they can be trained from unlabeled data, as they represent properties of the input. So far, pivot features on the *word level* were used (Blitzer et al., 2006; Blitzer et al., 2007; Blitzer, 2008), e.g. “Does the bigram *not buy* occur in this document?” (Blitzer, 2008).

Pivot features are the key ingredient for SCL, and they should align well with the NLP task. For PoS tagging and Sentiment Analysis, features on the word level are intuitively well-related to the problem at hand. For the task of parse disambiguation based on a conditional model this is not the case.

Hence, we actually introduce an additional and new layer of abstraction, which, we hypothesize, aligns well with the task of parse disambiguation: we first *parse* the unlabeled data. In this way we obtain full parses for given sentences as produced by the grammar, allowing access to more abstract representations of the underlying pivot predictor training data (for reasons of efficiency, we here use only the first generated parse as training data for the pivot predictors, rather than n-best).

Thus, instead of using word-level features, our features correspond to properties of the generated parses: application of grammar rules (*r1,r2* features), dependency relations (*dep*), PoS tags (*f1,f2*), syntactic features (*s1*), precedence (*mf*), bilexical preferences (*z*), apposition (*appos*) and further features for unknown words, temporal phrases, coordination (*h,in_year* and *p1*, respectively). This allows us to get a possibly noisy, but more abstract representation of the underlying data. The set of features used in Alpino is further described in van Noord and Malouf (2005).

Selection of pivot features As pivot features should be common across domains, here we restrict our pivots to be of the type *r1,p1,s1* (the most frequently occurring feature types). In more detail, *r1* indicates which grammar rule applied, *p1* whether coordination conjuncts are parallel, and *s1* whether topicalization or long-distance dependencies occurred. We count how often each feature appears in the parsed source and target domain data, and select those *r1,p1,s1* features as *pivot features*, whose count is $> t$, where t is a specified threshold. In all our experiments, we set $t = 5000$. In this way we obtained on average 360 pivot features, on the datasets described in Section 5.

Predictive features As pointed out by Blitzer et al. (2006), each instance will actually contain features which are totally predictive of the pivot features (i.e. the pivot itself). In our case, we additionally have to pay attention to ’more specific’ features, e.g. *r2* is a feature that extends *r1*, in the sense that it incorporates more information than its parent (i.e. which grammar rules applied in the construction of daughter nodes). It is crucial to remove these predictive features when creating the training data for the pivot predictors.

Matrix and SVD Following Blitzer et al. (2006) (which follow Ando and Zhang (2005)), we only use positive entries in the pivot predictors weight vectors to compute the SVD. Thus, when constructing the matrix W, we disregard all negative entries in W and compute the SVD ($W = UDV^T$) on the resulting non-negative sparse matrix. This sparse representation saves both time and space.

4.2 Further practical issues of SCL

In practice, there are more free parameters and model choices (Ando and Zhang, 2005; Ando, 2006; Blitzer et al., 2006; Blitzer, 2008) besides the ones discussed above.

Feature normalization and feature scaling. Blitzer et al. (2006) found it necessary to normalize and scale the new features obtained by the projection θ, in order to “allow them to receive more weight from a regularized discriminative learner”. For each of the features, they centered them by subtracting out the mean and normalized them to unit variance (i.e. $x - mean/sd$). They then rescaled the features by a factor α found on held-out data: $\alpha\theta x$.

Restricted Regularization. When training the supervised model on the augmented feature space $\langle x, \theta x\rangle$, Blitzer et al. (2006) only regularize the weight vector of the original features, but not the one for the new low-dimensional features. This was done to encourage the model to use the new low-dimensional representation rather than the higher-dimensional original representation (Blitzer, 2008).

Dimensionality reduction by feature type. An extension suggested in Ando and Zhang (2005) is to compute separate SVDs for blocks of the matrix W corresponding to feature types (as illustrated in Figure 2), and then to apply separate projection for every type. Due to the positive results in Ando (2006), Blitzer et al. (2006) include this in their standard setting of SCL and report results using block SVDs only.

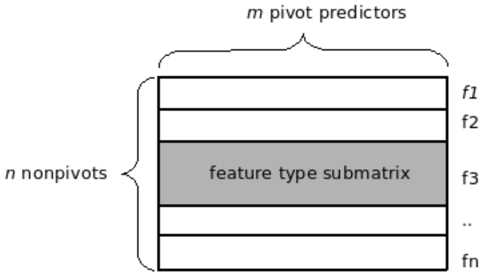

Figure 2: Illustration of dimensionality reduction by feature type (Ando and Zhang, 2005). The grey area corresponds to a feature type (submatrix of W) on which the SVD is computed (block SVD); the white area is regarded as fixed to zero matrices.

5 Experiments and Results

5.1 Experimental design

The base (source domain) disambiguation model is trained on the Alpino Treebank (van Noord, 2006) (newspaper text), which consists of approximately 7,000 sentences and 145,000 tokens. For parameter estimation of the disambiguation model, in all reported experiments we use the TADM[2] toolkit (toolkit for advanced discriminative training), with a Gaussian prior (σ^2=1000) and the (default) limited memory variable metric estimation technique (Malouf, 2002).

For training the binary pivot predictors, we use the MegaM[3] Optimization Package with the so-called "bernoulli implicit" input format. To compute the SVD, we use SVDLIBC.[4]

The output of the parser is dependency structure. A standard evaluation metric is to measure the amount of generated dependencies that are identical to the stored dependencies (correct labeled dependencies), expressed as f-score. An alternative measure is concept accuracy (CA), which is similar to f-score, but allows possible discrepancy between the number of returned dependencies (van Noord, 2006; Plank and van Noord, 2008). CA is usually slightly lower than f-score. Let D_p^i be the number of dependencies produced by the parser for sentence i. D_g^i is the number of dependencies in the treebank parse, and D_o^i is the number of correct dependencies produced by the parser. Then,

$$\text{CA} = \frac{D_o}{\sum_i \max(D_g^i, D_p^i)}$$

If we want to compare the performance of disambiguation models, we can employ the ϕ measure (van Noord and Malouf, 2005; van Noord, 2007). Intuitively, it tells us how much of the disambiguation problem has been solved.

$$\phi = \frac{CA - base}{oracle - base} \times 100$$

In more detail, the ϕ measure incorporates an upper and lower bound: *base* measures the accuracy of a model that simply selects the first parse for each sentence; *oracle* represents the accuracy achieved by a model that always selects the best parse from the set of potential parses (within the coverage of the parser). In addition, we also report *relative error reduction* (rel.er), which is the relative difference in ϕ scores for two models.

As target domain, we consider the Dutch part of Wikipedia as data collection, described in the following.

5.2 Wikipedia as resource

In our experiments, we exploit Wikipedia both as testset and as unlabeled data source. We assume that in order to parse data from a very specific domain, say about the artist Prince, then data related to that domain, like information about the New Power Generation, the Purple rain movie, or other American singers and artists, should be of help. Thus, we exploit Wikipedia and its category system to gather domain-specific target data.

Construction of target domain data In more detail, we use the Dutch part of Wikipedia provided by WikiXML,[5] a collection of Wikipedia articles converted to XML format. As the corpus is encoded in XML, we can exploit general purpose XML Query Languages, such as XQuery, Xslt and XPath, to extract relevant information from the Wikipedia corpus.

Given a wikipage p, with $c \in categories(p)$, we can identify pages related to p of various

[2] http://tadm.sourceforge.net/
[3] http://www.cs.utah.edu/~hal/megam/
[4] http://tedlab.mit.edu/~dr/svdlibc/
[5] http://ilps.science.uva.nl/WikiXML/

types of 'relatedness': directly related pages (those that share a category, i.e. all p' where $\exists c' \in categories(p')$ such that $c = c'$), or alternatively, pages that share a sub- or supercategory of p, i.e. p' where $c' \in categories(p')$ and $c' \in sub\_categories(p)$ or $c' \in super\_categories(p)$. For example, Figure 3 shows the categories extracted for the Wikipedia article about pope Johannes Paulus II.

```
<wikipage id="6677">
 <cat t="direct" n="Categorie:Paus"/>
 <cat t="direct" n="Categorie:Pools_theoloog"/>
 <cat t="super" n="Categorie:Religieus leider"/>
 <cat t="super" n="Categorie:Rooms-katholiek persoon"/>
 <cat t="super" n="Categorie:Vaticaanstad"/>
 <cat t="super" n="Categorie:Bisschop"/>
 <cat t="super" n="Categorie:Kerkgeschiedenis"/>
 <cat t="sub" n="Categorie:Tegenpaus"/>
 <cat t="super" n="Categorie:Pools persoon"/>
</wikipage>
```

Figure 3: Example of extracted Wikipedia categories for a given article (direct, sup- and subcats).

To create the set of related pages for a given article p, we proceed as follows:

1. Find sub- and supercategories of p
2. Extract all pages that are related to p (through sharing a direct, sub or super category)
3. Optionally, filter out certain pages

In our empirical setup, we followed Blitzer et al. (2006) and tried to balance the size of source and target data. Thus, depending on the size of the resulting target domain dataset, and the "broadness" of the categories involved in creating it, we might wish to filter out certain pages. We implemented a filter mechanism that excludes pages of a certain category (e.g. a supercategory that is hypothesized to be "too broad"). Alternatively, we might have used a filter mechanism that excludes certain pages directly.

In our experiments, we always included pages that are directly related to a page of interest, and those that shared a subcategory. Of course, the page itself is not included in that dataset. With regard to supercategories, we usually included all pages having a category $c \in super\_categories(p)$, unless stated otherwise.

Test collection Our testset consists of a selection of Wikipedia articles that have been manually corrected in the course of the D-Coi/LASSY project.[6] An overview of the testset including size indications is given in Table 1. Table 2 provides information on the target domain datasets constructed from Wikipedia.

Wiki/DCOI ID	Title	Sents
6677/026563	Prince (musician)	358
6729/036834	Paus Johannes Paulus II	232
182654/041235	Augustus De Morgan	259

Table 1: Size of test datasets.

Related to	Articles	Sents	Tokens	Relationship
Prince	290	9,772	145,504	filtered super
Paus	445	8,832	134,451	all
De Morgan	394	8,466	132,948	all

Table 2: Size of related unlabeled data; relationship indicates whether all related pages are used or some are filtered out (see section 5.2).

5.3 Empirical Results

For all reported results, we randomly select $n = 200$ maximum number of parses per sentence for evaluation.

Baseline accuracies Table 3 shows the baseline performance (of the standard Alpino model) on the various Wikipedia testsets (CA, f-score). The third and fourth column indicate the upper- and lower bound measures (defined in section 5.1).

Title	CA	f-score	base	oracle
Prince (musician)	85.03	85.38	71.95	88.70
Paus Johannes Paulus II	85.72	86.32	74.30	89.09
Augustus De Morgan	80.09	80.61	70.08	83.52

Table 3: Baseline results.

While the parser normally operates on an accuracy level of roughly 88-89% (van Noord, 2007) on its own domain (newspaper text), the accuracy on these subdomains drops to around 85%. The biggest performance decrease (to 80%) was on the article about the British logician and mathematician De Morgan. This confirms the intuition that this specific subdomain is the "hardest", given that mathematical expressions might emerge in the data (e.g. "Wet der distributiviteit : a(b+c) = ab+ac" - distributivity law).

SCL results Table 4 shows the results of our instantiation of SCL for parse disambiguation, with varying h parameter (dimensionality parameter;

[6] Ongoing project, see http://www.let.rug.nl/~vannoord/Lassy/

$h = 25$ means that applying the projection $x\theta$ resulted in adding 25 new features to every source domain instance).

	CA	f-score	ϕ	rel.er.
baseline Prince	85.03	85.38	78.06	0.00
SCL[+/-], $h = 25$	85.12	85.46	78.64	2.64
SCL[+/-], $h = 50$	85.29	85.63	79.66	7.29
SCL[+/-], $h = 100$	85.19	85.53	79.04	4.47
SCL[+/-], $h = 200$	85.21	85.54	79.18	5.10
baseline Paus	85.72	86.32	77.23	0.00
SCL[+/-], $h = 25$	85.87	86.48	78.26	4.52
SCL[+/-], $h = 50$	85.82	86.43	77.87	2.81
SCL[+/-], $h = 100$	85.87	86.49	78.26	4.52
SCL[+/-], $h = 200$	85.87	86.48	78.26	4.52
baseline DeMorgan	80.09	80.61	74.44	0.00
SCL[+/-], $h = 25$	80.15	80.67	74.92	1.88
SCL[+/-], $h = 50$	80.12	80.64	74.68	0.94
SCL[+/-], $h = 100$	80.12	80.64	74.68	0.94
SCL[+/-], $h = 200$	80.15	80.67	74.91	1.88

Table 4: Results of our instantiation of SCL (with varying h parameter and no feature normalization).

The results show a (sometimes) small but consistent increase in absolute performance on all testsets over the baseline system (up to $+0.26$ absolute CA score), as well as an increase in ϕ measure (absolute error reduction). This corresponds to a relative error reduction of up to 7.29%. Thus, our first instantiation of SCL for parse disambiguation indeed shows promising results.

We can confirm that changing the dimensionality parameter h has rather little effect (Table 4), which is in line with previous findings (Ando and Zhang, 2005; Blitzer et al., 2006). Thus we might fix the parameter and prefer smaller dimensionalities, which saves space and time.

Note that these results were obtained *without* any of the additional normalization, rescaling, feature-specific regularization, or block SVD issues, etc. (discussed in section 4.2). We used the same Gaussian regularization term (σ^2=1000) for all features (original and new features), and did not perform any feature normalization or rescaling. This means our current instantiation of SCL is an actually *simplified* version of the original SCL algorithm, applied to parse disambiguation. Of course, our results are preliminary and, rather than warranting many definite conclusions, encourage further exploration of SCL and related semi-supervised adaptation techniques.

5.4 Additional Empirical Results

In the following, we describe additional results obtained by extensions and/or refinements of our current SCL instantiation.

Feature normalization. We also tested feature normalization (as described in Section 4.2). While Blitzer et al. (2006) found it necessary to normalize (and scale) the projection features, we did not observe any improvement by normalizing them (actually, it slightly degraded performance in our case). Thus, we found this step unnecessary, and currently did not look at this issue any further.

A look at θ To gain some insight of which kind of correspondences SCL learned in our case, we started to examine the rows of θ. Recall that applying a row of the projection matrix θ_i to a training instance x gives us a new real-valued feature. If features from different domains have similar entries (scores) in the projection row, they are assumed to correspond (Blitzer, 2008). Figure 4 shows example of correspondences that SCL found in the Prince dataset. The first column represents the score of a feature. The labels `wiki` and `alp` indicate the domain of the features, respectively. For readability, we here grouped the features obtaining similar scores.

```
0.00010248|dep35('Chaka Khan',name('PER'),hd/su,verb,ben)|wiki
0.00010248|dep35(de,det,hd/det,adj,'Afro-Amerikaanse')|wiki
0.00010248|dep35('Yvette Marie Stevens',name('PER'),hd/app,
                noun,zangeres)|wiki
0.000102772|dep34(leraar,noun,hd/su,verb)|alp

0.000161095|dep34(commissie,noun,hd/obj1,prep)|16|alp
0.00016113|dep34('Confessions Tour',name,hd/obj1,prep)|2|wiki
0.000161241|dep34(orgel,noun,hd/obj1,prep)|1|wiki

0.000217698|dep34(tournee,noun,hd/su,verb)|1|wiki
0.000223301|dep34(regisseur,noun,hd/su,verb)|15|wiki
0.000224517|dep34(voorsprong,noun,hd/su,verb)|2|alp
0.000224684|dep34(wetenschap,noun,hd/su,verb)|2|alp
0.000226617|dep34(pop_rock,noun,hd/su,verb)|1|wiki
0.000228918|dep34(plan,noun,hd/su,verb)|9|alp
```

Figure 4: Example projection from θ (row 2).

SCL clustered information about 'Chaka Khan', an 'Afro-Amerikaanse' 'zangeres' (afro-american singer) whose real name is 'Yvette Marie Stevens'. She had close connections to Prince, who even wrote one of her singles. These features got aligned to the Alpino feature 'leraar' (teacher). Moreover, SCL finds that 'tournee', 'regisseur' and 'pop_rock' in the Prince domain behave like 'voorsprong' (advance), 'wetenschap' (research) and 'plan' as possible heads in a subject relation in the newspaper domain. Similarly, correspondences between the direct object features 'Confessions Tour' and 'orgel' (pipe organ) to 'commissie' (commission) are discovered.

More unlabeled data In the experiments so far, we balanced the amount of source and target data. We started to examine the effect of more unlabeled target domain data. For the Prince dataset, we included all supercategories in constructing the related target domain data. The so obtained dataset contains: 859 articles, 29,186 sentences and 385,289 tokens; hence, the size approximately tripled (w.r.t. Table 2). Table 5 shows the effect of using this larger dataset for SCL with $h = 25$. The accuracy increases (from 85.12 to 85.25). Thus, there seems to be a positive effect (to be investigated further).

	CA	f-score	ϕ	rel.er.
baseline Prince	85.03	85.38	78.06	0.00
SCL[+/-], $h = 25$, all	85.25	85.58	79.42	6.20

Table 5: First result on increasing unlabeled data.

Dimensionality reduction by feature type We have started to implement the extension discussed in section 4.2, i.e. perform separate dimensionality reductions based on blocks of nonpivot features. We clustered nonpivots (see section 4.1 for a description) into 9 types (ordered in terms of decreasing cluster size): *dep*, *f1/f2* (pos), *r1/r2* (rules), *appos_person*, *mf*, *z*, *h1*, *in_year*, *dist*. For each type, a separate SVD was computed on submatrix W_t (illustrated in Figure 2). Then, separate projections were applied to every training instance.

The results of these experiments on the Prince dataset are shown in Figure 5. Applying SCL with dimensionality reduction by feature type (SCL block) results in a model that performs better (CA 85.27, ϕ 79.52, rel.er. 6.65%) than the model with no feature split (no block SVDs), thus obtaining a relative error reduction of 6.65% over the baseline. The same figure also shows what happens if we remove a specific feature type at a time; the apposition features contribute the most on this Prince domain. As a fact, one third of the sentences in the Prince testset contain constructions with appositions (e.g. about film-, album- and song titles).

6 Conclusions and Future Work

The paper presents an application of Structural Correspondence Learning (SCL) to parse disambiguation. While SCL has been successfully applied to PoS tagging and Sentiment Analysis (Blitzer et al., 2006; Blitzer et al., 2007), its effectiveness for parsing was rather unexplored.

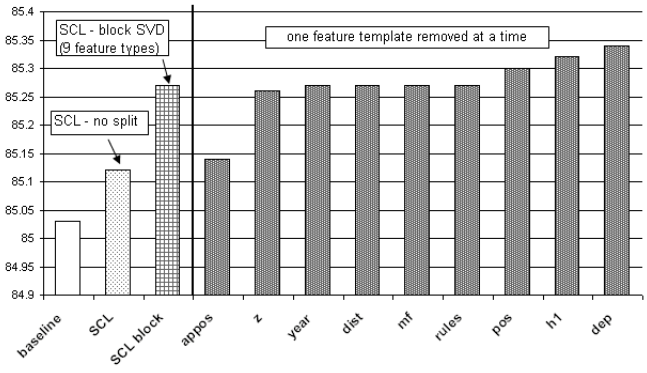

Figure 5: Results of dimensionality reduction by feature type, $h = 25$; block SVD included all 9 feature types; the right part shows the accuracy when one feature type was removed.

The empirical results show that our instantiation of SCL to parse disambiguation gives promising initial results, even without the many additional extensions on the feature level as done in Blitzer et al. (2006). We exploited Wikipedia as primary resource, both for collecting unlabeled target domain data, as well as test suite for empirical evaluation. On the three examined datasets, SCL slightly but constantly outperformed the baseline. Applying SCL involves many design choices and practical issues, which we tried to depict here in detail. A novelty in our application is that we first actually parse the unlabeled data from both domains. This allows us to get a possibly noisy, but more abstract representation of the underlying data on which the pivot predictors are trained.

In the near future, we plan to extend the work on semi-supervised domain adaptation for parse disambiguation, viz. (1) further explore/refine SCL (block SVDs, varying amount of target domain data, other testsets, etc.), and (2) examine self-training. Studies on the latter have focused mainly on generative, constituent based, i.e. data-driven parsing systems. Furthermore, from a machine learning point of view, it would be interesting to know a measure of corpus similarity to estimate the success of porting an NLP system from one domain to another. This relates to the general question of what is meant by domain.

References

Steven P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23:597–618.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*.

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, Prague, Czech Republic.

John Blitzer. 2008. *Domain Adaptation of Natural Language Processing Systems*. Ph.D. thesis, University of Pennsylvania.

Eugene Charniak. 1996. Tree-bank grammars. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Mark Dredze, John Blitzer, Pratha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for parsing. In *Proceedings of the CoNLL Shared Task Session - Conference on Natural Language Learning*, Prague, Czech Republic.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tadayoshi Hara, Miyao Yusuke, and Jun'ichi Tsujii. 2005. Adapting a probabilistic disambiguation model of an hpsg parser to a new domain. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic “unification-based” grammars. In *Proceedings of the 37th Annual Meeting of the ACL*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, Taipei.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

Nelleke Oostdijk. 2000. The Spoken Dutch Corpus: Overview and first evaluation. In *Proceedings of Second International Conference on Language Resources and Evaluation (LREC)*, pages 887–894.

Barbara Plank and Gertjan van Noord. 2008. Exploring an auxiliary distribution based approach to domain adaptation of a syntactic disambiguation model. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation (PE)*, Manchester, August.

A. Ratnaparkhi. 1997. A simple introduction to maximum entropy models for natural language processing. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised pcfg adaptation to novel domains. In *In Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Nobuyuki Shimizu and Hiroshi Nakagawa. 2007. Structural correspondence learning for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Gertjan van Noord and Robert Malouf. 2005. Wide coverage parsing with stochastic attribute value grammars. Draft available from http://www.let.rug.nl/˜vannoord. A preliminary version of this paper was published in the Proceedings of the IJCNLP workshop Beyond Shallow Analyses, Hainan China, 2004.

Gertjan van Noord. 2006. **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, pages 20–42, Leuven.

Gertjan van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the Tenth International Conference on Parsing Technologies. IWPT 2007, Prague.*, pages 1–10, Prague.