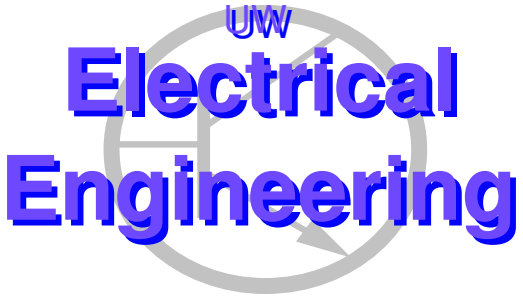
Lexicon Acquisition for Resource-Poor Languages Using Transductive Learning

Kevin Duh and Katrin Kirchhoff
{duh,katrin}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500



UWEE Technical Report
Number UWEETR-2006-0012
April 2006

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Lexicon Acquisition for Resource-Poor Languages Using Transductive Learning

Kevin Duh and Katrin Kirchhoff
{duh,katrin}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2006-0012
April 2006

Abstract

We investigate the problem of learning a part-of-speech (POS) lexicon for resource-poor languages. Developing a high-quality lexicon is often the first step towards building a POS tagger, which is in turn the front-end to many NLP systems. We frame the lexicon acquisition problem as a transductive learning problem, and perform comparisons on three transductive algorithms: Transductive SVMs, Spectral Graph Transducers, and a novel Transductive Clustering method. We test on two datasets: dialectal Arabic (a resource-poor language) and Wall Street Journal with artificially limited training data. For dialectal Arabic, we demonstrate that lexicon learning is an important task and leads to significant improvements in tagging accuracy. For Wall Street Journal, we observe that transductive learning does not necessary lead to improvements in lexicon accuracy and present some preliminary analyses of results.

1 Introduction

Due to the rising importance of globalization and multilingualism, there is a need to build natural language processing (NLP) systems for an increasingly wider range of languages, including those languages that have traditionally not been the focus of NLP research. The development of NLP technologies for a new language is a challenging task since one needs to deal not only with language-specific phenomena but also with a potential lack of available resources (e.g. lexicons, text, annotations). In this study we investigate the problem of learning a part-of-speech (POS) lexicon for a resource-poor language, dialectal Arabic.

Developing a high-quality POS lexicon is the first step towards training a POS tagger, which in turn is typically the front end for other NLP applications such as parsing and language modeling. In the case of resource-poor languages (and dialectal Arabic in particular), this step is much more critical than is typically assumed: a lexicon with too few constraints on the possible POS tags for a given word can have disastrous effects on tagging accuracy. Whereas such constraints can be obtained from large hand-labeled corpora or high-quality annotation tools in the case of resource-rich languages, no such resources are available for dialectal Arabic. Instead, constraints on possible POS tags must be inferred from a small amount of tagged words, or imperfect analysis tools. This can be seen as the problem of learning complex, structured outputs (multi-class labels, with a different number of classes for different words and dependencies among the individual labels) from partially labeled data.

Our focus is on investigating several machine learning techniques for this problem. In particular, we argue that lexicon learning in resource-poor languages can be best viewed as transductive learning. The main contribution of this work are: (1) a comprehensive evaluation of three transductive algorithms (Transductive SVM, Spectral Graph Transducer, and a new technique called Transductive Clustering) as well as an inductive SVM on this task; and (2) a demonstration that lexicon learning is a worthwhile investment and leads to significant improvements in the tagging accuracy for dialectal Arabic.

In addition, we present results on lexicon learning on the Wall Street Journal (WSJ) corpus (which is not a resource-poor domain). With this dataset, we are able to vary the amount of training data and directly measure the lexicon

accuracy. Contrary to the results on dialectal Arabic, here we observe that transductive learning does not necessarily lead to improved lexicon accuracies. We present some preliminary analyses and speculations on this result.

The outline of the paper is as follows: Section 2 describes the problem in more detail and discusses the situation in dialectal Arabic. The transductive framework and algorithms for lexicon learning are elaborated in Section 3. Sections 4 and 5 describe the data and system. Experimental results on dialectal Arabic and WSJ are presented in Sections 6 and 7, respectively. We discuss some related work in Section 8 before concluding in Section 9.

2 The Importance of Lexicons in Resource-poor POS Tagging

2.1 Unsupervised Tagging

The lack of annotated training data in resource-poor languages necessitates the use of unsupervised or semi-supervised taggers. One commonly-used unsupervised tagger is the Hidden Markov model (HMM), which models the joint distribution of a word sequence $w_{0:M}$ and tag sequence $t_{0:M}$ as:

$$P(t_{0:M}, w_{0:M}) = \prod_{i=0}^{M} p(w_i|t_i)p(t_i|t_{i-1}, t_{i-2}) \tag{1}$$

The above equation is a trigram HMM. Unsupervised learning is performed by running the Expectation-Maximization (EM) algorithm on raw text. In this procedure, the tag sequences are unknown, and the probability tables $p(w_i|t_i)$ and $p(t_i|t_{i-1}, t_{i-2})$ are iteratively updated to maximize the likelihood of the observed word sequences.

Although previous research in unsupervised tagging have achieved high accuracies rivaling supervised methods [15, 2], much of the success is due to the use of artificially *constrained* lexicons. Specifically, the lexicon is a wordlist where each word is annotated with the set of all its possible tags. (We will call the set of possible tags of a given word the *POS-set* of that word; an example: POS-set of the English word `bank` may be {`NN`,`VB`}.) Banko and Moore [1] showed that unsupervised tagger accuracies on English degrade from 96% to 77% if the lexicon is not constrained such that only high frequency tags exist in the POS-sets for each word.

Why is the lexicon so critical in unsupervised tagging? The answer is that it provides additional knowledge about word-tag distributions that may otherwise be difficult to glean from raw text alone. In the case of unsupervised HMM taggers, the lexicon provides constraints on the probability tables $p(w_i|t_i)$ and $p(t_i|t_{i-1}, t_{i-2})$. Specifically, the lexical probability table is initialized such that $p(w_i|t_i) = 0$ if and only if tag t_i is not included in the POS-set of word w_i. The transition probability table is initialized such that $p(t_i|t_{i-1}, t_{i-2}) = 0$ if and only if the tag sequence (t_i, t_{i-1}, t_{i-2}) never occurs in the tag lattice induced by the lexicon on the raw text. The effect of these zero-probability initialization is that they will always stay zero throughout the EM procedure (modulo the effects of smoothing). This therefore acts as hard constraints and biases the EM algorithm to avoid certain solutions when maximizing likelihood. If the lexicon is accurate, then the EM algorithm can learn very good predictive distributions from raw text only; conversely, if the lexicon is poor, EM will be faced with more confusability during training and may not produce a good tagger.

Thus, a critical aspect of resource-poor POS tagging is the acquisition of a high-quality lexicon. This task is challenging because the lexicon learning algorithm must not be resource-intensive. In practice, one may be able to find analysis tools or partial annotations such that only a partial lexicon is available. The focus is therefore on effective machine learning algorithms for inferring a full high-quality lexicon from a partial, possibly noisy initial lexicon. We shall now discuss this situation in the context of dialectal Arabic.

2.2 Dialectal Arabic

The Arabic language consist of a collection of spoken dialects and a standard written language (Modern Standard Arabic, or MSA). Spoken dialectal Arabic is an important set of languages since they are the native languages of Arabic speakers, and are used extensively in everyday situations. NLP technology for dialectal Arabic is still in its infancy, however, due to the lack of data and resources. Apart from small amounts of written dialectal material in e.g. plays, novels, chat rooms, etc., data can only be obtained by recording and manually transcribing actual conversations. Annotated corpora are scarce because it requires another stage of manual effort beyond transcription work. In addition, basic resources such as lexicons, morphological analyzers, tokenizers, etc. have been developed for MSA, but are scarce or non-existent for dialectal Arabic.

In this study, we address lexicon learning for Levantine Colloquial Arabic. We assume that only two resources are available during training: (1) raw text transcriptions of Levantine speech and (2) a morphological analyzer developed for MSA.

The lexicon learning task begins with a partial lexicon generated by applying the MSA analyzer to the Levantine wordlist. Since MSA differs from Levantine considerably in terms of syntax, morphology, and lexical choice, not all Levantine words receive an analysis. In our data, 23% of the words are un-analyzable. Thus, the goal of lexicon learning is to infer the POS-sets of the un-analyzable words, given the partially-annotated lexicon and raw text.

Details on the Levantine data and overall system are provided in Sections 4 and 5. We discuss the learning algorithms in the next section.

3 Learning Frameworks and Algorithms

Let us formally define the lexicon learning problem. We have a wordlist of size $m + u$. A portion of these words (m) are annotated with POS-set labels, which may be acquired by manual effort or an analysis tool. The set of labeled words $\{X_m\}$ is the training set, also referred to as the partial lexicon. The task is to predict the POS-sets of the remaining u unlabeled words $\{X_u\}$, the test set. The goal of lexicon learning is to label $\{X_u\}$ with low error. The final result is a full lexicon that contains POS-sets for all $m + u$ words.

3.1 Framework: Transductive Learning with Structured Outputs

We argue that the above problem formulation lends itself to a transductive learning framework. First, we define what we mean by transductive learning and constrast it with other popular learning frameworks (inductive learning and inductive semi-supervised learning).

Standard *inductive learning* uses a training set of fully labeled samples in order to learn a classification function. After completion of the training phase, the learned model is then used to classify samples from a new, previously unseen test set. *Semi-supervised* inductive learning exploits unlabeled data in addition to labeled data to better learn a classification function. *Transductive learning*, first described by Vapnik [19] also describes a setting where both labeled and unlabeled data are used *jointly* to decide on a label assignment to the unlabeled data points. However, the goal here is not to learn a general classification function that can be applied to new test sets multiple times but to achieve a high-quality one-time labeling of a particular data set. Transductive learning and inductive semi-supervised learning are sometimes confused in the literature. Both approaches use unlabeled data in learning – the key difference is that a transductive classifier only optimizes the performance on the given unlabeled data while inductive semi-supervised classifier is trained to perform well on any new unlabeled data.

Lexicon learning fits in the transductive learning framework as follows: The test set $\{X_u\}$, which are the unlabeled words, is static and known during learning time; we are not interested in inferring POS-sets for any words outside the word list.

An additional characterization of the lexicon learning problem is that it is a problem of learning with complex, structured outputs. The label for each word is its POS-set, which may contain a group of one to K POS tags (where K is the size of the tagset). This differs from traditional classification tasks where the ouput is a single scalar variable. Structured output problems like lexicon learning can be characterized by the granularity of the basic unit of labels. We define two cases: single-label and compound-label. In the single-label framework, each individual POS tag is the the target of classification; in the compound-label framework, each POS-set is "compounded" together to form one atomic unit, and these units are the targets of classification.

Different frameworks require different methods for setting up the classifiers (See Figure 1). In the single-label framework, we have K binary classifiers each hypothesizing whether a word has a POS tag k (where k indexes some POS tag in the tagset). Another overall classifier takes the results of the K individual classifiers and outputs a POS-set. This overall classifier can simply take all POS tags hypothesized positive by the individual binary classifiers to form the POS-set, or use a more sophisticated scheme for determining the number of POS tags [11].

The alternative compound-label framework treats each POS-set as an atomic label for classification. A POS-set such as {"NN", "VB"} is concatenated into one label "NN-VB", which results in a different label than, say, "NN" or "NN-JJ". Suppose there exist N distinct POS-sets in the training data; then we have N atomic units for labeling. Thus a (N-ary) multi-class classifier is employed to directly predict the POS-set of a word. If only binary classifiers are available (i.e. in the case of Support Vector Machines), one can use one-vs-rest, pairwise, or error correcting code

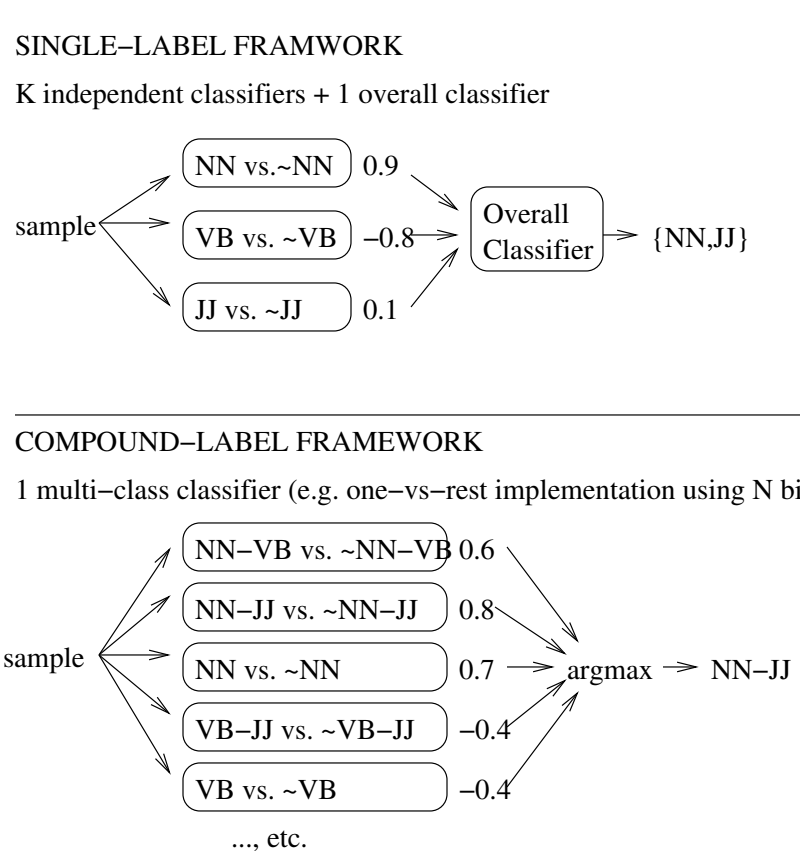


Figure 1: Learning with Structured Outputs using Multi-label or Multi-class classification

schemes to implement the multi-class classification. In our implementation, we use the one-vs-rest scheme due to computational reasons[1].

The single-label framework is potentially ill-suited for capturing the dependencies between POS tags. Dependencies between POS tags arise since some tags, such as "NN" and "NNP" can often be tagged to the same word and therefore co-occur in the POS-set label. The compound-label framework implicitly captures tag co-occurrence, but potentially suffers from over-fragmentation of training data. In our initial experiments, the compound-label framework gave better classification results; therefore we implemented all of our algorithms in the multi-class framework (using the one-vs-rest scheme and choosing the argmax as the final decision)[2].

3.2 Algorithms for Transductive Learning

3.2.1 Transductive Clustering

How does a transductive algorithm effectively utilize unlabeled samples in the learning process? One popular approach is the application of the so-called *cluster assumption*, which intuitively states that samples close to each other (i.e. samples that form a cluster) should have similar labels.

Transductive clustering (TC) is a simple algorithm that directly implements the cluster assumption. The algorithm

[1] Suppose N=300, then a one-vs-rest scheme implements 300 binary classifiers. A pairwise scheme will implement 300*299/2 binary classifiers. In general, it is difficult to say whether one-vs-rest is computationally more efficient than pairwise scheme because although pairwise uses more classifiers, each classifier is trained on a small subset of the training data. In our case, however, one-vs-rest is clearer computationally better because we are including unlabeled samples for each classifiers. Since the unlabeled dataset size may be large, this dwarves any computational advantage the small pairwise classifiers may have.

[2] The Transductive Clustering algorithm as we will describe allows one additional framework, which we call "extended-compound-label." It is similar to the compound-label framework in that the atomic unit may consist of multiple POS tags. It differs in that compound-labels other than ones found in training data may be hypothesized.

clusters labeled and unlabeled samples jointly, then use the labels of labeled samples to infer the labels of unlabeled words in the same cluster. This idea is relatively straightforward, yet what is needed is a principled way of deciding the correct number of clusters and the precise way of label transduction (e.g. based on majority vote vs. probability thresholds). Typically, such parameters are decided heuristically (e.g. [8]) or by tuning on a labeled development set; for resource-poor languages, however, no such set may be available.

As suggested by [10], the TC algorithm can utilize a theoretical error bound as a principled way of determining the parameters. Formally, we define the transductive learning problem as having a data sample $X_{m+u} = \{x_1, \ldots, x_{m+u}\}$, from which we draw randomly without replacement a training sample $X_m \subset X_{m+u}$ to receive the labels $y_1, ..., y_m$. The remaining set $X_u = X_{m+u} \backslash X_m$ is the unlabeled (test) sample. The goal is to find a labeling of X_u (a hypothesis h) from the set of all possible hypotheses H such that error on the test set (test risk) is minimized. Following the derivation by [6]: Let $R_h(X) = \frac{1}{M} \sum_{i=1}^{M} l(h(x_i), y_i)$ be the risk of some hypothesis h on data set X (of size M), with $l(h(x), y) \in [0, B]$ being a loss function (for B=1, this is the well-known 0/1 loss). It has been shown that with probability $1 - \delta$ over choices of X_m ($\delta \in (0, 1)$), the risk on the unlabeled data, $R_h(X_u)$, is bounded by

$$R_h(X_u) \leq \hat{R}_h(X_m) + B\sqrt{\left(\frac{m+u}{u}\right)\left(\frac{u+1}{u}\right)\left(\frac{\ln 1/p(h) + \ln 1/\delta}{2m}\right)} \tag{2}$$

i.e. the test risk is bounded by the empirical risk on the labeled data, $\hat{R}_h(X_m)$, plus a term that varies with the prior $p(h)$ of the hypothesis or classifier. This is an example of the PAC-Bayesian bound [17]. The prior $p(h)$ indicates ones prior belief on the hypothesis h over the set of all possible hypotheses. If the prior is low or the empirical risk is high, then the bound is large, implying that test risk may be large. A good hypothesis (i.e. classifier) will ideally have a small value for the bound, thus predicting a small expected test risk.

The PAC-Bayesian bound is important because it provides a theoretical guarantee on the quality of a hypothesis. Moreover, the bound in Eq. 2 is particularly useful because it is easily computable on any hypothesis h, assuming that one is given the value of the prior $p(h)$. Given two hypothesized labelings of the test set, h_1 and h_2, the one with the lower PAC-Bayesian bound will achieve a lower expected test risk. Therefore, one can use the bound as a principled way of choosing the parameters in the Transductice Clustering algorithm: First, a large numbers of different clusterings is trained; then the one that achieves the lowest PAC-Bayesian bound is chosen. The pseudo-code is given in Figure 2.

[10] has applied the Transductive Clustering algorithm successfully to binary classification problems and demonstrated improvements over the current state-of-the-art Spectral Graph Transducers (Section 3.2.3). Here we extend the algorithm to the structured output case as needed by our lexicon learning problem [9].

Extensions to Structured Outputs

In the situation where each data point can have $1, ..., K$ possible labels, there are three obvious ways of extending the pseudo-code in Figure 2 (for illustrative purposes, suppose $K = 3$ and the labels are A,B,C):

1. **Single-label framework: K binary learners:** Apply K independent transductive learners, each indicating whether a label is present or absent (ie. A vs. ¬A, B vs. ¬B, etc.). The final labeling is the combined output of the K learners. This has the obvious disadvantage of not exploiting dependencies between different labels; neither does it ensure that each data point receives at least one and fewer than K positive labels.

2. **Compound-label framework: one N-ary learner:** use all N label combinations observed on X_m (i.e. compound-labels) as individual labels (e.g. "A-B", "A-B-C", "B-C"). Apply one transductive learner as described previously. The potential drawback of this method is that not all possible combinations of labels may have been observed, particularly when the X_m is small.

3. **Extended-compound-label framework: Multi-label learner:** Apply one transductive learner. When labeling a cluster, do not only use the majority vote among labels in a given cluster but choose all labels above some frequency threshold. Suppose there are 3 A's, 5 B's, and 2 C's in total and the threshold is the top 80%. We then label all data points "A-B"; if it is top 50% or less, we use "B" only. We call this a extended-compound label framework because it the basic output labels are POS-sets like the compound-label, but additional POS-sets not originally in the training set may be hypothesized.

1 Apply a clustering algorithm to X_{m+u} to generate $C-1$ different partitions with τ clusters each ($2 \leq \tau \leq C$).
2 For each partition
- generate a label hypothesis h_τ that labels each cluster with the most frequent tag among its labeled data points
- calculate the bound for h_τ as defined in Eq. 2.
3 Output the classification of X_u using the hypothesis with the smallest bound.

Figure 2: Pseudo-code for transductive clustering.

In all three cases, the bound defined in Eq. 2 applies. In the first two cases, the 0/1 loss is used to compute the risk while in the third case the loss function is bounded by $[0, K]$. If uniform priors were used, $p(h)$ would be $1/(C-1)2^\tau$, $1/(C-1)n^\tau$, and $1/(C-1)2^{K\tau}$, respectively. As mentioned above, when n or K are large (as they are in practice), the prior probabilities will be too low and the bound will be too loose for effective model (hypothesis) selection. We therefore need to develop better ways of computing priors. Note that priors are also a way of integrating domain knowledge and dependencies between labels – this makes the PAC-Bayesian scheme more attractive than other model selection techniques such as BIC. Among the possible techniques we note:

- for case 1: using conditional priors to incorporate dependencies between learners (e.g. $p(h_B)|p(h_A)$). However, this assumes that labels are assigned in a fixed order (i.e. first the learner for A vs.¬A is applied, then the learner for B vs. ¬B, etc.), and the best ordering may be impossible to find.
- for cases 2 and 3: estimating priors for the joint occurrence of labels from the labeled data (though X_m may be small and priors therefore unreliable), or estimating the priors from some taxonomically similar language with more labeled data.

As we will show in our experiments on WSJ, estimating good priors is critical for applying Transductive Clustering on structured problems. In particular, we use the maximum likelihood estimate of the joint occurrence of labels from the training data as priors, which outperforms uniform priors.

3.2.2 Transductive SVM

Transductive SVM (TSVM) [13] is an algorithm that implicitly implements the cluster assumption. In standard inductive SVM (ISVM), the learning algorithm seeks to maximize the margin subject to misclassification constraints on the training samples. In TSVM, this optimization is generalized to include additional constraints on the unlabeled samples. The resulting optimization algorithm seeks to maximize the margin on both labeled and unlabeled samples.

3.2.3 Spectral Graph Transducer

Spectral Graph Transducers (SGT) [14] achieves transduction via an extension of the normalized mincut clustering criteria. First, a data graph is constructed where the vertices are labeled or unlabeled samples and the edge weights represent similarities between samples. The mincut criteria seeks to partition the graph such that sum of cut edges is minimized. SGT extends this idea to transductive learning by incorporating constraints that require samples of the same label to be in the same cluster. The resulting partitions decide the label of unlabeled samples.

4 Data

Experiments were performed on two data sets: Levantine Arabic, which represents a true resource-poor language, and Wall Street Journal (WSJ), for additional simulation and experimentation.

4.1 Levantine Arabic

4.1.1 Corpus

The dialect addressed in this work is Levantine Colloquial Arabic (LCA), primarily spoken in Jordan, Lebanon, Palestine, and Syria. Our development/test data comes from the Levantine Arabic CTS Treebank provided by LDC. The training data comes from the Levantine CTS Audio Transcripts. Both are from the Fisher collection of conversational telephone speech between Levantine speakers previously unknown to each other. The LCA data was transcribed in standard MSA script and transliterated into ASCII characters using the Buckwalter transliteration scheme[3]. No diacritics are used in either the training or development/test data. Speech effects such as disfluencies and noises were removed prior to our experiments.

The training set consists of 476k tokens and 16.6k types. It is not annotated with POS tags – this is the raw text we use to train the unsupervised HMM tagger. The test set consists of 15k tokens and 2.4k types, and is manually annotated with POS tags. We used the reduced tagset known as the Bies tagset [16], which focuses on major part-of-speech and excludes detailed morphological information. The tagset is listed in Table 4.1.1.

Symbol	Gloss	(%)
NN	noun	21.57
IN	preposition	12.35
UH	disfluency, interjection	10.66
PRP	pronoun	10.40
VBP	imperfect verb	9.34
RP	particle	7.00
JJ	adjective	6.73
CC	coordinating conjunction	3.45
PRP$	possessive determiner	3.32
NNP	proper noun	3.25
VBD	perfect verb	3.12
RB	adverb	2.61
DT	determiner	1.74
NNS	non-singular noun	1.23
WP	pronoun, question	1.14
CD	cardinal number	0.98
VB	verb, aux	0.72
NOFUNC	none	0.18
WRB	adverb, question	0.05
VBN	passive verb	0.03

Table 1: Bies tagset and percentage of occurrence of each tag in the test set.

4.1.2 Morphological Analyzer

We employ the LDC-distributed Buckwalter analyzer for morphological analyses of Arabic words. For a given word, the analyzer outputs all possible morphological analyses, including stems, POS tags, and diacritizations. The information regarding possible POS tags for a given word is crucial for constraining the unsupervised learning process in HMM taggers.

The Buckwalter analyzer is based on an internal stem lexicon combined with rules for affixation. It was originally developed for the MSA, so only a certain percentage of Levantine words can be correctly analyzed. Table 4.1.2 shows the percentages of words in the LCA training text that received N possible POS tags from the Buckwalter analyzer. Roughly 23% of the (types) and 28% of the tokens are received no tags (N=0) and are considered un-analyzable.

[3] http://www.ldc.upenn.edu/myl/morph/buckwalter.html

N	Type	Token
0	23.3	28.2
1	52.5	40.4
2	17.7	19.9
3	5.2	10.5
4	1.0	2.3
5	0.1	0.6

Table 2: Percentage of word types/tokens with N possible tags, as determined by the Buckwalter analyzer. Words with 0 tags are un-analyzable.

4.2 Wall Street Journal Data and Lexicon

The Wall Street Journal (WSJ) data is not resource-poor. We chose it for experimentation because it is widely available and allows us to simulate various sparse-data scenarios by artificially limiting the amount of training data. Further, this dataset allows us to directly evaluate lexicon accuracy since oracle POS-sets for all words in the lexicon can be obtained by the manual POS annotation on the training data.

The WSJ data has a word list of 44.5k words and the tagset size (K) of 46. We use the standard training, development and test definitions used in previous studies, i.e. sections 0-18 for training, 19-21 for development and 22-24 for testing. In order to create a baseline WSJ lexicon, we collected all the POS labels from the WSJ text. Since it has been shown that such method creates a somewhat low-quality lexicon for the purposes of unsupervised POS we create a second lexicon by filtering the possible POS labels for a word according to their frequency in the training corpus.[4] Using a 2nd order HMM tagger, the unfiltered lexicon achieves a tagging accuracy of 74.00% while the filtered lexicon achieves 90.16%. Note that the 2nd order HMM we employ is a simple word-based model with Dirichlet smoothing that uses only contextual knowledge and no extra word features, such a capitalization, affix information etc. Higher accuracies could be achieved by e.g. integrating word features, improving the smoothing technique, etc.. However, the goal of this study is not to develop the best possible WSJ POS tagger but to evaluate the effect of transductive learning on the tagging lexicon.

We sampled the above two lexicons in two different ways to create partial lexicons for simulating the sparse-data situation for transductive learning. In the first method, we perform random samplings of 5k, 10k, 20k, 30k, and 40k words of the original unfiltered/filtered lexicons. The selected words retain their labels extracted from the training corpus, while the labels for the remaining words are discarded. In the second case, we sampled the words based on their corpus frequency, since in practical settings, an annotator may choose to label the most frequent words first. Specifically, we sort the words by their count in the training corpus and took the top 5k, 10k, 20k, 30k, and 40k words. The sampling ranges of 5k to 40k simulate the effect of moving from a small labeled set to an almost fully labeled set.

5 System

We describe our overall system for Levantine Arabic (see Figure 3): In Step 1, the Buckwalter analyzer is applied to the word list derived from the raw training text. The result is a partial POS lexicon, which lists the set of possible POS tags for those words for which the analyzer provided some output. All possibilities suggested by the analyzer are included. [5]

The focus of Step 2 is to infer the POS-sets of the remaining, unannoted words using one of the automatic learning procedures described in Section 3. Finally, Step 3 involves training an HMM tagger using the learned lexicon. This is the standard unsupervised learning component of the system. We use a trigram HMM, although modifications such as the addition of affixes and variables modeling speech effects may improve tagging accuracy. Our concern here, is the evaluation of the lexicon learning component in Step 2.

A note about error propagation is needed. In Step 1, the MSA analyzer may give incorrect POS-sets to analyzable words. It may not posit the correct tag (low recall), or it may give too many tags (low precision). Both have a negative

[4] Naturally, this is not a viable method in practicical scenarios since tagged corpora is not available.

[5] Note that the analyzer outputs the possible POS tags for each word in isolation; finding the actual POS tag of a word in context requires a POS tagger.

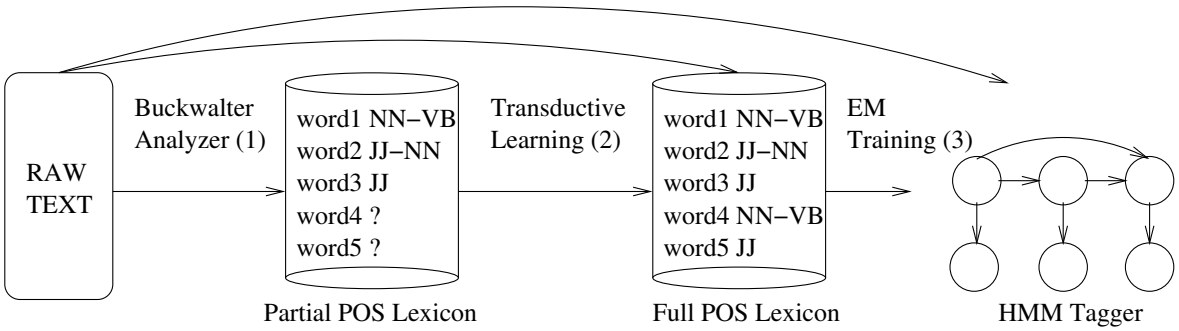


Figure 3: Overall System for dialectal Arabic: (1) Apply Buckwalter Analyzer to dialectal Arabic raw text, obtaining a partial POS lexicon. (2) Use Transductive Learning to infer missing POS-sets. (3) Unsupervised training of HMM Tagger using both raw text and inferred lexicon.

effect on lexicon learning and EM training. For lexicon learning, Step 1 errors represent corrupt training data; For EM training, Step 1 error may cause the HMM tagger to never hypothesize the correct tag (low recall) or have too much confusibility during training (low precision). We attempted to measure the extent of this error by calculating the tag precision/recall on words that occur in the test set: Among the 12k words analyzed by the analyzer, 1483 words occur in the test data. We used the annotations in the test data and collected all the "oracle" POS-sets for each of these 1483 words. The (micro)average precision of the analyzer-generated POS-sets against the oracle is 56.46%. The average recall is 81.25%. Note that precision is low–this implies that the partial lexicon is not very constrained. The recall of 81.25% means that 18.75% of the words may never receive the correct tag in tagging. In the experiments, we will investigate to what extent this kind of error affects lexicon learning and EM training.

The system for WSJ is similar, except that we construct the partial lexicon (Step 1) by randomly deciding which words are labeled and which words are not.

6 Experiments (Dialectal Arabic)

We seek to answer the following three questions in our experiments on dialectal Arabic:

- How useful is the lexicon learning step in an end-to-end POS tagging system? Do the machine learning algorithms produce lexicons that result in higher tagging accuracies, when compared to a baseline lexicon that simply hypothesizes all POS tags for un-analyzable words? The answer is a definitive **yes**.
- What machine learning algorithms perform the best on this task? Do transductive learning outperform inductive learning? The empirical answer is that TSVM performs best, SGT performs worst, and TC and ISVM are in the middle.
- What is the relative impact of errors from the MSA analyzer on lexicon learning and EM training? The answer is that Step 1 errors affect EM training more, and lexicon learning is comparably robust to these errors.

In our problem, we have 12k training samples and 3970 test samples. Note that we have more training samples than test samples, a situation different from typical semi-supervised or transductive learning scenarios. We define the feature of each sample as listed in Table 3. The contextual features are generated by co-occurence statistics gleaned from the training data. For instance, for a word `foo`, we collect all bigrams consisting of `foo` from the raw text; all features [w_{t-1} = `voc`] that correspond to the bigrams (`voc`, `foo`) are set to 1. The idea is that words with similar orthographic and/or contextual features should receive similar POS-sets.

All results, unless otherwise noted, are tagging accuracies on the test set given by training a HMM tagger on a specific lexicon. Table 4 gives tagging accuracies of the four machine learning methods (TSVM, TC, ISVM, SGT) as well as two baseline approaches for generating a lexicon: (all tags) gives all 20 possible tags to the un-analyzable words, whereas (open class) gives only the subset of open-class POS tags. The TC algorithm uses a N-ary classifier with priors estimated from joint occurrences of POS tags in the training data. The results are given in descending order

Orthographic features:
w_i matches /^pre/, pre = {set of data-derived prefixes}
w_i matches /suf$/, suf = {set of data-derived suffixes}
Contextual features:
w_{i-1} = voc, voc = {set of words in lexicon}
t_{i-1} = tag, tag = {set of POS tags}
t_{i+1} = tag, tag = {set of POS tags}
w_{i-1} is an un-analyzable word
w_{i+1} is an un-analyzable word

Table 3: Binary features used for predicting POS-sets of un-analyzable words.

of overall tagging accuracy. With the exception of TSVM (63.54%) vs. TC (62.89%), all differences are statistically significant. As seen in the table, applying a machine learning step for lexicon learning is a worthwhile effort since it always leads to better tagging accuracies than the baseline methods.

Method	Accuracy	UnkAcc
TSVM	63.54	26.19
TC	62.89	26.71
ISVM	61.53	27.68
SGT	59.68	25.82
open class	57.39	27.08
all tags	55.64	25.00

Table 4: Tagging Accuracies for lexicons derived by machine learning (TSVM, TC, ISVM, SGT) and baseline methods. Accuracy=Overall accuracy; UnkAcc=Accuracy of unknown words.

The poor performance of SGT is somewhat surprising since it is contrary to results presented in other papers. We attributed this to the difficulty in constructing the data graph, which is an added-level of parameter required for tuning. In this experiment, we constructed kNN graphs for SGT using the cosine distance between feature vectors; the parameters that can be tuned include the distance metric, the value of k in kNN, etc. It is not straightforward to tune so many parameters in a resource-poor scenario. Finally, we note that besides the performance of SGT, transductive learning methods (TSVM, TC) outperform the inductive ISVM.

We also compute precision/recall statistics of the final lexicon on the test set words (in a fashion similar to Section 5) and measure the average size of the POS-sets (||POSset||). As seen in Table 5, POS-set sizes of machine-learned lexicon is a factor of 2 or 3 smaller than that of the baseline lexicons. On the other hand, recall is better for the baseline lexicons. These observations, combined with the fact that machine-learned lexicons gave better tagging accuracy, implies that we have a *constrained* lexicon effect here: i.e. in the context of EM training, it is better to constrain the lexicon with small POS-sets than to achieve high recall.

Method	Precision	Recall	||POSset||
TSVM	58.15	88.85	1.89
TC	59.19	87.88	1.80
ISVM	58.09	88.44	1.87
SGT	53.98	82.60	1.87
open class	54.03	96.77	3.39
all tags	53.31	98.53	5.17

Table 5: Statistics of the Lexicons in Table 4.

Next, we examined the effects of error propagation from the MSA analyzer in Step 1. We attempted to correct these errors by using POS-sets of words derived from the development data (which is annotated with export POS

information). In particular, of the 1562 partial lexicon words that also occur in the development set, we found 1044 words without entirely matching POS-sets. These POS-sets are replaced with the oracle POS-sets derived from the development data, and the result is treated as the (corrected) partial lexicon of Step 1. In this procedure, the average POS-set size of the partial lexicon decreased from 2.13 to 1.10, recall increased from 82.44% to 100%, and precision increased from 57.15% to 64.31%. We apply lexicon learning to this corrected partial lexicon and evaluate tagging results, shown in Table 6. The fact that all numbers in Table 6 represent significant improvements over Table 4 implies that error from the MSA analyzer is not trivial, and automatic error correction methods may be desired.

Method	Accuracy	UnkAcc
TSVM	66.54	27.38
ISVM	65.08	26.86
TC	64.05	28.20
SGT	63.78	27.23
all tags	62.96	27.91
open class	61.26	27.83

Table 6: Tag accuracies by correcting mistakes in the partial lexicon prior to lexicon learning. Interestingly, we note ISVM outperforms TC here, which differs from Table 4.

Finally, we determine whether error propagation impacts lexicon learning (Step 2) or EM training (Step 3) more. Table 7 shows the results of TSVM for four scenarios: correcting analyzer errors in the the lexicon: (A) prior to lexicon learning, (B) prior to EM training, (C) both, or (D) none. As seen in Table 7, correcting the lexicon at Step 3 (EM training) gives the most improvements, indicating that analyzer errors affect EM training more than lexicon learning. This implies that lexicon learning is relatively robust to training data corruption, and that one can mainly focus on improved estimation techniques for EM training [20] if the goal is to alleviate the impact of analyzer errors. The same evaluation on the other machine learning method (TC, ISVM, SGT) show similar results.

Scenario	Step2	Step3	TSVM
(B)	N	Y	66.70
(C)	Y	Y	66.54
(A)	Y	N	64.93
(D)	N	N	63.54

Table 7: Effect of Correcting the Lexicon in different steps. Y=yes, lexicon corrected, while N=no, POS-set remains the same as analyzer's hypothesis.

7 Experiments (Wall Street Journal)

The purpose of the WSJ experiments is to examine the effects of lexicon learning in a controlled setting. We present three sets of experiments:

1. Lexicon Evaluation: We directly measure lexicon accuracy/F1-score since the oracle POS-sets are available from the annotated training data. Here, we examined various implementations of the Transductive Clustering algorithm, in particular comparing (1) k-binary learner vs. N-ary learner vs. multi-label learner; and (2) uniform vs. joint priors. (Section 7.1)
2. Tagger Evaluation: We present the tagger accuracies for the lexicons in the Lexicon Evaluation experiment. (Section 7.2)
3. Comparison of Algorithms: Here we also examine TSVM, SGT, and ISVM and show interesting results where transductive learning does not seem to lead to better lexicon accuracy. (Section 7.3)

In the Appendices we show tuning experiments for ISVM, TSVM, and SGT to illustrate the effect of several different parameters. In the following three sections, all results are presented using the best-tuned classifiers.

	Accuracy					F1-score				
m	oracle	ER	n-uni	n-joint	multi	oracle	ER	n-uni	n-joint	multi
5k	49.54	39.59	44.16	44.87	**47.36**	56.04	48.29	50.17	51.27	**54.68**
10k	50.70	43.91	44.47	**48.05**	47.22	57.69	51.91	50.72	54.56	**54.76**
20k	51.75	48.06	44.37	**48.37**	47.25	58.74	54.28	50.88	**55.03**	54.80
30k	52.35	**49.61**	44.30	48.37	47.26	59.01	**56.61**	50.67	54.88	55.01
40k	53.57	**51.81**	45.16	45.16	48.24	60.38	**58.21**	51.43	51.43	55.75

Table 8: **Accuracy/F1-score comparison:** [m]=# labeled data. [oracle]= oracle performance when selecting the hypothesis with the lowest risk on the test data . [ER]= selection based on the empirical risk on the training data. [n-uni]= n-ary learner with uniform prior. [n-joint]=n-ary learner with estimated joint prior. [multi]=multi-label with estimated joint prior. Best results are in bold. These results are for unfiltered lexicon with random sampling.

7.1 WSJ Lexicon Evaluation

First we present experimental results on 5k, 10k, 20k, 30k, and 40k samplings of the unfiltered full lexicon and evaluated the resulting lexicon generated by transductive clustering by calculating its accuracy/F-score on the full lexicon. The full sample is partitioned into 10, 20, 30, ..., 2000 clusters using the Brown clustering algorithm [3]. Our main goals are to determine (a) whether the PAC-Bayesian model selection criterion is more effective than choosing a hypothesis based on $\hat{R}_h(X_m)$ alone, and (b) which of the structured extensions described in Section 3.2.1 work best. Results are reported as the accuracy on concatenated tags and F1-score on the individual tags, given the full lexicon as truth (e.g. If reference="A-B" & decision="A", then accuracy=0; precision=1, recall=0.5, F1-score=0.67).

Figure 4 illustrates the curves for the bound, empirical risk, and test risk varied by the number of clusters. In general, the empirical risk decreases with the number of clusters, so choosing hypotheses simply based on empirical risk may result in overfitting. The bound curve follows the test risk curve and therefore results in a satisfactory (although not best) test risk.

Table 8 compares the lexicon accuracy/F-score of (1) n-ary with uniform prior [n-uni], (2) n-ary with estimated joint label priors [n-joint], and (3) multi-label with joint priors [multi]. The column [oracle] shows the best possible accuracy that can be achieved among all proposed hypotheses; the column [ER] reports the accuracy for the case where the hypothesis selection is based on the minimum empirical risk on the labeled data. (The former is an upper limit to the performance of the transductive clustering scheme while the latter indicates the usefulness of the prior-based penalty term in the PAC-Bayesian bound).

Not surprisingly, the results highlight the importance of using good priors: Both methods that utilize priors over combinations of labels estimated from X_m ([n-joint],[multi]) outperform [n-uni]. We also note that [multi] tends to have a better F1-score but lower accuracy when compared to [n-joint], which is expected due to their different label-assignment schemes. Since no method is the clear winner, it is important to choose the proper evaluation criteria as one that best matches the goals of the downstream application/user of the lexicon. Finally, it is interesting to note that all three PAC-Bayesian learners outperform [ER] when the number of labeled data is small ($m << u$), but the usefulness of the bound decreases as the m increases, indicating that the empirical risk is actually a good indicator of test risk in those cases.

We also experimented with the K-binary case. Due to the unbalanced data problem created by the binary decomposition (i.e. the number of "¬A" labels far exceeds "A" labels), 40% of words received no positive labels from any of the k binary learners. This demonstrates the problem that arises when decomposing a multi-label problem without regard to its structure. The 60% of test words that were classified achieved an accuracy of 60.0% to 61.7% and F1-score of 69.7% to 72.5%.

7.2 WSJ Tagger Evaluation

For tagging experiments, we present tag accuracies for different taggers using the lexicons generated by [oracle], [n-joint], and [multi]. We compared the result against a baseline lexicon that hypothesizes all possible tags for each unlabeled word, which is the only possibility in the absence of lexical information. The main questions to answer are (1) Does transductive clustering create lexicons that outperform the baseline lexicon? (2) Which of the transductive clustered lexicons gives the best tagging accuracy? (3) What are the effects of filtered vs. unfiltered full lexicons, and

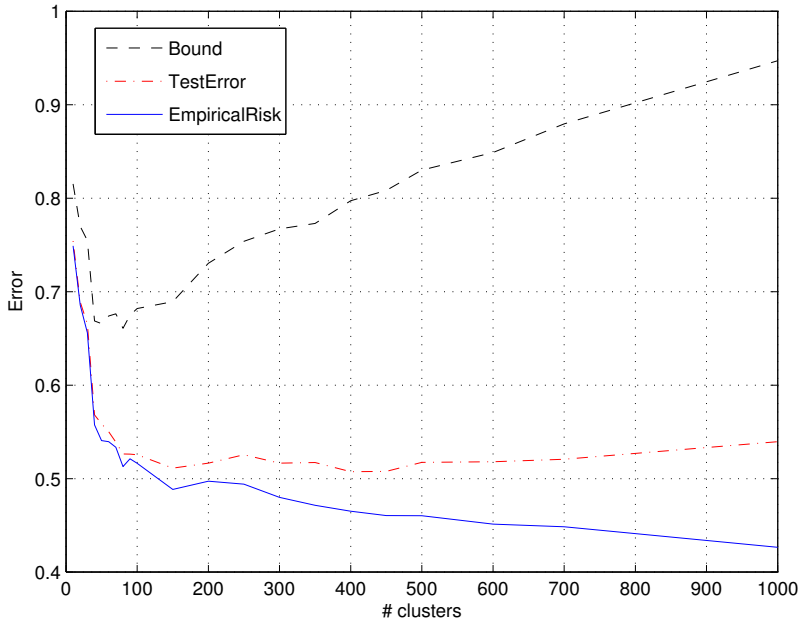


Figure 4: Plot of PAC-Bayesian bound, empirical risk on X_m, and actual test error on X_u for different number of clusters (different hypotheses). (Multi-label, m=5k)

random vs. frequency sampling on tagging accuracy? The results are presented in the four cases (unfiltered+random, unfiltered+freqeuncy, filtered+random, filtered+frequency).

Table 9 shows the tagging accuracies for the unfiltered lexicon with labeled words generated by random sampling. This corresponds to the lexicon accuracy/F-score results shown in Table 8. First, note that the performance of [baseline] significantly degrades as the number of labeled word decreases. This is because the lack of constraints reduces the EM algorithm's effectiveness for tagging. In contrast, the accuracy degradation for the transductive learners is much less severe. This points to the effectiveness of using transductive learning in extreme sparse data (e.g. m = 5k to 20k out of 44.5k) conditions.

It is also important to note that the baseline performs *better* than the transductive learners when $m >> u$ (e.g. the 40k case). The reason can be explained by a precision-recall tradeoff on the POS tags hypothesized on the unlabeled words. Low recall influences tagging accuracy since the correct tag of a word is never known to the tagger; low precision, on the other hand, increases confusion and affects the effectiveness of EM training. When the majority of words are already labeled (e.g. 40k), the tendency for EM confusion is low, and therefore recall is more important than precision; the oppositive is true for sparse data situations. Therefore, a straightforward way to improve the performance of transductive learners in the 40k situation is to improve the recall–this can be accomplished naturally in the [multi] case by adjusting the threshold of tags for labeling each cluster. The best threshold is not known a priori, but the same PAC-Bayesian model selection criteria can be applied – one can simply try various thresholds (similar to trying various number of clusterings), and the one that achieves the lowest bound is chosen. This is a very important theoretical result, and it points to the generalizability of learning with a PAC-Bayesian framework.

Table 10 shows the results for frequency sampling of the unfiltered lexicon. Compared to random sampling (Table 9), the tagging accuracies are much higher. As expected, the differences between the transductive learners as well as the baseline technique becomes closer, since the lexicons are only differ in the labels for the less frequent words.

Tables 11 and 12 present the results for a filtered lexicon. Comparing these two tables to the ones for unfiltered lexicon (Tables 9, 10), we observe that accuracies for the filtered lexicon are in general higher (as expected). However, an interesting opposite result occurs for m=5k,10k for random sampling. (e.g. The 5k-[multi] achieves only 49.83% in Table 11 compared to 50.52% in Table 9.) We believe this is, again, a result of the precision-recall tradeoff. The filtered lexicon contains fewer tags per word, and therefore a transductive learner based on it will have lower recall

m	oracle	n-joint	multi	baseline
5k	51.07	45.90	50.52	17.92
10k	53.95	51.60	53.70	29.34
20k	61.93	60.40	60.43	48.70
30k	67.42	66.32	65.99	63.98
40k	70.47	68.92	70.08	71.85

Table 9: Random Sampling, Unfiltered Lexicon

m	oracle	n-joint	multi	baseline
5k	68.98	68.67	68.98	67.57
10k	71.04	71.07	71.12	70.39
20k	72.82	72.63	72.69	71.99
30k	73.47	73.98	73.57	73.72
40k	73.86	73.83	73.86	74.14

Table 10: Frequency Sampling, Unfiltered Lexicon

(but possibly higher precision). Since the clustering algorithm is not perfect and the filtered lexicon has fewer tags per word to begin with, transductive clustering may not give unlabeled words any of it true labels (low recall).

Finally, we observe for all tables that there is no clear winner in terms of tagging accuarcy between [n-joint], and [multi], although differences in lexicon accuracy/F-score are statistically signficant. Some differences in the four tables are statistically significant, while some are not, and we do not observe any general trend. The lesson is that the interaction of lexicons and unsupervised learning is complex and in general any of the transductive clustering schemes proposed above would work in practice.

7.3 Comparisons of Algorithms (TC, ISVM, TSVM, SGT)

In this section, we evaluate lexicon accuracy for a variety of transductive and inductive algorithms. All experiments are performed on the 5k labeled WSJ dataset. First, Table 14 shows the lexicon accuracies of the following algorithms:

1. TC-Brown: Tranductive clustering using clusters obtained by the Brown algorithm
2. TC-Feature: Transductive clustering using clusters obtained by an bottom-up agglomerative clustering algorithm with pairwise distances depending on word and context features. These are the same feature sets used in the SVM and SGT learners, so TC-Feature can be compared to SVM/SGT directly. The set of features are listed in Table 13
3. ISVM: inductive SVM
4. TSVM: transductive SVM
5. TSVM-r: transductive SVM (cheating version using oracle positive/negative ratio). The TSVM algorithm requires a parameter, r, which is the ratio of positive/negative examples in the unlabeled test set. The algorithm will maintain this ratio when making the classification. In default TSVM, this ratio is estimated by the percentage of positive vs. negative samples in the training example. Since this is a critical tuning factor that may affect accuracy, TSVM-r is a cheating experiment that uses the oracle ratio measured from the test set.
6. SGT: spectral graph transducer

By comparing the lexicon accuracies, we conclude that: (1) TC-Feature outperforms TC-Brown, so adding features to improve the clustering result is a worthwhile endeavor. (2) the difference between ISVM and TSVM is not significant, so transductive learning does not necessarily outperform inductive learning in the particular implementation used here. (3) TSVM-r is not significantly better than TSVM, implying that the use of an estimated ratio in the default TSVM is not the reason for its unsatisfactory performance.

Why does TSVM not outperform ISVM? We conjecture four hypotheses:

m	oracle	n-joint	multi	baseline
5k	50.52	50.51	49.83	34.83
10k	52.29	52.98	49.49	47.03
20k	62.40	62.01	61.99	63.40
30k	71.95	71.13	71.11	72.61
40k	83.90	84.11	84.18	84.79

Table 11: Random Sampling, Filtered Lexicon

m	oracle	n-joint	multi	baseline
5k	83.03	83.07	82.52	81.78
10k	84.87	84.23	84.95	83.57
20k	86.53	86.56	86.56	86.06
30k	87.40	87.25	87.24	87.14
40k	87.79	87.66	87.67	87.78

Table 12: Frequency Sampling, Filtered Lexicon

1. The TSVM implementation of [13] uses a greedy method for adjusting the TSVM hyperplane. This may be sub-optimal, leading to poor results. A potential method to test this hypothesis is to try a different TSVM implementation, such as [5] which uses a convex-concave procedure as a relaxation to the discrete optimization problem.

2. The data distribution does not lend itself to transductive learning. For example, the cluster assumption may be violated, e.g. the positive/negative samples may be highly overlapped in feature space. We attempted to see whether our samples are overlapped by mapping the 50k-long feature vectors to a 2-dimension plane via Sammons mapping. The results showed overlapped positive/negative samples, but were inconclusive because we cannot determine whether the overlap was due to the inherent property of the data, or the artifacts of Sammons mapping.

3. The transductive learner may be maximizing the "wrong margin". The idea of maximizing the wrong margin comes from [21]. Figure 5 shows a schematic illustration of this. In short, the hypothesis is that in our one-vs-rest scheme, the negative samples are distributed like a mixture of clusters, where each cluster may a specific compounded label. There exists many low density regions within these clusters, so the TSVM has a tendency to move the hyperplane inside the negative samples' region.

4. Individual TSVM classifiers may outperform individual ISVM classifiers, but the one-vs-rest output for TSVM may not necessariky outperform the one-vs-rest output for ISVM. For instance, consider a sample that is labeled by each individual ISVM as negative; this sample is labeled positive correctly by one individual TSVM and labeled negative correctly by the remaining TSVMs. In the one-vs-rest scheme, we take the argmax as the output decision, even if all classifiers labeled the sample negative. Thus, if in the ISVM case, the classifier with the least negative output happens to be the correct one, then the argmax decision of ISVM will be just as good as the TSVM. In other words, this hypothesis states that the argmax operation of one-vs-rest may interact with the binary classifiers such that the correct output may be predicted even when all classifiers hypothesize incorrectly individually.

We test hypothesis 3 above by simulating the "maximizing the wrong margin" scenario on a simple 2-dimension plane. In Figures 6(a) and 6(b), the positive and negative samples are distributed on the right and left halves of the plane, respectively. The oracle hyperplane with zero classification error occurs at index=50. The simulate the low density region formed by mixture of clusters in the negative samples, there is a gap between index 20 and 30 that contains no samples. The center region (index 30 to 50) are negative samples. In Figure 6(a), some of these center samples are labeled, whereas in Figure 6(b) none of these samples are labeled. According to hypothesis 3, the TSVM hyperplane should move to the low density region between index 20 and 30, which maximizes the margin (the wrong margin). Contrary to the hypothesis, this does not occur in either Figures 6(a) nor 6(b). As a result, the TSVMs in

Orthographic features:
w_i matches /^pre/, pre = {set of data-derived prefixes}
w_i matches /suf$/, suf = {set of data-derived suffixes}
w_i matches /[A-Z]/ (capitalization feature)
w_i matches /[A-Z]+/ (acronym feature)
w_i matches /[A-Z][a-z]+/
w_i matches /[A-Z]+[a-z]+[A-Z]+[a-z]+/
w_i matches /[0-9]/ (number feature)
w_i matches /-/ (has-hyphen feature)
w_i matches /[&%$]/ (non-standard character feature)
number of characters in w_i (length feature)
Contextual features:
w_{i-1} = voc, voc = {set of words in lexicon}
t_{i-1} = tag, tag = {set of POS tags}
t_{i+1} = tag, tag = {set of POS tags}
w_{i-1} is an un-labeled word
w_{i+1} is an un-labelede word

Table 13: WSJ Features

Method	Acc
TC-Brown	47.36
TC-Feature	52.87
ISVM	69.84
TSVM	69.95
TSVM-r	70.10
SGT	63.26

Table 14: Comparison of WSJ Lexicon Accuracies for TC, ISVM, TSVM, and SGT. TC-Brown uses Brown clustering, whereas TC-Feature uses agglomerative clustering with the same feature set as the SVMs and SGT. TSVM-r uses the oracle positive/negative ratio and is a cheating result.

these two figures outperform the ISVMs. Post-hoc analysis revealed that the reason for this effect was the use of the positive/negative ratio in the TSVM implementation. Since the TSVM maintains a constant positive/negative ratio on the test samples, it will not move the hyperplane entirely to the low density region (this would violate the ratio). Thus, this ad-hoc parameter, which has been argued as a disadvantage of the TSVM [14], turns out to be a factor that maintains the robustness of the algorithm. Although hypothesis 3 was not validated, we believe the "maximizing the wrong margin" effect or other similar effects may still exist in the actual data. Additional analysis on the interaction between actual data distribution and the assumptions of transductive algorithms is needed.

Hypothesis 4 states that the poor lexicon accuracy of TSVMs may be due to the multi-class implementation that chooses the argmax among various individual binary classifiers. To test this hypothesis, we first measured the (micro)averaged precision and recall of the binary TSVM and ISVM classifiers. (In other words, we compute precision/recall for, e.g., the "NN" classifier, "RB" classifier, etc. and take the average.) As shown in Table 15, TSVM classifiers do outperform ISVM classifiers individually. The results imply that the ISVM one-vs-rest scheme must be somehow robust to misclassifications in individual ISVM classifiers; in other words, even though a sample may receive no positives from the ISVM classifiers, the least negative ISVM classifier is the correct label (and therefore the argmax picks it). To see whether this is the case, we computed the one-vs-rest outputs differently based on three cases and show lexicon accuracy results in Table 16:

1. Do not take the argmax of all classifiers – if a sample is hypothesized negative by all classifiers, the output label is automatically counted as incorrect

2. Ignore all samples hypothesized negative by all classifiers in the lexicon accuracy calculation

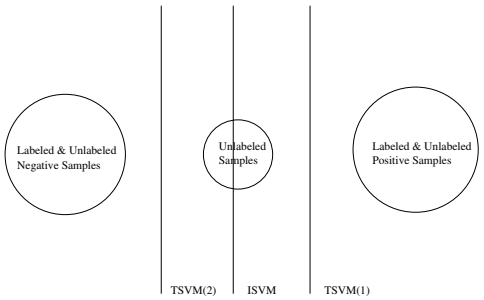


Figure 5: Example of "maximizing the wrong margin". The circles represent high density regions. An ISVM using only labeled data will place a hyperplane in the middle, whereas a TSVM using unlabeled data will shift the hyperplane to a low density region. It may be shifted to the left or right (TSVM1 or TSVM2) depending the particular locations of the samples. Suppose the unlabeled point in the center circle are all evenly split between positive and negative– then ISVM will outperform TSVM. However, if the unlabeled points in the center are all negative, TSVM may be drastically better or worse than ISVM depending on whether the TSVM1 or TSVM2 hyperplane is achieved. We suspect that the unlabeled points in the center are all negative in our data because we use an one-vs-rest scheme in our implementation. In this scheme, all samples that do not come from the target class as are negative samples. Therefore, one can imagine that the negative sample distribution actually looks like a mixture of clusters, where each individual cluster represents one of the multiple compound labels. In this mixture of clusters, we have have many low-density regions *within* the negative samples (rather than *between* the negative and positive samples). This may therefore lead to hyperplanes like TSVM(2), which will misclassify the entire center circle.

3. Ignore the subset of samples that are hypothesized negative by all TSVM and ISVM classifiers in the lexicon accuracy calculation. This differs from Case 2 in that it ensures that lexicon accuracy for TSVM and ISVM are calculated on the same set of words.

	ISVM	TSVM
Average Precision	60.81	63.65
Average Recall	43.11	46.25

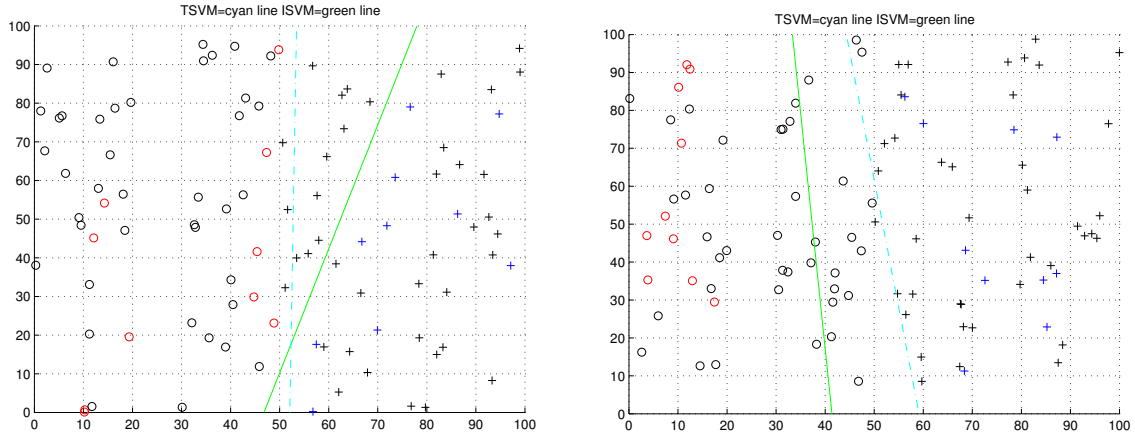
Table 15: Average Precision/Recall of ISVM and TSVM classifiers in the one-vs-rest scheme. The results imply that the TSVMs are individually better than the ISVMs. This gives supporting evidence for hypothesis 4.

In Case 1, TSVM outperforms ISVM; however, in Case 2 and 3, ISVM outperforms TSVM. This implies that ISVM is correctly predicting the samples that are hypothesized negative by all classifiers, relative to the TSVM. Thus, we conclude that hypothesis 4 is one of the reasons TSVM does not outperform ISVM in these WSJ experiments. Further analysis is required to understand why this happens.

8 Related Work

There is an increasing amount of work in NLP tools for Arabic. [7] achieves high accuracy on MSA with the direct application of SVM classifiers. [12] argues that the rich morphology of Arabic necessitates the use of a morphological analyzer in combination with POS tagging. This can be considered similar in spirit to the learning of lexicons for unsupervised tagging.

The work done at a recent JHU Workshop [4, 18] is very relevant in that it investigates a method for improving Levantine tagging that is orthogonal to our approach. Their approach is based on adapting a supervised tagger/parser trained on MSA. They do not use the raw Levantine text as we have done in this paper, and it would be very interesting to investigate the combination of our methods.



(a) (b)

Figure 6: Testing the "Maximizing the wrong margin" hypothesis. Simulation on 2-dimension plane. The positive samples (+) are on the right half, while the negative samples (o) are on the left half. The labeled samples are colored (red or blue); unlabeled samples are black. Cyan line = TSVM hyperplane; Green line = ISVM hyperplane. In these simulations, the hypothesis is not validated–the TSVM hyperplane does not go to the low density region (index 20-30) due to the enforcement of the positive/negative ratio.

	Lexicon Accuracy		Num Samples	
	ISVM	TSVM	ISVM	TSVM
Case 1	41.97	44.76	39509	39509
Case 2	79.81	79.54	20766	22224
Case 3	79.90	79.43	20691	20691

Table 16: Different lexicon accuracy calculations to examine the effects of samples labeled negative by all individual TSVM/ISVM classifiers. Num Samples = number of samples used to calculate the lexicon accuracy number (i.e. denominator). Note that (Case 2) the percent of samples labeled negative in ISVM (47.45%) is larger than that of TSVM (43.73%), yet TSVM does not outperform ISVM.

9 Conclusion

In this study, we investigated several machine learning algorithms on the task of lexicon learning and demonstrated the impact of this task on dialectal Arabic tagging. Future work includes a more detailed analysis of transductive learning in the WSJ dataset and possible solutions to alleviating error propagation in the dialectal Arabic dataset. Further, as a supervised HMM tagger trained on the development set (2000 sentences) achieves a 79% accuracy, there is still much research that needs to be done regarding the effective use of transductive, semi-supervised, and unsupervised techniques for resource-poor languages.

References

[1] M. Banko and R. Moore. Part-of-speech tagging in context. In *Proc. of COLING 2004*, 2004.

[2] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proc. of the Third Workshop on Very Large Corpora*, 1995.

[3] P.F. Brown et al. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[4] David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safi Shareef. Arabic dialect parsing. In *EACL*, 2006.

[5] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, 2006.

[6] P. Derbeko, R. El-Yaniv, and R. Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *JAIR*, 2004.

[7] M. Diab, K. Hacioglu, and D. Jurafsky. Automatic tagging of Arabic text: from raw text to base phrase chunks. In *Proceedings of HLT/NAACL*, 2004.

[8] K. Duh and K. Kirchhoff. POS tagging of dialectal arabic: a minimally-supervised approach. In *ACL 2005, Semitic Languages Workshop*, 2005.

[9] Kevin Duh and Katrin Kirchhoff. Structured multi-label transductive learning. In *NIPS Workshop on Advances in Structured Learning for Text/Speech Processing*, 2005.

[10] R. El-Yaniv and L. Gerzon. Effective transductive learning via objective model selection. *Pattern Recognition Letters*, 2005.

[11] A. Elisseeff and J. Weston. Kernel methods for multi-labelled classification and categorical regression problems, 2001.

[12] Nizar Habash and Owen Rambow. Arabic tokenization, morphological analysis, and part-of-speech tagging in one fell swoop. In *ACL*, 2005.

[13] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[14] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

[15] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6, 1992.

[16] Mohamed Maamouri, Ann Bies, and Tim Buckwalter. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR Conf. on Arabic Language Resources and Tools*, 2004.

[17] D. McAllester. Some PAC-bayesian theorems. *Machine Learning*, 37(3):255-36, 1999.

[18] Owen Rambow et al. Parsing Arabic dialects. Technical report, Final Report, 2005 JHU Summer Workshop., 2005.

[19] V. Vapnik. *Statistical Learning Theory*. Wiley Interscience, 1998.

[20] Qin Iris Wang and Dale Schuurmans. Improved estimation for unsupervised part-of-speech tagging. In *IEEE NLP-KE*, 2005.

[21] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *ICML*, 2000.

Appendix 1: Tuning TSVMs and ISVM

We present tuning TSVM and ISVM results on the 5k labeled WSJ dataset. In the ISVM, there is a single tuning parameter, c, that indicates the cost of training errors. In the TSVM, there is an additional parameter, d, which adjusts the training error of unlabeled samples. The TSVM objective function is as follows:

$$\frac{1}{2}||w||^2 + c\sum_{i=1}^{m}\xi_i + c * d\sum_{j=1}^{u}\xi_j \tag{3}$$

where ξ are the slack variables. We found that the TSVM implementation in SVMlight actually sets $d = 1$, although the paper [13] notes that d and c are separate user-defined variables. (More specifically, SVMlight begins with a small value of $c * d$ and increments it until it reaches c.) We modified the code to allow for user-specified d values. Table 17 shows the lexicon accuracy obtained by tuning c and d using a linear kernel SVM. As seen in this table, the ISVM outperforms all TSVMs for any given c, and TSVMs with smaller d seem to perform better than TSVMs with larger d. In other words, increasing the impact of unlabeled samples (i.e. increasing d) degrades the accuracy.

c	d=0	d=0.2	d=0.4	d=0.6	d=0.8	d=1
0.005	63.96	56.18	55.61	55.21	55.01	55.44
0.010	65.04	59.46	59.3	58.89	58.9	58.74
0.05	69.39	64.32	63.53	62.8	62.77	62.7
0.1	70.01	65.58	65.52	65.67	65.47	65.1
0.5	68.9	67.2	66.27	66.17	65.84	65.69
1	68.57	65.81	65.53	65.69	65.62	65.61

Table 17: Tuning c and d for linear kernel ISVM/TSVM. ISVM results are shown in the d=0 column.

We also performed similar tuning experiments on different kernels. In Table 18, we show lexicon accuracies for linear kernels and polynomial kernels of degrees two and three. The conjecture is that additional unlabeled data may require a more flexible classifier class. However, the results show the contrary: TSVM does not improve relative to ISVM in the presence of polynomial kernels.

kernel	d=0	d=0.0001	d=0.001	d=0.005	d=0.01	d=0.05	d=0.1
linear	70.01	70.17	69.14	68.07	67.74	66.17	65.78
poly2	68.54	66.81	67.86	67.25	58.15	64.34	64.2
poly3	66.21	63.92	61.51	60.17	60.6	60.77	60.61

Table 18: Effects of different kernels in tuning TSVM. c is set to 0.1 in this table. linear = linear kernel; poly2 = polynomial kernel, degree 2; poly3 = polynomial kernel, degree 3.

Appendix 2: Tuning SGT

We present tuning SGT results on the 5k labeled WSJ dataset. We varied three parameters:

1. kk: number of neighbors in k nearest neighbor graph construction
2. dd: number of eigenvectors to use in the spectral relaxation
3. c: cost of training error.

The lexicon accuracy results are shown in Table 19. The table shows that larger c, dd, and kk all lead to better accuracies, although c has the most significant impact.

kk	dd	c	accuracy
10	50	0.1	33.36
10	50	1.0	36.5
10	50	10	46.51
10	100	0.1	33.33
10	100	1.0	33.46
10	100	10	37.79
10	200	1	36.61
10	200	10	46.73
10	200	100	46.27
10	200	500	59.16
10	200	1000	59.5
10	200	5000	60.17
10	200	10000	60.52
30	50	1	35.62
30	50	10	45.48
30	50	100	55.15
30	50	500	57.81
30	50	1000	59.33
30	200	1	36.23
30	200	10	45.56
30	200	100	56.18
30	200	500	58.56
30	200	1000	59.34
30	500	1	35.62
30	500	10	45.35
30	500	100	56.24
30	500	500	58.82
30	500	1000	59.63
30	200	5000	59.96
30	200	10000	61.69
100	50	1	36.99
100	50	10	53.98
100	50	100	56.53
100	50	500	58.2
100	50	1000	57.8
100	200	1	36.71
100	200	10	54.11
100	200	100	57.48
100	200	500	58.82
100	200	1000	58.95
100	500	1	36.71
100	500	10	54.11
100	500	100	57.43
100	500	500	59.2
100	500	1000	58.49
100	200	5000	61.85
100	200	10000	63.26

Table 19: Tuning SGT. kk = number of nearest neighbors in graph constructure; dd = number of eigenvectors in spectral relaxation; c = cost of training error; accuracy = lexicon accuracy. The table shows that larger c, dd, and kk all lead to better accuracies, although c has the most significant impact.