Explaining Unintelligible Words by Means of their Context

Balázs Pintér, Gyula Vörös, Zoltán Szabó and András Lőrincz
Faculty of Informatics, Eötvös Loránd University, Pázmány P. sétány 1/C, H-1117 Budapest, Hungary

Keywords: Unintelligible Words, Wikification, Link Disambiguation, Natural Language Processing, Structured Sparse Coding.

Abstract: Explaining unintelligible words is a practical problem for text obtained by optical character recognition, from the Web (e.g., because of misspellings), etc. Approaches to wikification, to enriching text by linking words to Wikipedia articles, could help solve this problem. However, existing methods for wikification assume that the text is correct, so they are not capable of wikifying erroneous text. Because of errors, the problem of disambiguation (identifying the appropriate article to link to) becomes large-scale: as the word to be disambiguated is unknown, the article to link to has to be selected from among hundreds, maybe thousands of candidate articles. Existing approaches for the case where the word is known build upon the distributional hypothesis: words that occur in the same contexts tend to have similar meanings. The increased number of candidate articles makes the difficulty of spuriously similar contexts (when two contexts are similar but belong to different articles) more severe. We propose a method to overcome this difficulty by combining the distributional hypothesis with structured sparsity, a rapidly expanding area of research. Empirically, our approach based on structured sparsity compares favorably to various traditional classification methods.

1 INTRODUCTION

Many common types of errors can occur in free text that produce *unintelligible words*. A word may be misspelled. Errors can be introduced by Optical Character Recognition (OCR) because of imperfect scans or errors committed by the algorithm. Automatic speech recognition can also introduce errors. Explaining these unintelligible words with Wikipedia articles could help users and computer algorithms alike to understand them.

Enriching text documents with links to Wikipedia articles, wikification, has been in the focus of much attention recently. Starting from the work of (Mihalcea and Csomai, 2007) wikification consists of two phases: *link detection* and *link disambiguation*. The detection phase identifies the terms and phrases from which links should be made. The disambiguation phase identifies the appropriate article for each detected term to link to. For example, the term *bank* could link to an article about financial institutions, or river banks.

We consider link disambiguation as our starting point to approach explaining unintelligible words. The words to be disambiguated are assumed given: they are the erroneous words in the text. They can be selected by the user, or detected automatically by methods such as (Kukich, 1992; Leacock et al., 2010).

Current approaches to link disambiguation do not handle text with errors because of a tacit assumption: the *disambiguation of different word types*[1] *are treated as independent problems*. This constraint is essential to reduce the complexity of the disambiguation problem: without it, the article the target word[2] will link to has to be selected from among hundreds, maybe thousands of candidate articles. If an error makes the surface form[3] of the target word unusable, the problem cannot be decomposed to disambiguation on different word types: the vast number of candidate articles yields a large-scale problem. Due to the large scale, an additional difficulty appears.

Typical methods to disambiguate *known* target words apply the *distributional hypothesis*. According to the distributional hypothesis, words that occur in the same contexts tend to have similar meanings (Harris, 1954). Because the disambiguation problem with unknown target words is intrinsically large-scale, exceptions to the distributional hypothesis can occur

[1] In "A rose is a rose is a rose", there are three word types (a, rose, is), but eight word tokens.

[2] The word to be explained with a Wikipedia article.

[3] The form of a word as it appears in the text.

Pintér B., Vörös G., Szabó Z. and Lőrincz A. (2013).
Explaining Unintelligible Words by Means of their Context.
In *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods*, pages 382-387
DOI: 10.5220/0004267003820387

more frequently. Particularly, let us call two contexts *spuriously similar* if they are similar but belong to different articles. The amount of spuriously similar contexts tends to increase inherently with the number of candidate articles. This makes the learning problem considerably hard.

In this paper, we propose a method to explain unintelligible words with Wikipedia articles that addresses this problem by using the *distributional hypothesis* in a novel way. Structured sparse coding (Bach et al., 2012) is introduced to diminish the effect of spurious similarities of contexts by utilizing the structure of semantic space (Section 3).

The **contributions** of the paper are summarized as follows: (i) we propose a method to disambiguate unintelligible words to Wikipedia articles. (ii) We show that structured sparsity reduces the effect of spurious similarities of contexts. (iii) We perform large-scale evaluations where we disambiguate from among 1000 Wikipedia articles at once.

In the next section we review related work. Our method and results are described in Section 3 and 4. We discuss our results in Section 5 and conclude in Section 6.

2 RELATED WORK

The main difference between previous methods in the literature and ours is that they consider the disambiguation problems of different word types independently. In our case, as the word type is *unintelligible*, this would be unfeasible.

(Mihalcea and Csomai, 2007) introduced the concept of wikification: they proposed a method to automatically enrich text with links to Wikipedia articles. They used keyword extraction to detect the most important terms in the text, and disambiguated them to Wikipedia articles with supervised learning using the contexts. The same task was solved in (Milne and Witten, 2008) more efficiently. Here, contexts were taken into account also for the detection phase. Disambiguation was done using sense *commonness* and sense *relatedness* scores.

Unlike the previously mentioned works, which introduce links to important terms in the text to achieve better readability, the goal of (Kulkarni et al., 2009) was to add as many links as possible to help indexing for information retrieval. The terms were disambiguated by assuming that coherent documents refer to entities from one or a few related topics or domains. (Ratinov et al., 2011) proposed a similar disambiguation system called GLOW (global wikification), which used several local and global features to obtain a set of disambiguations that are coherent in the whole text.

In information retrieval and speech recognition, unintelligible words pose a practical problem. The TREC-5 confusion track (Kantor and Voorhees, 2000) studied the impact of data corruption introduced by scanning or OCR errors on retrieval performance. In the subsequent spoken document retrieval tracks (Garofolo et al., 2000), the errors were introduced by automatic speech recognition.

Structured sparsity has been successfully applied to natural language processing problems different from ours in works such as (Jenatton et al., 2011) and (Martins et al., 2011). (Jenatton et al., 2011) apply sparse hierarchical dictionary learning to learn hierarchies of topics from a corpora of NIPS proceedings papers. In a more recent application (Martins et al., 2011), structured sparsity was used to perform effective feature template selection on three natural language processing tasks (chunking, entity recognition, and dependency parsing).

3 THE METHOD

We start from a list of candidate articles the unintelligible word could be linked to. For each *candidate article*, we collect a number of *contexts*. A context of a candidate article consists of the N non-stopword words occurring before and after the anchor of the link that points to the article. There can be at most $2N$ words in a context.

The presented method makes use of a collection of such *contexts* arranged in a word-context matrix $\mathbf{D}$ (Turney and Pantel, 2010) (Figure 1). In this matrix, each context is a column represented as a bag-of-words vector $\mathbf{v}$ of word frequencies, where v_i is the number of occurrences of the ith word in the context.

The article is determined in two steps. First, we formulate an inverse problem, and compute a representation vector $\boldsymbol{\alpha}$. In the *second step*, a single candidate article is selected based on the weights in this vector.

To compute the representation vector $\boldsymbol{\alpha}$, the context $\mathbf{x} \in \mathbb{R}^m$ of the target word is approximated linearly with the columns of the word-context matrix $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n] \in \mathbb{R}^{m \times n}$, called the dictionary in the terminology of sparse coding. The columns of the dictionary contain contexts, each labeled with the candidate article $l_i \in L$ the context was collected for. Please note that multiple contexts can be, and in many cases are, tagged with the same candidate article: $l_i = l_j$ is possible. There are m words in the vocabulary, and n contexts in the dictionary.

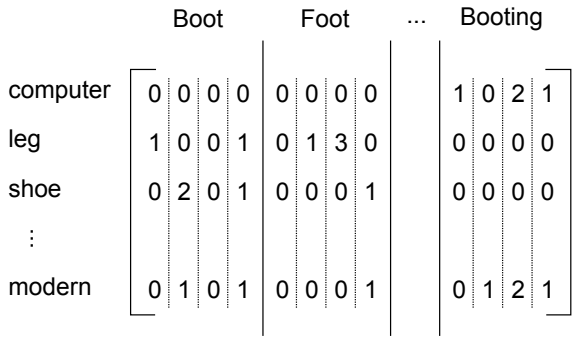


Figure 1: The word-context matrix $\mathbf{D}$. Each column is a context of a candidate article (e.g., *Boot*, *Foot*). Each element D_{ij} of the matrix holds the number of occurences of the ith word in the jth context. For example, the word *leg* occurs three times in the 7th context, which is the 3rd context labeled with *Foot*.

The representation vector $\boldsymbol{\alpha}$ consists of the coefficients of a linear combination

$$\mathbf{x} = \alpha_1 \mathbf{d}_1 + \alpha_2 \mathbf{d}_2 + \ldots + \alpha_n \mathbf{d}_n. \quad (1)$$

For each target word, whose context is $\mathbf{x} \in \mathbb{R}^m$, a representation vector $\boldsymbol{\alpha} = [\alpha_1; \alpha_2; \ldots; \alpha_n] \in \mathbb{R}^n$ is computed.

To diminish the effect of spurious similarities, we introduce a structured sparsity inducing regularization by organizing the contexts in $\mathbf{D}$ into *groups*. Each group contains the contexts annotated with a single candidate article. As only a single candidate is selected for each target word, ideally only a single group should be active in each representation vector $\boldsymbol{\alpha}$. Sparsity on the groups is realized by computing $\boldsymbol{\alpha}$ with a *group Lasso* regularization (Yuan et al., 2006) determined by the labels.

The groups are introduced as a family of sets $\mathcal{G} = \{G_l\}_{l \in L} \subseteq 2^{\{1,\ldots,n\}}$. There are as many sets in $\mathcal{G}$ as there are distinct candidate articles in L. For each article $l \in L$, there is exactly one set $G_l \in \mathcal{G}$ that contains the indices of all the columns $\mathbf{d}_i$ tagged with l. $\mathcal{G}$ forms a partition.

The representation vector $\boldsymbol{\alpha}$ of the target word whose context is $\mathbf{x}$ is defined as the minimum of the loss function

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{l \in L} w_l \|\boldsymbol{\alpha}_{G_l}\|_2, \quad (2)$$

where $\boldsymbol{\alpha}_{G_l} \in \mathbb{R}^{|G_l|}$ denotes the vector where only the coordinates present in the set $G_l \subseteq \{1, \ldots, n\}$ are retained.

The first term is the approximation error, the second one realizes the structured sparsity inducing regularization. Parameter $\lambda > 0$ controls the tradeoff between the two terms. The parameters $w_l > 0$ denote the weights for each group G_l.

If each group is a singleton (i.e., $\mathcal{G} = \{\{1\}, \{2\}, \ldots, \{n\}\}$) the Lasso problem (Tibshirani, 1994) is recovered:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_{i=1}^{n} w_i |\alpha_i|. \quad (3)$$

Setting $\lambda = 0$ yields the least squares cost function.

For the sake of simplicity, we represent each candidate article with the same number of contexts: there are an equal number of columns in $\mathbf{D}$ for each label $l \in L$ ($|G_1| = |G_2| = \cdots = |G_{|L|}|$). The weights w_l of the groups are set to 1.

In the *second step*, the link is disambiguated to a single article based on the weights in this vector. We utilize the group structure to condense the vector $\boldsymbol{\alpha}$ to a single article. The weights are summed in each group $G_l \in \mathcal{G}$, and the article $l^* \in L$ whose group contains the most weight is selected:

$$l^* = \arg\max_{l \in L} \sum_i (\boldsymbol{\alpha}_{G_l})_i.$$

In this group Lasso formulation, whole *groups* are selected. Each group $G_l \in \mathcal{G}$ contains contexts tagged with the same candidate article $l \in L$, and only a few groups can be selected. A context similar to the context of the target word, $\mathbf{x}$, only by accident has a smaller chance to be selected: as it is in a group labeled with a candidate article that is less related in meaning to the target word than the correct candidate, its group contains mainly contexts less similar to $\mathbf{x}$. Therefore, errors introduced by spurious similarities of contexts can be effectively diminished (Section 4).

4 RESULTS

To evaluate the method, we solve the disambiguation task of wikification, with a significant difference: we assume that the surface form of the target word is unknown.

4.1 The Datasets

The datasets used in our experiments are obtained by randomly sampling the links in Wikipedia. Each dataset consists of contexts labeled with candidate articles $(\mathbf{c}_1, l_1), (\mathbf{c}_2, l_2), \ldots$. Each labeled context is obtained by processing a link: the bag-of-words vector generated from the context of the anchor text is annotated with the target of the link.

We use the English Wikipedia database dump from October 2010[4]. Disambiguation pages, and articles that are too small to be relevant (i.e., have less than 200 non-stopwords in their texts, or less than 20 incoming and 20 outgoing links) are discarded. Inflected words are reduced to root forms by the Porter stemming algorithm (Porter, 1997).

To produce a dataset, a list of anchor texts are generated that match a number of criteria. These criteria have been chosen to obtain (i) words that are frequent enough to be suitable training examples and (ii) are proper English words. The anchor text has to be a single word between 3 and 20 characters long, must consist of the letters of the English alphabet, must be present in Wikipedia at least 100 times, and must point to at least two different Wikipedia articles, but not to more than 20. It has to occur at least once in *WordNet* (Miller, 1995) and at least three times in the *British National Corpus* (BNC Consortium, 2001).

A number of anchor texts are selected from this list randomly, and their linked occurrences are collected along with their N-wide contexts. Each link is processed to obtain a labeled context $(\mathbf{c}_i, l_i)$.

To ensure that there are an equal number of contexts tagged with each article $l \in L$, d randomly selected contexts are collected for each label. Labels with less than d contexts are discarded. We do not perform feature selection, but we remove the words that appear less than five times across all contexts, in order to discard very rare words.

4.2 Evaluations

The task we proposed is a disambiguation problem where the algorithm has to decide between all candidate articles at once, because the surface form of the target word is not available. Given a context $\mathbf{x} \in \mathbb{R}^m$ of a word, the goal is to determine the appropriate candidate article $l \in L$. The performance of the algorithms is measured as the accuracy of this classification.

We compare our group Lasso based method to three baselines: two different regularizations (least squares and the Lasso) of the inverse problem described in Section 3, and a Support Vector Machine (SVM). The SVM is a multiclass Support Vector Machine with a linear kernel, used successfully for wikification in previous works (Milne and Witten, 2008; Ratinov et al., 2011).

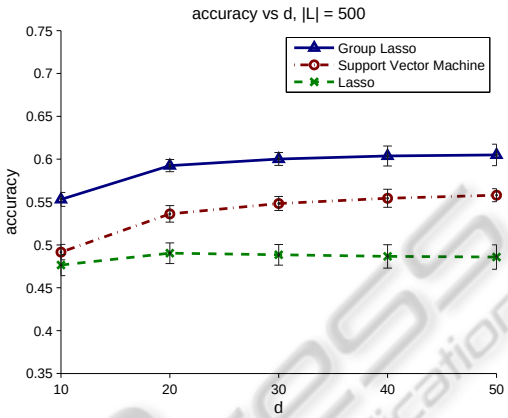


Figure 2: Dependency of the accuracy on the number of contexts per candidate article. There are $d-1$ such contexts in each step of the cross-validation, as there is one test example for each article. The data points are the mean of values obtained on the five datasets. The error bars denote the standard deviations. The results of least squares are not illustrated as the standard deviations were very large. It performs consistently below the Lasso.

For least squares and the Lasso, the link is disambiguated to the article that corresponds to the largest coefficient in $\boldsymbol{\alpha}$. For the SVM, a classification problem is solved using the labeled contexts $(\mathbf{c}_i, l_i)$ as training and test examples.

The minimization problems of both the *Lasso* and the *group Lasso* (Eq. 2) are solved by the Sparse Learning with Efficient Projections (SLEP) package (Liu et al., 2009). For the *support vector machine*, we use the implementation of LIBSVM (Chang and Lin, 2001).

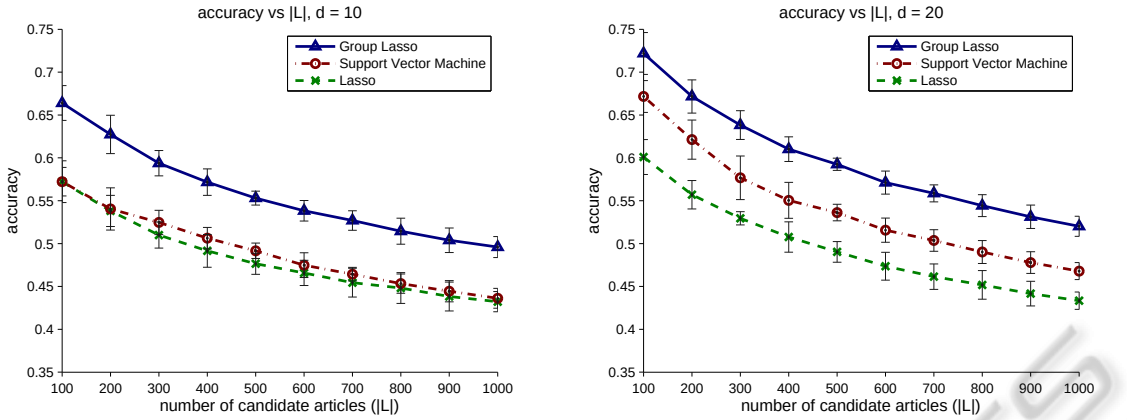
The algorithms are evaluated on five disjoint datasets generated from Wikipedia (Section 4.1), each with different candidate articles. The mean and standard deviation of the accuracy across these five datasets are reported.

There are $|L| = 1000$ different articles in each dataset, and $d = 50$ contexts tagged with each article. The algorithms are evaluated on datasets of different sizes (i.e., d and $|L|$ are different), generated from the original five datasets by removing contexts and their labels randomly .

In accord with (Lee and Ng, 2002; Schütze, 1998), and others, we use a broad context, $N = 20$. We found that a broad context improves the performance of all four algorithms.

Before evaluating the algorithms, we examined the effect of their parameters on the results. We

[4]Downloaded from http://dumps.wikimedia.org/enwiki/.



(a) (b)

Figure 3: Dependency of the accuracy on the number of candidate articles, $|L|$. The data points are the mean of values obtained on the five datasets. The error bars denote the standard deviations. The results of least squares are not illustrated, as the standard deviations were very large. It performs consistently below the Lasso.

found that the algorithms are robust: for the Lasso, $\lambda = 0.005$, for the group Lasso, $\lambda = 0.05$, and for the SVM, $C = 1$ was optimal in almost every validation experiment.

In the first evaluation, we examine the effect of the number of training examples per candidate article on the accuracy of the four algorithms. The starting datasets consist of $|L| = 500$ articles with $d = 10$ contexts (or examples) each. Stratified 10-fold cross-validation is used to determine the accuracy of the classification. The dataset is partitioned into 10 subsets (the same as d), where each subset contains exactly $|L|$ examples – one annotated with each article. In one iteration, one subset is used for testing, and the other 9 subsets form the columns of $\mathbf{D}$: there are $|L|$ test examples and $n = (d-1)|L|$ columns in $\mathbf{D}$ in each iteration. For the SVM, the columns of $\mathbf{D}$ are used as training examples.

To examine the effect of additional contexts, we add contexts to $\mathbf{D}$ for each candidate article, and examine the change in accuracy. In order to evaluate the effect correctly (i.e., to not make the learning problem harder), the test examples remain the same as with $d = 10$. In other words, we perform the same cross-validation as before, only we add additional columns to $\mathbf{D}$ in each step. In Figure 2, we report the results for $d = 10, 20, 30, 40, 50$.

In the second evaluation, the accuracy of the algorithms is examined as the number of candidate articles $|L|$ increases. As in the first evaluation, there are $d = 10$ examples per candidate article, and stratified 10-fold cross-validation is performed. Then, the number of examples is raised to $d = 20$ in the same way (i.e., the new examples are not added to the test examples). We report the results for $|L| = 100, 200, \ldots, 1000$ candidate articles in Figure 3.

5 DISCUSSION

The results are very consistent across the five disjoint datasets, except in the case when the representation vector was computed with least squares. The performance of least squares was the worst of the four algorithms, and it was so erratic that we did not plot it in order to keep the figure uncluttered.

For group Lasso and the SVM, additional training examples help up to 20 examples per article (Figure 2), but only small gains can be achieved by adding more than 20 examples.

In sharp contrast, the Lasso-based representation does not benefit from new training examples *at all* when there are many candidate articles. This may be the effect of spurious similarities. As more and more candidate articles are added, the less chance Lasso has to select the right article from among the candidates.

Representation vectors computed with structured sparsity inducing regularization significantly outperform the other methods, including SVM (Figure 3). This illustrates the efficiency of our method: structured sparsity decreases the chance of selecting contexts spuriously similar to the context of the target word.

6 CONCLUSIONS

We proposed a method to explain unintelligible words with Wikipedia articles. In addition to explaining un-

intelligible words, the method can also be used to help wikify possibly erroneous text from real-world sources such as the Web, optical character recognition, or speech recognition. The numerical evaluations demonstrated that the method works consistently even in large-scale experiments, when disambiguating between up to 1000 Wikipedia articles.

A possible future application of the presented method is the verification of links to Wikipedia. The method can assign a single weight to each candidate article: the sum of the weights in its group in the representation vector $\boldsymbol{\alpha}$. If the weight corresponding to the target of the link is small in contrast to weights of other articles, the link is probably incorrect.

The presented method can be generalized, as it can work with arbitrarily labeled text fragments as well as contexts of Wikipedia links. This more general framework may have further applications, as the idea of distributional similarity offers solutions to many natural language processing problems. For example, topics might be assigned to documents as in centroid-based document classification (Han and Karypis, 2000).

ACKNOWLEDGEMENTS

The research has been supported by the 'European Robotic Surgery' EC FP7 grant (no.: 288233). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of other members of the consortium or the European Commission.

REFERENCES

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106.

BNC Consortium (2001). The British National Corpus, version 2 (BNC World).

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*.

Garofolo, J. S., Auzanne, C. G. P., and Voorhees, E. M. (2000). The TREC Spoken Document Retrieval Track: A Success Story. In *RIAO*, pages 1–20.

Han, E.-H. and Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In *PKDD*, pages 116–123.

Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2011). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

Kantor, P. B. and Voorhees, E. M. (2000). The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text. *Information Retrieval*, 2:165–176.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Kulkarni, S., Singh, A., Ramakrishnan, G., and Chakrabarti, S. (2009). Collective annotation of Wikipedia entities in web text. In *KDD*, pages 457–466.

Leacock, C., Chodorow, M., Gamon, M., and Tetreault, J. (2010). *Automated Grammatical Error Detection for Language Learners*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Lee, Y. K. and Ng, H. T. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *EMNLP*, pages 41–48.

Liu, J., Ji, S., and Ye, J. (2009). *SLEP: Sparse Learning with Efficient Projections*. Arizona State University.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011). Structured Sparsity in Structured Prediction. In *EMNLP*, pages 1500–1511.

Mihalcea, R. and Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38:39–41.

Milne, D. and Witten, I. H. (2008). Learning to link with Wikipedia. In *CIKM*, pages 509–518.

Porter, M. F. (1997). *An algorithm for suffix stripping*, pages 313–316. Morgan Kaufmann Publishers Inc.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to Wikipedia. In *ACL-HLT*, pages 1375–1384.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Yuan, M., Yuan, M., Lin, Y., and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67.