Morphology Induction From Term Clusters

Dayne Freitag
HNC Software, LLC
3661 Valley Centre Drive
San Diego, CA 92130, USA
daynefreitag@fairisaac.com

Abstract

We address the problem of learning a morphological automaton directly from a monolingual text corpus without recourse to additional resources. Like previous work in this area, our approach exploits orthographic regularities in a search for possible morphological segmentation points. Instead of affixes, however, we search for affix transformation rules that express correspondences between term clusters induced from the data. This focuses the system on substrings having syntactic function, and yields cluster-to-cluster transformation rules which enable the system to process unknown morphological forms of known words accurately. A stem-weighting algorithm based on Hubs and Authorities is used to clarify ambiguous segmentation points. We evaluate our approach using the CELEX database.

1 Introduction

This paper presents a completely unsupervised method for inducing morphological knowledge directly from a large monolingual text corpus. This method works by searching for transformation rules that express correspondences between term clusters which are induced from the corpus in an initial step. It covers both inflectional and derivational morphology, and is able to process previously unseen morphs of a word, as long as one of its morphs has been assigned to a cluster.

Aside from its academic appeal, acquisition of this morphological knowledge is a step toward the goal of rapidly retargetable natural language processing. Toward this end, we envisage two uses for it:

1. It can be used to perform morphological normalization (i.e., *stemming* (Porter, 1980)).

2. In the form of *transformation rules*, it can help us classify unknown words, thereby enhancing the utility of cluster-based features for applications such as information extraction (Miller et al., 2004; Freitag, 2004).

There is a considerable literature on the problem of morphology induction in general, and unsupervised (or lightly supervised) induction in particular. Much of the work attempts to exploit orthographic regularities alone, seeking affixation patterns (or *signatures*) that permit a compressive representation of the corpus. Several researchers propose algorithms based on the minimum description length (MDL) principle, achieving reasonable success in discovering regular morphological patterns (Brent et al., 1995; Goldsmith, 2000; Creutz and Lagus, 2002; Argamon et al., 2004). MDL has information theoretic underpinnings, and an information theoretic objective function achieves similar success (Snover et al., 2002). Note that none of these approaches attempts to account for the syntactic dimension of affixation. And all must adopt strategies to cope with a very large search space (the power set of the vocab-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 128–135, Ann Arbor, June 2005.

ulary, in the limit). Such strategies form a common theme in these papers.

Our approach implicitly employs term co-occurrence statistics in the form of statistically derived term clusters. A number of researchers use such statistics directly. A common technique is to cast a word as a distribution over other words that occur within some limited window across the corpus. This definition of co-occurrence yields a semantic distance measure which tends to draw together inflectional variants of a word. Combined with heuristics such as string edit distance, it can be used to find reliable conflation sets (Xu and Croft, 1998; Baroni et al., 2002). A somewhat tighter definition of co-occurrence, which nevertheless yields a semantic distance measure, serves as the basis of a method that captures *irregular* inflectional transformations in Yarowsky and Wicentowski (2001).[1] Schone and Jurafsky (2001) employ distributions over adjacent words (yielding a syntactic distance metric) to improve the precision of their conflation sets.

In contrast with these approaches, ours is predicated on a strictly local notion of co-occurrence. It is well known that clustering terms from a corpus in English or a related language, using a distance measure based on local co-occurrence, yields clusters that correspond roughly to part of speech categories (Schütze, 1995; Clark, 2000). The heart of our idea is to search for affix transformation rules mapping terms in one cluster to those in another. The search for such rules has previously been conducted in the context of supervised part-of-speech tagging (Mikheev, 1997), but not to our knowledge using word clusters. Basing the search for affix patterns on a syntactic partition of the vocabulary, albeit a noisy one, greatly reduces the size of the space of possible conflation sets. Furthermore, the resulting rules can be assigned a syntactic interpretation.

2 Clustering

A prerequisite of our method is a clustering of terms in the corpus vocabulary into rough syntactic groups. To achieve this, we first collect co-occurrence statistics for each word, measuring the frequency of words found immediately adjacent to it in the corpus, treating left occurrences as distinct from right occurrences. This co-occurrence database serves as input to *information theoretic co-clustering* (Dhillon et al., 2003), which seeks a partition of the vocabulary that maximizes the mutual information between term categories and their contexts. This approach to term clustering is closely related to others from the literature (Brown et al., 1992; Clark, 2000).[2]

`recently soon slightly quickly` …
`underwriter designer commissioner` …
`increased posted estimated raised` …
`agreed declined expects wants` …

Table 1: Sample members of four clusters from the Wall Street Journal corpus.

Recall that the *mutual information* between random variables X and Y can be written:

$$\mathrm{M}_{XY} = \sum_{xy} \mathrm{P}(x,y) \ln \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)} \qquad (1)$$

Here, X and Y correspond to term and context clusters, respectively, each event x and y the observation of some term and contextual term in the corpus. We perform an approximate maximization of M_{XY} using a simulated annealing procedure in which each random trial move takes a word x or context y out of the cluster to which it is tentatively assigned and places it into another.

We performed this procedure on the Wall Street Journal (WSJ) portion of the North American News corpus, forming 200 clusters. Table 1 shows sample terms from several hand-selected clusters.

3 Method

In our experiments and the discussion that follows, *stems* are sub-strings of words, to which attach *affixes*, which are sub-string classes denoted by perl-style regular expressions (e.g., `e?d$` or `^re`). A *transform* is an affix substitution which entails a change of clusters. We depict the affix part of the

[1] Note that this method presupposes the availability of several resources in addition to a corpus, including a list of canonical inflectional suffixes.

[2] While we have not experimented with other clustering approaches, we assume that the accuracy of the derived morphological information is *not* very sensitive to the particular methodology.

transform using a perl-style `s///` operator. For example, the transform `s/ed$/ing/` corresponds to the operation of replacing the suffix `ed` with `ing`.

3.1 Overview

The process of moving from term clusters to a transform automaton capable of analyzing novel forms consists of four stages:

1. **Acquire candidate transformations**. By searching for transforms that align a large number of terms in a given pair of clusters, we quickly identify affixation patterns that are likely to have syntactic significance.

2. **Weighting stems and transforms**. The output of Step 1 is a set of transforms, some overlapping, others dubious. This step weights them according to their utility across the vocabulary, using an algorithm similar to Hubs and Authorities (Kleinberg, 1998).

3. **Culling transforms**. We segment the words in the vocabulary, using the transform weights to choose among alternative segmentations. Following this segmentation step, we discard any transform that failed to participate in at least one segmentation.

4. **Constructing an automaton**. From the remaining transforms we construct an automaton, the nodes of which correspond to clusters, the edges to transforms. The resulting data structure can be used to construct morphological parses.

The remainder of this section describes each of these steps in detail.

3.2 Acquiring Transforms

Once we are in possession of a sufficiently large number of term clusters, the acquisition of candidate transforms is conceptually simple. For each pair of clusters, we count the number of times each possible transform is in evidence, then discard those transforms occurring fewer than some small number of times.

For each pair of clusters, we search for suffix or prefix pairs, which, when stripped from matching members in the respective clusters lead to as large a cluster intersection as possible. For example, if `walked` and `talked` are in Cluster 1, and `walking` and `talking` are in Cluster 2, then `walk` and `talk` are in the intersection, given the transform `s/ed$/ing/`. In our experiments, we retain any cluster-to-cluster transform producing an intersection having at least three members.

`s/ful$/less/`	`pain harm use` ...
`s/^/over/`	`charged paid hauled` ...
`s/cked$/wing/`	`kno sho che` ...
`s/nd$/ts/`	`le se fi` ...
`s/s$/ed/`	`recall assert add` ...
`s/ts$/ted/`	`asser insis predic` ...
`s/es$/ing/`	`argu declar acknowledg` ...
`s/s$/ing/`	`recall assert add` ...

Table 2: Sample transforms and matching stems from the Wall Street Journal after the acquisition step.

Table 2 lists some transforms derived from the WSJ as part of this process, along with a few of the stems they match. These were chosen for the sake of illustration; this list does not necessarily the reflect the quality or distribution of the output. (For example, transforms based on the pattern `s/$/s/` easily form the largest block.)

A frequent problem is illustrated by the transforms `s/s$/ed/` and `s/ts$/ted/`. Often, we observe alternative segmentations for the same words and must decide which to prefer. We resolve most of these questions using a simple heuristic. If one transform subsumes another—if the vocabulary terms it covers is a strict superset of those covered by the other transform—then we discard the second one. In the table, all members of the transform `s/ts$/ted/` are also members of `s/s$/ed/`, so we drop `s/ts$/ted/` from the set.

The last two lines of the table represent an obvious opportunity to generalize. In cases like this, where two transforms are from the same cluster pair and involve source or destination affixes that differ in a single letter, the other affixes being equal, we introduce a new transform in which the elided letter is optional (in this example, the transform `s/e?s$/ing/`). The next step seeks to resolve this uncertainty.

s/$/s/	0.2
s/e?$/ed/	0.1
s/e?$/ing/	0.1
s/s$/ses/	1.6e-14
s/w$/ws/	1.6e-14
s/^b/c/	1.6e-14

Table 3: The three highest-weighted and lowest-weighted transforms.

3.3 Weighting Stems and Transforms

The observation that morphologically significant affixes are more likely to be frequent than arbitrary word endings is central to MDL-based systems. Of course, the same can be said about word stems: a string is more likely to be a stem if it is observed with a variety of affixes (or transforms). Moreover, our certainty that it is a valid stem increases with our confidence that the affixes we find attached to it are valid.

This suggests that candidate affixes and stems can "nominate" each other in a way analogous to "hubs" and "authorities" on the Web (Kleinberg, 1998). In this step, we exploit this insight in order to weight the "stem-ness" and "affix-ness" of candidate strings. Our algorithm is closely based on the Hubs and Authorities Algorithm. We say that a stem and transform are "linked" if we have observed a stem to participate in a transform. Beginning with a uniform distribution over stems, we zero the weights associated with transforms, then propagate the stem weights to the transforms. For each stem S and transform T, such that S and T are linked, the weight of S is added to the weight of T. Next, the stem weights are zeroed, and the transform weights propagated to the stems in the same way. This procedure is iterated a few times or until convergence (five times in these experiments).

3.4 Culling Transforms

The output of this procedure is a weighting of candidate stems, on the one hand, and transforms, on the other. Table 3 shows the three highest-weighted and three lowest-weighted transforms from an experiment involving the 10,000 most frequent words in the WSJ.

Although these weights have no obvious linguistic interpretation, we nevertheless can use them to filter further the transform set. In general, however, there is no single threshold that removes all dubious transforms. It does appear to hold, though, that correct transforms (e.g., s/$/s/) outweigh competing incorrect transforms (e.g., s/w$/ws/). This observation motivates our culling procedure: We apply the transforms to the vocabulary in a competitive segmentation procedure, allowing highly weighted transforms to "out-vote" alternative transforms with lower weights. At the completion of this pass through the vocabulary, we retain only those transforms that contribute to at least one successful segmentation.

```
1:  procedure SEGMENT(w)
2:      A[*] ← ∅          ▷ Expansions to transform sets
3:      S[*] ← 0                    ▷ Stems to scores
4:      for each transform t do
5:          if there exists s s.t. w ∈ s · t then
6:              A[s · t] ← A[s · t] ∪ {t}
7:          end if
8:      end for
9:      for T ∈ Range(A) do
10:         B[*] ← 0
11:         for t ∈ T do
12:             s ← w ÷ t
13:             B[s] ← B[s] + Weight(t)
14:         end for
15:         s ← MaxArg_x B[x]
16:         S[s] ← S[s] + B[s]
17:     end for
18:     return MaxArg_x S[x]
19: end procedure
```

Table 4: The segmentation procedure.

Table 4 lists the segmentation procedure. In this pseudocode, w is a word, t a transform, and s a stem. The operation $s \cdot t$ produces the set of (two) words generated by applying the affixes of t to s; the operation $w \div t$ (the *stemming* operation) removes the longest matching affix of t from w. Given a word w, we first find the set of transforms associated with w, grouping them by the pair of words to which they correspond (Lines 4–8). For example, given the word "created", and the transforms s/ed$/ing/, s/ted$/ting/, and s/s$/d/,

the first two transforms will be grouped together in A (with index $\{\text{created}, \text{creating}\}$), while the third will be part of a different group.

Once we have grouped associated transforms, we use them to stem w, accumulating evidence for different stemmings in B. In Line 15, we then discard all but the highest scoring stemming. The score of this stemming is then added to its "global" score in Line 16.

The purpose of this procedure is the suppression of spurious segmentations in Line 15. Although this pseudocode returns only the highest weighted segmentation, it is usually the case that *all* candidate segmentations stored in S are valid, i.e., that several or all breakpoints of a product of multiple affixation are correctly found. And it is a byproduct of this procedure that we require for the final step in our pipeline: In addition to accumulating stemming scores, we record the transforms that contributed to them. We refer to this set of transforms as the *culled* set.

3.5 Constructing an Automaton

Given the culled set of transforms, creation of a parser is straightforward. In the last two steps we have considered a transform to be a pair of affixes (A_S, A_D). Recall that for each such transform there are one or more cluster-specific transforms of the form (C_S, A_S, C_D, A_D) in which the source and destination affixes correspond to clusters. We now convert this set of specific transforms into an automaton in which clusters form the nodes and arcs are affixation operations. For every transform (C_S, A_S, C_D, A_D), we draw an arc from C_S to C_D, labeling it with the general transform (A_S, A_D), and draw the inverse arc from C_D to C_S.

We can now use this automaton for a kind of unsupervised morphological analysis. Given a word, we construct an analysis by finding paths through the automaton to known (or possibly unknown) stem words. Each step replaces one (possibly empty) affix with another one, resulting in a new word form. In general, many such paths are possible. Most of these are redundant, generated by following given affixation arcs to alternative clusters (there are typically several plural noun clusters, for example) or collapsing compound affixations into a single operation.

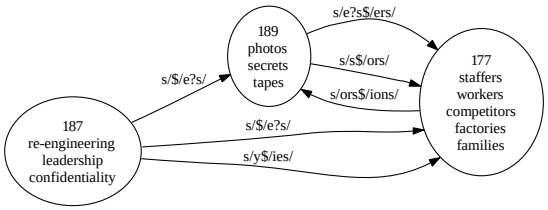

Figure 1: A fragment of the larger automaton from the Wall Street Journal corpus.

In our experiments, we generate all possible paths under the constraint that an operation lead to a known longer wordform, that it be a possible stem of the given word, and that the operation not constitute a loop in the search.[3] We then sort the analysis traces heuristically and return the top one as our analysis. In comparing two traces, we use the following criteria, in order:

- Prefer the trace with the shorter starting stem.
- Prefer the trace involving fewer character edits. (The number of edits is summed across the traces, the trace with the smaller sum preferred.)
- Prefer the trace having more correct cluster assignments of intermediate wordforms.
- Prefer the longer trace.

Note that it is not always clear how to perform an affixation. Consider the transform `s/ing$/e?d/`, for example. In practice, however, this is not a source of difficulty. We attempt both possible expansions (with or without the "e"). If either produces a known wordform which is found in the destination cluster, we discard the other one. If neither resulting wordform can be found in the destination cluster, both are added to the frontier in our search.

4 Evaluation

We evaluate by taking the highest-ranked trace, using the ordering heuristics described in the previous section, as the system's analysis of a given

[3] One wordform w_1 is a possible stem of another w_2, if after stripping any of the affixes in the culled set the resulting string is a sub-string of w_2.

word. This analysis takes the form of a sequence of hypothetical wordforms, from a putative stem to the target wordform (e.g., `decide`, `decision`, `decisions`). The CELEX morphological database (Baayen et al., 1995) is used to produce a reference analysis, by tracing back from the target wordform through any inflectional affixation, then through successive derivational affixations until a stem is reached. Occasionally, this yields more than one analysis. In such cases, all analyses are retained, and the system's analysis is given the most optimistic score. In other words, if a CELEX analysis is found which matches the system's analysis, it is judged to be correct.

4.1 Results

In evaluating an analysis, we distinguish the following outcomes (ordered from most favorable to least):

- **Cor**. The system's analysis matches CELEX's.
- **Over**. The system's analysis contains all the wordforms in CELEX's, also contains additional wordforms, and each of the wordforms is a legitimate morph of the CELEX stem.
- **Under**. The system's analysis contains some of the wordforms in CELEX's; it may contain additional wordforms which are legitimate morphs of the CELEX stem. This happens, for example, when the CELEX stem is unknown to the system.
- **Fail**. The system failed to produce an analysis for a word for which CELEX produced a multiwordform analysis.
- **Spur**. The system produced an analysis for a word which CELEX considered a stem.
- **Incor**. All other (incorrect) cases.

Note that we discard any wordforms which are not in CELEX. Depending on the vocabulary size, anywhere from 15% to 30% are missing. These are often proper nouns.

In addition, we measure precision, recall, and F1 as in Schone and Jurafsky (2001). These metrics reflect the algorithm's ability to group known terms which are morphologically related. Groups are formed by collecting all wordforms that, when analyzed, share a root form. We report these numbers as **Prec**, **Rec**, and **F1**.

	1K	5K	10K	10K+1K	20K
Cor	0.74	0.74	0.75	0.64	0.71
Over	0	0.004	0.003	0.002	0.002
Under	0.005	0.04	0.05	0.06	0.07
Fail	0.25	0.21	0.18	0.28	0.14
Spur	0	0.002	0.01	0.01	0.02
Incor	0	0.003	0.01	0.02	0.05
Prec	1.0	0.98	0.95	1.0	0.80
Rec	0.85	0.82	0.81	0.96	0.82
F1	0.92	0.90	0.87	0.98	0.81

Table 5: Results of experiments using the Wall Street Journal corpus.

We performed the procedure outlined in Section 3.1 using the k most frequent terms from the Wall Street Journal corpus, for k ranging from 1000 to 20,000. The expense of performing these steps is modest compared with that of collecting term co-occurrence statistics and generating term clusters. Our perl implementation of this procedure consumes just over two minutes on a lightly loaded 2.5 GHz Intel machine running Linux, given a collection of 10,000 wordforms in 200 clusters.

The header of each column in Table 5 displays the size of the vocabulary. The column labeled *10K+1K* stands for an experiment designed to assess the ability of the algorithm to process novel terms. For this column, we derived the morphological automaton from the 10,000 most frequent terms, then used it to analyze the next 1000 terms.

The surprising precision/recall scores in this column—scores that are high despite an actual degradation in performance—argues for caution in the use and interpretation of the precision/recall metrics in this context. The difficulty of the morphological conflation set task is a function of the size and constituency of a vocabulary. With a small sample of terms relatively low on the Zipf curve, high precision/recall scores mainly reflect the algorithm's ability to determine that most of the terms are *not* related—a Pyrrhic victory. Nevertheless, these metrics give us a point of comparison with Schone and Jurafsky (2001) who, using a vocabulary of English words occurring at least 10 times in a 6.7 million-word newswire corpus, report F1 of 88.1 for con-

flation sets based only on suffixation, and 84.5 for circumfixation. While a direct comparison would be dubious, the results in Table 5 are comparable to those of Schone and Jurafsky. (Note that we include both prefixation and suffixation in our algorithm and evaluation.)

Not surprisingly, precision and recall degrade as the vocabulary size increases. The top rows of the table, however, suggest that performance is reasonable at small vocabulary sizes and robust across the columns, up to 20K, at which point the system increasingly generates incorrect analyses (more on this below).

4.2 Discussion

A primary advantage of basing the search for affixation patterns on term clusters is that the problem of non-morphological orthographic regularities is greatly mitigated. Nevertheless, as the vocabulary grows, the inadequacy of the simple frequency thresholds we employ becomes clear. In this section, we speculate briefly about how this difficulty might be overcome.

At the 20K size, the system identifies and retains a number of non-morphological regularities. An example are the transforms `s/$/e/` and `s/$/o/`, both of which align members of a name cluster with other members of the same cluster (Clark/Clarke, Brook/Brooke, Robert/Roberto, etc.). As a consequence, the system assigns the analysis `tim => time` to the word "time", suggesting that it be placed in the name cluster.

There are two ways in which we can attempt to suppress such analyses. One is to adjust parameters so that noise transforms are less likely. The procedure for acquiring candidate transforms, described in Section 3.2, discards any that match fewer than 3 stems. When we increase this parameter to 5 and run the 20K experiment again, the incorrect rate falls to 0.02 and F1 rises to 0.84. While this does not solve the larger problem of spurious transforms, it does indicate that a search for a more principled way to screen transforms should enhance performance.

The other way to improve analyses is to corroborate predictions they make about the constituent wordforms. If the `tim => time` analysis is correct, then the word "time" should be at home in the name cluster. This is something we can check. Recall that in our framework both terms and clusters are associated with distributions over adjacent terms (or clusters). We can hope to improve precision by discarding analyses that assign a term to a cluster from which it is too distributionally distant. Applying such a filter in the 20K experiment, has a similar impact on performance as the transform filter of the previous paragraph, with F1 rising to 0.84.[4]

Several researchers have established the utility of a filter in which the broader context distributions surrounding two terms are compared, in an effort to insure that they are semantically compatible (Schone and Jurafsky, 2001; Yarowsky and Wicentowski, 2001). This would constitute a straightforward extension of our framework.

Note that the system is often able to produce the correct analysis, but ordering heuristics described in Section 3.5 cause it to be discarded in favor of an incorrect one. The analyses `us => using` and `use => using` are an example, the former being the one favored for the word "using". Note, though, that our automaton construction procedure discards a potentially useful piece of information—the amount of support each arc receives from the data (the number of stems it matches). This might be converted into something like a traversal probability and used in ordering analyses.

Of course, a further shortcoming of our approach is its inability to account for irregular forms. It shares this limitation with all other approaches based on orthographic similarity (a notable exception is Yarowsky and Wicentowski (2001)). However, there is reason to believe that it could be extended to accommodate at least some irregular forms. We note, for example, the cluster pair 180/185, which is dominated by the transform `s/e?$/ed/`. Cluster 180 contains words like "make", "pay", and "keep", while Cluster 185 contains "made", "paid", and "kept". In other words, once a strong correspondence is found between two clusters, we can search for an alignment which covers the orphans in the respective clusters.

[4] Specifically, we take the Hellinger distance between the two distributions, scaled into the range $[0, 1]$, and discard those analyses for which the term is at a distance greater than 0.5 from the proposed cluster.

5 Conclusion

We have shown that automatically computed term clusters can form the basis of an effective unsupervised morphology induction system. Such clusters tend to group terms by part of speech, greatly simplifying the search for syntactically significant affixes. Furthermore, the learned affixation patterns are not just orthographic features or morphological conflation sets, but cluster-to-cluster transformation rules. We exploit this in the construction of morphological automata able to analyze previously unseen wordforms.

We have not exhausted the sources of evidence implicit in this framework, and we expect that attending to features such as transform frequency will lead to further improvements. Our approach may also benefit from the kinds of broad-context semantic filters proposed elsewhere. Finally, we hope to use the cluster assignments suggested by the morphological rules in refining the original cluster assignments, particularly of low-frequency words.

Acknowledgments

This material is based on work funded in whole or in part by the U.S. Government. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the U.S. Government.

References

S. Argamon, N. Akiva, A. Amir, and O. Kapah. 2004. Efficient unsupervised recursive word segmentation using minimum description length. In *Proc. 20th International Conference on Computational Linguistics (Coling-04)*.

R.H. Baayen, R. Piepenbrock, and L. Gulikers. 1995. *The CELEX Lexical Database (CD-ROM)*. LDC, University of Pennsylvania, Philadelphia.

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proc. ACL-02 Workshop on Morphological and Phonological Learning*.

M. Brent, S.K. Murthy, and A. Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proc. 5th International Workshop on Artificial Intelligence and Statistics*.

P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL 2000*, September.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Morphological and Phonological Learning: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*.

I.S. Dhillon, S. Mallela, and D.S. Modha. 2003. Information-theoretic co-clustering. Technical Report TR-03-12, Dept. of Computer Science, U. Texas at Austin.

D. Freitag. 2004. Trained named entity recognition using distributional clusters. In *Proceedings of EMNLP-04*.

J. Goldsmith. 2000. Unsupervised learning of the morphology of a natural language. http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/Paper/paper.html.

J.M. Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*.

A. Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT/NAACL 04*.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proc. NAACL-01*.

H. Schütze. 1995. Distributional part-of-speech tagging. In *Proc. 7th EACL Conference (EACL-95)*, March.

M.G. Snover, G.E. Jarosz, and M.R. Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *Morphological and Phonological Learning: Proc. 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*.

J. Xu and W.B. Croft. 1998. Corpus-based stemming using co-occurrence of word variants. *ACM TOIS*, 18(1).

D. Yarowsky and R. Wicentowski. 2001. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-01*.