HITS-based Seed Selection and Stop List Construction for Bootstrapping

Tetsuo Kiso **Masashi Shimbo** **Mamoru Komachi** **Yuji Matsumoto**
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
{tetsuo-s,shimbo,komachi,matsu}@is.naist.jp

Abstract

In bootstrapping (seed set expansion), selecting good seeds and creating stop lists are two effective ways to reduce semantic drift, but these methods generally need human supervision. In this paper, we propose a graph-based approach to helping editors choose effective seeds and stop list instances, applicable to Pantel and Pennacchiotti's *Espresso* bootstrapping algorithm. The idea is to select seeds and create a stop list using the rankings of instances and patterns computed by Kleinberg's HITS algorithm. Experimental results on a variation of the lexical sample task show the effectiveness of our method.

1 Introduction

Bootstrapping (Yarowsky, 1995; Abney, 2004) is a technique frequently used in natural language processing to expand limited resources with minimal supervision. Given a small amount of sample data (*seeds*) representing a particular semantic class of interest, bootstrapping first trains a classifier (which often is a weighted list of surface patterns characterizing the seeds) using the seeds, and then apply it on the remaining data to select instances most likely to be of the same class as the seeds. These selected instances are added to the seed set, and the process is iterated until sufficient labeled data are acquired.

Many bootstrapping algorithms have been proposed for a variety of tasks: word sense disambiguation (Yarowsky, 1995; Abney, 2004), information extraction (Hearst, 1992; Riloff and Jones, 1999; Thelen and Riloff, 2002; Pantel and Pennacchiotti, 2006), named entity recognition (Collins and Singer, 1999), part-of-speech tagging (Clark et al., 2003), and statistical parsing (Steedman et al., 2003; McClosky et al., 2006).

Bootstrapping algorithms, however, are known to suffer from the problem called *semantic drift*: as the iteration proceeds, the algorithms tend to select instances increasingly irrelevant to the seed instances (Curran et al., 2007). For example, suppose we want to collect the names of common tourist sites from a web corpus. Given seed instances {*New York City*, *Maldives Islands*}, bootstrapping might learn, at one point of the iteration, patterns like *"pictures of X"* and *"photos of X,"* which also co-occur with many irrelevant instances. In this case, a later iteration would likely acquire frequent words co-occurring with these *generic* patterns, such as *Michael Jackson*.

Previous work has tried to reduce the effect of semantic drift by making the *stop list* of instances that must not be extracted (Curran et al., 2007; McIntosh and Curran, 2009). Drift can also be reduced with carefully selected seeds. However, both of these approaches require expert knowledge.

In this paper, we propose a graph-based approach to seed selection and stop list creation for the state-of-the-art bootstrapping algorithm Espresso (Pantel and Pennacchiotti, 2006). An advantage of this approach is that it requires zero or minimal supervision. The idea is to use the *hubness* score of instances and patterns computed from the pointwise mutual information matrix with the HITS algorithm (Kleinberg, 1999). Komachi et al. (2008) pointed out that semantic drift in Espresso has the same root as *topic drift* (Bharat and Henzinger, 1998) observed with HITS, noting the algorithmic similarity between them. While Komachi et al. proposed to use algorithms different from Espresso to

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers, pages 30–36,
Portland, Oregon, June 19-24, 2011.

avoid semantic drift, in this paper we take advantage of this similarity to make better use of Espresso.

We demonstrate the effectiveness of our approach on a word sense disambiguation task.

2 Background

In this section, we review related work on seed selection and stop list construction. We also briefly introduce the Espresso bootstrapping algorithm (Pantel and Pennacchiotti, 2006) for which we build our seed selection and stop list construction methods.

2.1 Seed Selection

The performance of bootstrapping can be greatly influenced by a number of factors such as the size of the seed set, the composition of the seed set and the coherence of the concept being expanded (Vyas et al., 2009). Vyas et al. (2009) studied the impact of the composition of the seed sets on the expansion performance, confirming that seed set composition has a significant impact on the quality of expansions. They also found that the seeds chosen by non-expert editors are often worse than randomly chosen ones. A similar observation was made by McIntosh and Curran (2009), who reported that randomly chosen seeds from the gold-standard set often outperformed seeds chosen by domain experts. These results suggest that even for humans, selecting good seeds is a non-trivial task.

2.2 Stop Lists

Yangarber et al. (2002) proposed to run multiple bootstrapping sessions in parallel, with each session trying to extract one of several mutually exclusive semantic classes. Thus, the instances harvested in one bootstrapping session can be used as the stop list of the other sessions. Curran et al. (2007) pursued a similar idea in their *Mutual Exclusion Bootstrapping*, which uses multiple semantic classes in addition to hand-crafted stop lists. While multi-class bootstrapping is a clever way to reduce human supervision in stop list construction, it is not generally applicable to bootstrapping for a single class. To apply the idea of multi-class bootstrapping to single-class bootstrapping, one has to first find appropriate competing semantic classes and good seeds for them, which is in itself a difficult problem. Along this line of research, McIntosh (2010) recently used clustering to find competing semantic classes (negative categories).

Algorithm 1 Espresso algorithm

Input: Seed vector $\mathbf{i}_0$
  Instance-pattern co-occurrence matrix $\mathbf{A}$
  Instance cutoff parameter k
  Pattern cutoff parameter m
  Number of iterations τ
Output: Instance score vector $\mathbf{i}$
  Pattern score vector $\mathbf{p}$
function ESPRESSO($\mathbf{i}_0, \mathbf{A}, k, m, \tau$)
  $\mathbf{i} \leftarrow \mathbf{i}_0$
  for $t = 1, 2, ..., \tau$ **do**
    $\mathbf{p} \leftarrow \mathbf{A}^{\mathrm{T}}\mathbf{i}$
    Scale $\mathbf{p}$ so that the components sum to one.
    $\mathbf{p} \leftarrow$ SELECTKBEST($\mathbf{p}, k$)
    $\mathbf{i} \leftarrow \mathbf{A}\mathbf{p}$
    Scale $\mathbf{i}$ so that the components sum to one.
    $\mathbf{i} \leftarrow$ SELECTKBEST($\mathbf{i}, m$)
  return $\mathbf{i}$ and $\mathbf{p}$
function SELECTKBEST($\mathbf{v}, k$)
  Retain only the k largest components of $\mathbf{v}$, resetting the remaining components to 0.
  return $\mathbf{v}$

2.3 Espresso

Espresso (Pantel and Pennacchiotti, 2006) is one of the state-of-the-art bootstrapping algorithms used in many natural language tasks (Komachi and Suzuki, 2008; Abe et al., 2008; Ittoo and Bouma, 2010; Yoshida et al., 2010). Espresso takes advantage of pointwise mutual information (pmi) (Manning and Schütze, 1999) between instances and patterns to evaluate their reliability. Let n be the number of all instances in the corpus, and p the number of all possible patterns. We denote all pmi values as an $n \times p$ instance-pattern matrix $\mathbf{A}$, with the (i, j) element of $\mathbf{A}$ holding the value of pmi between the ith instance and the jth pattern. Let $\mathbf{A}^{\mathrm{T}}$ denote the matrix transpose of $\mathbf{A}$.

Algorithm 1 shows the pseudocode of Espresso. The input vector $\mathbf{i}_0$ (called *seed vector*) is an n-dimensional binary vector with 1 at the ith component for every seed instance i, and 0 elsewhere. The algorithm outputs an n-dimensional vector $\mathbf{i}$ and an p-dimensional vector $\mathbf{p}$, respectively representing the final scores of instances and patterns. Note that for brevity, the pseudocode assumes fixed numbers (k and m) of components in $\mathbf{i}$ and $\mathbf{p}$ are carried over to the subsequent iteration, but the original Espresso

allows them to gradually increase with the number of iterations.

3 HITS-based Approach to Seed Selection and Stop List Construction

3.1 Espresso and HITS

Komachi et al. (2008) pointed out the similarity between Espresso and Kleinberg's HITS web page ranking algorithm (Kleinberg, 1999). Indeed, if we remove the pattern/instance selection steps of Algorithm 1 (lines 13 and 16), the algorithm essentially reduces to HITS. In this case, the outputs **i** and **p** match respectively the hubness and authority score vectors of HITS, computed on the bipartite graph of instances and patterns induced by matrix **A**.

An implication of this algorithmic similarity is that the outputs of Espresso are inherently biased towards the HITS vectors, which is likely to be the cause of semantic drift. Even though the pattern/instance selection steps in Espresso reduce such a bias to some extent, the bias still persists, as empirically verified by Komachi et al. (2008). In other words, the expansion process does not drift in random directions, but tend towards the set of instances and patterns with the highest HITS scores, regardless of the target semantic class. We exploit this observation in seed selection and stop list construction for Espresso, in order to reduce semantic drift.

3.2 The Procedure

Our strategy is extremely simple, and can be summarized as follows.

1. First, compute the HITS ranking of instances in the graph induced by the pmi matrix **A**. This can be done by calling Algorithm 1 with $k = m = \infty$ and a sufficiently large τ.

2. Next, check the top instances in the HITS ranking list manually, and see if these belong to the target class.

3. The third step depends on the outcome of the second step.

   (a) If the top instances are of the target class, use them as the seeds. We do not use a stop list in this case.

   (b) If not, these instances are likely to make a vector for which semantic drift is directed; hence, use them as the stop list. In this case, the seed set must be prepared manually, just like the usual bootstrapping procedure.

4. Run Espresso with the seeds or stop list found in the last step.

4 Experimental Setup

We evaluate our methods on a variant of the *lexical sample* word sense disambiguation task. In the lexical sample task, a small pre-selected set of a target word is given, along with an inventory of senses for each word (Jurafsky and Martin, 2008). Each word comes with a number of instances (context sentences) in which the target word occur, and some of these sentences are manually labeled with the correct sense of the target word in each context. The goal of the task is to classify unlabeled context sentences by the sense of the target word in each context, using the set of labeled sentences.

To apply Espresso for this task, we reformulate the task to be that of seed set expansion, and not classification. That is, the hand-labeled sentences having the same sense label are used as the seed set, and it is expanded over all the remaining (unlabeled) sentences.

The reason we use the lexical sample task is that every sentence (instance) belongs to one of the predefined senses (classes), and we can expect the most frequent sense in the corpus to form the highest HITS ranking instances. This allows us to completely automate our experiments, without the need to manually check the HITS ranking in Step 2 of Section 3.2. That is, for the most frequent sense (majority sense), we take Step 3a and use the highest ranked instances as seeds; for the rest of the senses (minority senses), we take Step 3b and use them as the stop list.

4.1 Datasets

We used the seven most frequent polysemous nouns (*arm, bank, degree, difference, paper, party* and *shelter*) in the SENSEVAL-3 dataset, and *line* (Leacock et al., 1993) and *interest* (Bruce and Wiebe,

Task	Method	MAP	AUC	R-Precision	P@30	P@50	P@100
arm	Random	84.3 ±4.1	59.6 ±8.1	**80.9** ±2.2	89.5 ±10.8	87.7 ±9.6	85.4 ±7.2
	HITS	**85.9**	**59.7**	79.3	**100**	**98.0**	**89.0**
bank	Random	74.8 ±6.5	61.6 ±9.6	72.6 ±4.5	82.9 ±14.8	80.1 ±13.5	76.6 ±10.9
	HITS	**84.8**	**77.6**	**78.0**	**100**	**100**	**94.0**
degree	Random	**69.4** ±3.0	**54.3** ±4.2	**66.7** ±2.3	**76.8** ±9.5	**73.8** ±7.5	**70.5** ±5.3
	HITS	62.4	49.3	63.2	56.7	64.0	66.0
difference	Random	48.3 ±3.8	54.5 ±5.0	47.0 ±4.4	53.9 ±10.7	50.7 ±8.8	47.9 ±6.1
	HITS	**50.2**	**60.1**	**51.1**	**60.0**	**60.0**	**48.0**
paper	Random	75.2 ±4.1	56.4 ±7.1	71.6 ±3.3	**82.3** ±9.8	79.6 ±8.8	76.9 ±6.1
	HITS	75.2	**61.0**	**75.2**	73.3	**80.0**	**78.0**
party	Random	79.1 ±5.0	57.0 ±9.7	76.6 ±3.1	84.5 ±10.7	82.7 ±9.2	80.2 ±7.5
	HITS	**85.2**	**68.2**	**78.5**	**100**	**96.0**	**87.0**
shelter	Random	74.9 ±2.3	51.5 ±3.3	**73.2** ±1.3	**77.3** ±7.8	76.0 ±5.6	74.5 ±3.5
	HITS	**77.0**	**54.6**	72.0	76.7	**84.0**	**79.0**
line	Random	44.5 ±15.1	36.3 ±16.9	40.1 ±14.6	75.0 ±21.0	69.8 ±24.1	62.3 ±27.9
	HITS	**72.2**	**68.6**	**68.5**	**100**	**100**	**100**
interest	Random	64.9 ±8.3	64.9 ±12.0	63.7 ±10.2	87.6 ±13.2	85.3 ±13.7	**81.2** ±13.9
	HITS	**75.3**	**83.0**	**80.1**	**100**	**94.0**	77.0
Avg.	Random	68.4	55.1	65.8	78.9	76.2	72.8
	HITS	**74.2**	**64.7**	**71.8**	**85.2**	**86.2**	**79.8**

Table 1: Comparison of seed selection for Espresso ($\tau = 5$, $n_{\text{seed}} = 7$). For Random, results are reported as (mean ± standard deviation). All figures are expressed in percentage terms. The row labeled "Avg." lists the values macro-averaged over the nine tasks.

1994) datasets[1] for our experiments. We lowercased words in the sentence and pre-processed them with the Porter stemmer (Porter, 1980) to get the stems of words.

Following (Komachi et al., 2008), we used two types of features extracted from neighboring contexts: collocational features and bag-of-words features. For collocational features, we set a window of three words to the right and left of the target word.

4.2 Evaluation methodology

We run Espresso on the above datasets using different seed selection methods (for majority sense of target words), and with or without stop lists created by our method (for minority senses of target words).

We evaluate the performance of the systems according to the following evaluation metrics: mean average precision (MAP), area under the ROC curve (AUC), R-precision, and precision@n (P@n) (Manning et al., 2008). The output of Espresso may contain seed instances input to the system, but seeds are excluded from the evaluation.

[1] http://www.d.umn.edu/~tpederse/data.html

5 Results and Discussion

5.1 Effect of Seed Selection

We first evaluate the performance of our seed selection method for the majority sense of the nine polysemous nouns. Table 1 shows the performance of Espresso with the seeds chosen by the proposed HITS-based seed selection method (HITS), and with the seed sets randomly chosen from the gold standard sets (Random; baseline). The results for Random were averaged over 1000 runs. We set the number of seeds $n_{\text{seed}} = 7$ and number of iterations $\tau = 5$ in this experiment.

As shown in the table, HITS outperforms the baseline systems except *degree*. Especially, the MAP reported in Table 1 shows that our approach achieved improvements of 10 percentage points on *bank*, 6.1 points on *party*, 27.7 points on *line*, and 10.4 points on *interest* over the baseline, respectively. AUC and R-precision mostly exhibit a trend similar to MAP, except R-precision in *arm* and *shelter*, for which the baseline is better. It can be seen from the P@n (P@30, P@50 and P@100) reported in Table 1 that our approach performed considerably better than baseline, e.g., around 17–20 points above

Task	Method	MAP	AUC	R-Precision	P@10	P@20	P@30
arm	NoStop	12.7 ±4.3	51.8 ±10.8	13.9 ±9.8	21.4 ±19.1	15.1 ±12.0	14.1 ±10.4
	HITS	**13.4** ±4.1	**53.7** ±10.5	**15.0** ±9.5	**23.8** ±17.7	**17.5** ±12.0	**15.5** ±10.2
bank	NoStop	32.5 ±5.1	73.0 ±8.5	45.1 ±10.3	80.4 ±21.8	70.3 ±21.2	62.6 ±18.1
	HITS	**33.7** ±3.7	**75.4** ±5.7	**47.6** ±8.1	**82.6** ±18.1	**72.7** ±18.5	**65.3** ±15.5
degree	NoStop	34.7 ±4.2	69.7 ±5.6	43.0 ±7.1	70.0 ±18.7	62.8 ±15.7	55.8 ±14.3
	HITS	**35.7** ±4.3	**71.7** ±5.6	**44.3** ±7.6	**72.4** ±16.4	**64.4** ±15.9	**58.3** ±16.2
difference	NoStop	20.2 ±3.9	57.1 ±6.7	22.3 ±8.3	35.8 ±18.7	27.7 ±14.0	25.5 ±11.9
	HITS	**21.2** ±3.8	**59.1** ±6.3	**24.2** ±8.4	**38.2** ±20.5	**30.2** ±14.0	**28.0** ±11.9
paper	NoStop	25.9 ±6.6	53.1 ±10.0	27.7 ±9.8	55.2 ±34.7	42.4 ±25.4	36.0 ±17.8
	HITS	**27.2** ±6.3	**56.3** ±9.1	**29.4** ±9.5	**57.4** ±35.3	**45.6** ±25.3	**38.7** ±17.5
party	NoStop	23.0 ±5.3	59.4 ±10.8	30.5 ±9.1	59.6 ±25.8	46.8 ±17.4	38.7 ±12.7
	HITS	**24.1** ±5.0	**62.5** ±9.8	**32.1** ±9.4	**61.6** ±26.4	**47.9** ±16.6	**40.8** ±12.7
shelter	NoStop	24.3 ±2.4	50.6 ±3.2	25.1 ±4.6	25.4 ±11.7	26.9 ±10.3	25.9 ±8.7
	HITS	**25.6** ±2.3	**53.4** ±3.0	**26.5** ±4.8	**28.8** ±12.9	**29.0** ±10.4	**28.1** ±8.2
line	NoStop	6.5 ±1.8	38.3 ±5.3	2.1 ±4.1	0.8 ±4.4	1.8 ±8.9	2.3 ±11.0
	HITS	**6.7** ±1.9	**38.8** ±5.8	**2.4** ±4.4	**1.0** ±4.6	**2.0** ±8.9	**2.5** ±11.1
interest	NoStop	29.4 ±7.6	61.0 ±12.1	33.7 ±13.2	69.6 ±40.3	67.0 ±39.1	65.7 ±37.8
	HITS	**31.2** ±5.6	**63.6** ±9.1	**36.1** ±10.5	**81.0** ±29.4	**78.1** ±27.0	**77.4** ±24.3
Avg.	NoStop	23.2	57.1	27.0	46.5	40.1	36.3
	HITS	**24.3**	**59.4**	**28.6**	**49.6**	**43.0**	**39.4**

Table 2: Effect of stop lists for Espresso ($n_{\text{stop}} = 10$, $n_{\text{seed}} = 10$, $\tau = 20$). Results are reported as (mean ± standard deviation). All figures are expressed in percentage. The row labeled "Avg." shows the values macro-averaged over all nine tasks.

the baseline on *bank* and 25–37 points on *line*.

5.2 Effect of Stop List

Table 2 shows the performance of Espresso using the stop list built with our proposed method (HITS), compared with the vanilla Espresso not using any stop list (NoStop).

In this case, the size of the stop list is set to $n_{\text{stop}} = 10$, and the number of seeds $n_{\text{seed}} = 10$ and iterations $\tau = 20$. For both HITS and NoStop, the seeds are selected at random from the gold standard data, and the reported results were averaged over 50 runs of each system. Due to lack of space, only the results for the second most frequent sense for each word are reported; i.e., the results for more minor senses are not in the table. However, they also showed a similar trend.

As shown in the table, our method (HITS) outperforms the baseline not using a stop list (NoStop), in all evaluation metrics. In particular, the P@n listed in Table 2 shows that our method provides about 11 percentage points absolute improvement over the baseline on *interest*, for all $n = 10$, 20, and 30.

6 Conclusions

We have proposed a HITS-based method for alleviating semantic drift in the bootstrapping algorithm Espresso. Our idea is built around the concept of *hubs* in the sense of Kleinberg's HITS algorithm, as well as the algorithmic similarity between Espresso and HITS. Hub instances are influential and hence make good seeds if they are of the target semantic class, but otherwise, they may trigger semantic drift. We have demonstrated that our method works effectively on lexical sample tasks. We are currently evaluating our method on other bootstrapping tasks, including named entity extraction.

Acknowledgements

We thank Masayuki Asahara and Kazuo Hara for helpful discussions and the anonymous reviewers for valuable comments. MS was partially supported by Kakenhi Grant-in-Aid for Scientific Research C 21500141.

References

Shuya Abe, Kentaro Inui, and Yuji Matsumoto. 2008. Acquiring event relation knowledge by learning cooccurrence patterns and fertilizing cooccurrence samples with verbal nouns. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP '08)*, pages 497–504.

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30:365–395.

Krishna Bharat and Monika R. Henzinger. 1998. Improved algorithms for topic distillation environment in a hyperlinked. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 104–111.

Rebecca Bruce and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94)*, pages 139–146.

Stephen Clark, James R. Curran, and Miles Osborne. 2003. Bootstrapping POS taggers using unlabelled data. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL '03)*, pages 49–55.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC '99)*, pages 189–196.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING '07)*, pages 172–180.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING '92)*, pages 539–545.

Ashwin Ittoo and Gosse Bouma. 2010. On learning subtypes of the part-whole relation: do not mix your seeds. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*, pages 1328–1336.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing*. Prentice Hall, 2nd edition.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Mamoru Komachi and Hisami Suzuki. 2008. Minimally supervised learning of semantic knowledge from query logs. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP '08)*, pages 358–365.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 1011–1020.

Claudia Leacock, Geoffrey Towell, and Ellen Voorhees. 1993. Corpus-based statistical sense resolution. In *Proceedings of the ARPA Workshop on Human Language Technology (HLT '93)*, pages 260–265.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*, pages 152–159.

Tara McIntosh and James R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP '09)*, volume 1, pages 396–404.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 356–365.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '06)*, pages 113–120.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence (AAAI/IAAI '99)*, pages 474–479.

Mark Steedman, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Example

selection for bootstrapping statistical parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL '03)*, volume 1, pages 157–164.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP '02)*, pages 214–221.

Vishnu Vyas, Patrick Pantel, and Eric Crestan. 2009. Helping editors choose better seed sets for entity set expansion. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*, pages 225–234.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING '02)*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics (ACL '95)*, pages 189–196.

Minoru Yoshida, Masaki Ikeda, Shingo Ono, Issei Sato, and Hiroshi Nakagawa. 2010. Person name disambiguation by bootstrapping. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pages 10–17.