Improving the Use of Pseudo-Words for Evaluating Selectional Preferences

Nathanael Chambers and Dan Jurafsky
Department of Computer Science
Stanford University
{natec,jurafsky}@stanford.edu

Abstract

This paper improves the use of pseudo-words as an evaluation framework for selectional preferences. While pseudo-words originally evaluated word sense disambiguation, they are now commonly used to evaluate selectional preferences. A selectional preference model ranks a set of possible arguments for a verb by their semantic fit to the verb. Pseudo-words serve as a proxy evaluation for these decisions. The evaluation takes an argument of a verb like *drive* (e.g. *car*), pairs it with an alternative word (e.g. *car/rock*), and asks a model to identify the original. This paper studies two main aspects of pseudo-word creation that affect performance results. (1) Pseudo-word evaluations often evaluate only a subset of the words. We show that selectional preferences should instead be evaluated on the data in its entirety. (2) Different approaches to selecting partner words can produce overly optimistic evaluations. We offer suggestions to address these factors and present a simple baseline that outperforms the state-of-the-art by 13% absolute on a newspaper domain.

1 Introduction

For many natural language processing (NLP) tasks, particularly those involving meaning, creating labeled test data is difficult or expensive. One way to mitigate this problem is with pseudo-words, a method for automatically creating test corpora without human labeling, originally proposed for word sense disambiguation (Gale et al., 1992; Schutze, 1992). While pseudo-words are now less often used for word sense disambigation, they are a common way to evaluate *selectional preferences*, models that measure the strength of association between a predicate and its argument filler, e.g., that the noun *lunch* is a likely object of *eat*. Selectional preferences are useful for NLP tasks such as parsing and semantic role labeling (Zapirain et al., 2009). Since evaluating them in isolation is difficult without labeled data, pseudo-word evaluations can be an attractive evaluation framework.

Pseudo-word evaluations are currently used to evaluate a variety of language modeling tasks (Erk, 2007; Bergsma et al., 2008). However, *evaluation design varies across research groups*. This paper studies the evaluation itself, showing how choices can lead to overly optimistic results if the evaluation is not designed carefully. We show in this paper that current methods of applying pseudo-words to selectional preferences vary greatly, and suggest improvements.

A pseudo-word is the concatenation of two words (e.g. *house/car*). One word is the original in a document, and the second is the *confounder*. Consider the following example of applying pseudo-words to the selectional restrictions of the verb *focus*:

***Original**: This story focuses on the campaign.*
***Test**: This story/part focuses on the campaign/meeting.*

In the original sentence, *focus* has two arguments: a subject *story* and an object *campaign*. In the test sentence, each argument of the verb is replaced by pseudo-words. A model is evaluated by its success at determining which of the two arguments is the original word.

Two problems exist in the current use of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 445–453,
Uppsala, Sweden, 11-16 July 2010.

pseudo-words to evaluate selectional preferences. First, selectional preferences historically focus on subsets of data such as *unseen* words or words in certain frequency ranges. While work on unseen data is important, evaluating on the entire dataset provides an accurate picture of a model's overall performance. Most other NLP tasks today evaluate *all* test examples in a corpus. We will show that *seen* arguments actually dominate newspaper articles, and thus propose creating test sets that include all verb-argument examples to avoid artificial evaluations.

Second, pseudo-word evaluations vary in how they choose confounders. Previous work has attempted to maintain a similar corpus frequency to the original, but it is not clear how best to do this, nor how it affects the task's difficulty. We argue in favor of using nearest-neighbor frequencies and show how using random confounders produces overly optimistic results.

Finally, we present a surprisingly simple baseline that outperforms the state-of-the-art and is far less memory and computationally intensive. It outperforms current similarity-based approaches by over 13% when the test set includes all of the data. We conclude with a suggested backoff model based on this baseline.

2 History of Pseudo-Word Disambiguation

Pseudo-words were introduced simultaneously by two papers studying statistical approaches to word sense disambiguation (WSD). Schütze (1992) simply called the words, 'artificial ambiguous words', but Gale et al. (1992) proposed the succinct name, *pseudo-word*. Both papers cited the sparsity and difficulty of creating large labeled datasets as the motivation behind pseudo-words. Gale et al. selected unambiguous words from the corpus and paired them with random words from different thesaurus categories. Schütze paired his words with confounders that were 'comparable in frequency' and 'distinct semantically'. Gale et al.'s pseudo-word term continues today, as does Schütze's frequency approach to selecting the confounder.

Pereira et al. (1993) soon followed with a selectional preference proposal that focused on a language model's effectiveness on *unseen* data. The work studied clustering approaches to assist in similarity decisions, predicting which of two verbs was the correct predicate for a given noun object. One verb v was the original from the source document, and the other v' was randomly generated. This was the first use of such verb-noun pairs, as well as the first to test only on *unseen* pairs.

Several papers followed with differing methods of choosing a test pair (v, n) and its confounder v'. Dagan et al. (1999) tested all *unseen* (v, n) occurrences of the most frequent 1000 verbs in his corpus. They then sorted verbs by corpus frequency and chose the neighboring verb v' of v as the confounder to ensure the closest frequency match possible. Rooth et al. (1999) tested 3000 random (v, n) pairs, but required the verbs and nouns to appear between 30 and 3000 times in training. They also chose confounders randomly so that the new pair was *unseen*.

Keller and Lapata (2003) specifically addressed the impact of *unseen* data by using the web to first 'see' the data. They evaluated unseen pseudo-words by attempting to first observe them in a larger corpus (the Web). One modeling difference was to disambiguate the nouns as selectional preferences instead of the verbs. Given a test pair (v, n) and its confounder (v, n'), they used web searches such as "v Det n" to make the decision. Results beat or matched current results at the time. We present a similarly motivated, but new web-based approach later.

Very recent work with pseudo-words (Erk, 2007; Bergsma et al., 2008) further blurs the lines between what is included in training and test data, using frequency-based and semantic-based reasons for deciding what is included. We discuss this further in section 5.

As can be seen, there are two main factors when devising a pseudo-word evaluation for selectional preferences: (1) choosing (v, n) pairs from the test set, and (2) choosing the confounding n' (or v'). The confounder has not been looked at in detail and as best we can tell, these factors have varied significantly. Many times the choices are well motivated based on the paper's goals, but in other cases the motivation is unclear.

3 How Frequent is Unseen Data?

Most NLP tasks evaluate their entire datasets, but as described above, most selectional preference evaluations have focused only on unseen data. This section investigates the extent of unseen examples in a typical training/testing environment

of newspaper articles. The results show that even with a small training size, seen examples dominate the data. We argue that, absent a system's need for specialized performance on unseen data, a representative test set should include the dataset in its entirety.

3.1 Unseen Data Experiment

We use the New York Times (NYT) and Associated Press (APW) sections of the Gigaword Corpus (Graff, 2002), as well as the British National Corpus (BNC) (Burnard, 1995) for our analysis. Parsing and SRL evaluations often focus on newspaper articles and Gigaword is large enough to facilitate analysis over varying amounts of training data. We parsed the data with the Stanford Parser[1] into dependency graphs. Let (v_d, n) be a verb v with grammatical dependency $d \in \{subject, object, prep\}$ filled by noun n. Pairs (v_d, n) are chosen by extracting every such dependency in the graphs, setting the head predicate as v and the head word of the dependent d as n. All prepositions are condensed into *prep*.

We randomly selected documents from the year 2001 in the NYT portion of the corpus as development and test sets. Training data for APW and NYT include all years 1994-2006 (minus NYT development and test documents). We also identified and removed duplicate documents[2]. The BNC in its entirety is also used for training as a single data point. We then record every seen (v_d, n) pair during training that is seen two or more times[3] and then count the number of unseen pairs in the NYT development set (1455 tests).

Figure 1 plots the percentage of unseen arguments against training size when trained on either NYT or APW (the APW portion is smaller in total size, and the smaller BNC is provided for comparison). The first point on each line (the highest points) contains approximately the same number of words as the BNC (100 million). Initially, about one third of the arguments are unseen, but that percentage quickly falls close to 10% as additional training is included. This suggests that an evaluation focusing only on unseen data is not representative, potentially missing up to 90% of the data.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] Any two documents whose first two paragraphs in the corpus files are identical.

[3] Our results are thus conservative, as including all single occurrences would achieve even smaller unseen percentages.

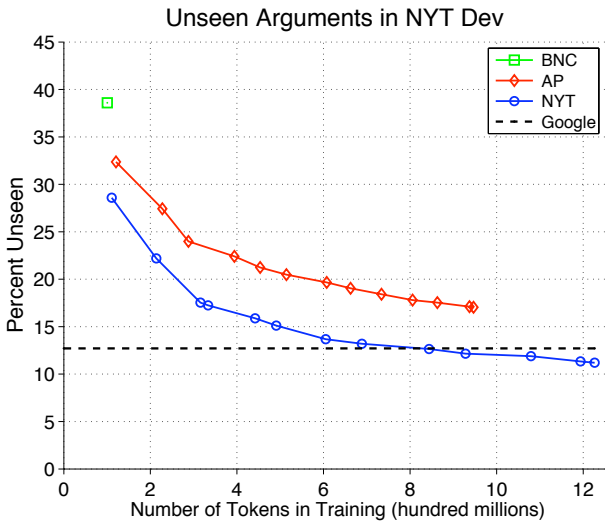

Figure 1: Percentage of NYT development set that is unseen when trained on varying amounts of data. The two lines represent training with NYT or APW data. The APW set is smaller in size from the NYT. The dotted line uses Google n-grams as training. The x-axis represents tokens $\times 10^8$.

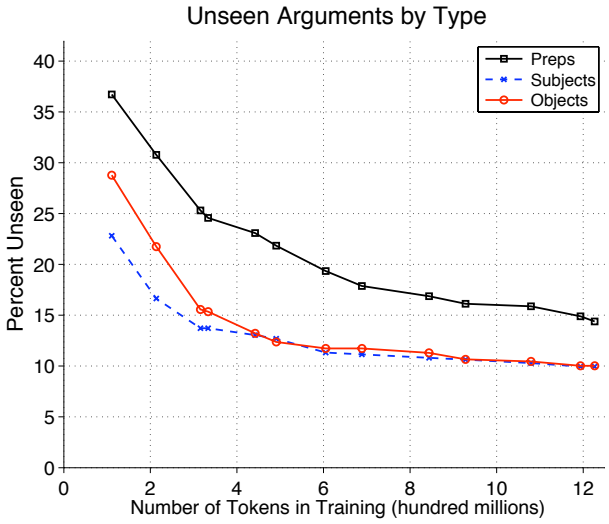

Figure 2: Percentage of subject/object/preposition arguments in the NYT development set that is unseen when trained on varying amounts of NYT data. The x-axis represents tokens $\times 10^8$.

The third line across the bottom of the figure is the number of unseen pairs using Google n-gram data as proxy argument counts. Creating argument counts from n-gram counts is described in detail below in section 5.2. We include these Web counts to illustrate how an openly available source of counts affects unseen arguments. Finally, figure 2 compares which dependency types are seen the least in training. Prepositions have the largest unseen percentage, but not surprisingly, also make up less of the training examples overall.

In order to analyze why pairs are unseen, we analyzed the distribution of *rare* words across unseen and seen examples. To define rare nouns, we order head words by their individual corpus frequencies. A noun is *rare* if it occurs in the lowest 10% of the list. We similarly define *rare verbs* over their ordered frequencies (we count verb lemmas, and do not include the syntactic relations). Corpus counts covered 2 years of the AP section, and we used the development set of the NYT section to extract the seen and unseen pairs. Figure 3 shows the percentage of rare nouns and verbs that occur in unseen and seen pairs. 24.6% of the verbs in unseen pairs are rare, compared to only 4.5% in seen pairs. The distribution of rare nouns is less contrastive: 13.3% vs 8.9%. This suggests that many unseen pairs are unseen mainly because they contain low-frequency verbs, rather than because of containing low-frequency argument heads.

Given the large amount of seen data, we believe evaluations should include all data examples to best represent the corpus. We describe our full evaluation results and include a comparison of different training sizes below.

4 How to Select a Confounder

Given a test set S of pairs $(v_d, n) \in S$, we now address how best to select a confounder n'. Work in WSD has shown that confounder choice can make the pseudo-disambiguation task significantly easier. Gaustad (2001) showed that human-generated pseudo-words are more difficult to classify than random choices. Nakov and Hearst (2003) further illustrated how random confounders are easier to identify than those selected from semantically ambiguous, yet related concepts. Our approach evaluates selectional preferences, not WSD, but our results complement these findings.

We identified three methods of confounder selection based on varying levels of corpus frequency: (1) choose a *random* noun, (2) choose a random noun from a *frequency bucket* similar to the original noun's frequency, and (3) select the *nearest neighbor*, the noun with frequency closest to the original. These methods evaluate the range of choices used in previous work. Our experiments compare the three.

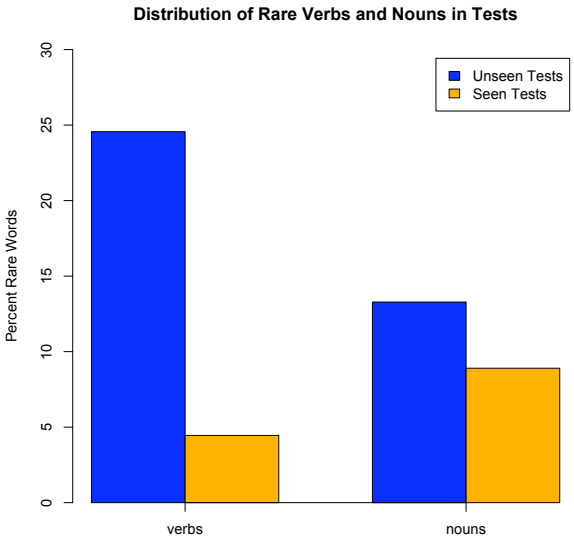

Figure 3: Comparison between seen and unseen tests (verb,relation,noun). 24.6% of unseen tests have rare verbs, compared to just 4.5% in seen tests. The rare nouns are more evenly distributed across the tests.

5 Models

5.1 A New Baseline

The analysis of unseen slots suggests a baseline that is surprisingly obvious, yet to our knowledge, has not yet been evaluated. Part of the reason is that early work in pseudo-word disambiguation explicitly tested only unseen pairs[4]. Our evaluation will include seen data, and since our analysis suggests that up to 90% is seen, a strong baseline should address this seen portion.

[4]Recent work does include some seen data. Bergsma et al. (2008) test pairs that fall below a mutual information threshold (might include some seen pairs), and Erk (2007) selects a subset of roles in FrameNet (Baker et al., 1998) to test and uses all labeled instances within this subset (unclear what portion of subset of data is seen). Neither evaluates all of the seen data, however.

We propose a conditional probability baseline:

$$P(n|v_d) = \begin{cases} \frac{C(v_d,n)}{C(v_d,*)} & \text{if } C(v_d,n) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $C(v_d, n)$ is the number of times the head word n was seen as an argument to the predicate v, and $C(v_d, *)$ is the number of times v_d was seen with any argument. Given a test (v_d, n) and its confounder (v_d, n'), choose n if $P(n|v_d) > P(n'|v_d)$, and n' otherwise. If $P(n|v_d) = P(n'|v_d)$, randomly choose one.

Lapata et al. (1999) showed that corpus frequency and conditional probability correlate with human decisions of adjective-noun plausibility, and Dagan et al. (1999) appear to propose a very similar baseline for verb-noun selectional preferences, but the paper evaluates *unseen* data, and so the conditional probability model is not studied.

We later analyze this baseline against a more complicated smoothing approach.

5.2 A Web Baseline

If conditional probability is a reasonable baseline, better performance may just require more data. Keller and Lapata (2003) proposed using the web for this task, querying for specific phrases like 'Verb Det N' to find syntactic objects. Such a web corpus would be attractive, but we'd like to find subjects and prepositional objects as well as objects, and also ideally we don't want to limit ourselves to patterns. Since parsing the web is unrealistic, a reasonable compromise is to make rough counts when pairs of words occur in close proximity to each other.

Using the Google n-gram corpus, we recorded all verb-noun co-occurrences, defined by appearing in any order in the same n-gram, up to and including 5-grams. For instance, the test pair $(throw_{subject}, ball)$ is considered *seen* if there exists an n-gram such that *throw* and *ball* are both included. We count all such occurrences for all verb-noun pairs. We also avoided over-counting co-occurrences in lower order n-grams that appear again in 4 or 5-grams. This crude method of counting has obvious drawbacks. Subjects are not distinguished from objects and nouns may not be actual arguments of the verb. However, it is a simple baseline to implement with these freely available counts.

Thus, we use conditional probability as defined in the previous section, but define the count $C(v_d, n)$ as the number of times v and n (ignoring d) appear in the same n-gram.

5.3 Smoothing Model

We implemented the current state-of-the-art smoothing model of Erk (2007). The model is based on the idea that the arguments of a particular verb slot tend to be similar to each other. Given two potential arguments for a verb, the correct one should correlate higher with the arguments observed with the verb during training.

Formally, given a verb v and a grammatical dependency d, the score for a noun n is defined:

$$S_{v_d}(n) = \sum_{w \in Seen(v_d)} sim(n,w) * C(v_d, w) \quad (1)$$

where $sim(n, w)$ is a noun-noun similarity score, $Seen(v_d)$ is the set of seen head words filling the slot v_d during training, and $C(v_d, n)$ is the number of times the noun n was seen filling the slot v_d The similarity score $sim(n, w)$ can thus be one of many vector-based similarity metrics[5]. We evaluate both Jaccard and Cosine similarity scores in this paper, but the difference between the two is small.

6 Experiments

Our training data is the NYT section of the Gigaword Corpus, parsed into dependency graphs. We extract all (v_d, n) pairs from the graph, as described in section 3. We randomly chose 9 documents from the year 2001 for a development set, and 41 documents for testing. The test set consisted of 6767 (v_d, n) pairs. All verbs and nouns are stemmed, and the development and test documents were isolated from training.

6.1 Varying Training Size

We repeated the experiments with three different training sizes to analyze the effect data size has on performance:

- **Train x1**: Year 2001 of the NYT portion of the Gigaword Corpus. After removing duplicate documents, it contains approximately 110 million tokens, comparable to the 100 million tokens in the BNC corpus.

[5] A similar type of smoothing was proposed in earlier work by Dagan et al. (1999). A noun is represented by a vector of verb slots and the number of times it is observed filling each slot.

- **Train x2**: Years 2001 and 2002 of the NYT portion of the Gigaword Corpus, containing approximately 225 million tokens.
- **Train x10**: The entire NYT portion of Gigaword (approximately 1.2 billion tokens). It is an order of magnitude larger than *Train x1*.

6.2 Varying the Confounder

We generated three different confounder sets based on word corpus frequency from the 41 test documents. Frequency was determined by counting all tokens with noun POS tags. As motivated in section 4, we use the following approaches:

- **Random**: choose a random confounder from the set of nouns that fall within some broad corpus frequency range. We set our range to eliminate (approximately) the top 100 most frequent nouns, but otherwise arbitrarily set the lower range as previous work seems to do. The final range was $[30, 400000]$.
- **Buckets**: all nouns are bucketed based on their corpus frequencies[6]. Given a test pair (v_d, n), choose the bucket in which n belongs and randomly select a confounder n' from that bucket.
- **Neighbor**: sort all seen nouns by frequency and choose the confounder n' that is the nearest neighbor of n with greater frequency.

6.3 Model Implementation

None of the models can make a decision if they identically score both potential arguments (most often true when both arguments were not seen with the verb in training). As a result, we extend all models to randomly guess (50% performance) on pairs they cannot answer.

The conditional probability is reported as **Baseline**. For the web baseline (reported as **Google**), we stemmed all words in the Google n-grams and counted every verb v and noun n that appear in Gigaword. Given two nouns, the noun with the higher co-occurrence count with the verb is chosen. As with the other models, if the two nouns have the same counts, it randomly guesses.

The smoothing model is named **Erk** in the results with both Jaccard and Cosine as the similarity metric. Due to the large vector representations of the nouns, it is computationally wise to trim their vectors, but also important to do so for best performance. A noun's representative vector consists of verb slots and the number of times the noun was seen in each slot. We removed any verb slot not seen more than x times, where x varied based on all three factors: the dataset, confounder choice, and similarity metric. We optimized x on the development data with a linear search, and used that cutoff on each test. Finally, we trimmed any vectors over 2000 in size to reduce the computational complexity. Removing this strict cutoff appears to have little effect on the results.

Finally, we report backoff scores for Google and Erk. These consist of always choosing the Baseline if it returns an answer (not a guessed unseen answer), and then backing off to the Google/Erk result for Baseline unknowns. These are labeled **Backoff Google** and **Backoff Erk**.

7 Results

Results are given for the two dimensions: *confounder choice* and *training size*. Statistical significance tests were calculated using the approximate randomization test (Yeh, 2000) with 1000 iterations.

Figure 4 shows the performance change over the different confounder methods. *Train x2* was used for training. Each model follows the same progression: it performs extremely well on the random test set, worse on buckets, and the lowest on the nearest neighbor. The conditional probability **Baseline** falls from 91.5 to 79.5, a 12% absolute drop from completely random to neighboring frequency. The **Erk** smoothing model falls 27% from 93.9 to 68.1. The **Google** model generally performs the worst on all sets, but its 74.3% performance with random confounders is significantly better than a 50-50 random choice. This is notable since the Google model only requires n-gram counts to implement. The **Backoff Erk** model is the best, using the **Baseline** for the majority of decisions and backing off to the Erk smoothing model when the Baseline cannot answer.

Figure 5 (shown on the next page) varies the training size. We show results for both Bucket Frequencies and Neighbor Frequencies. The only difference between columns is the amount of training data. As expected, the Baseline improves as the training size is increased. The Erk model, somewhat surprisingly, shows no continual gain with more training data. The Jaccard and Cosine simi-

[6] We used frequency buckets of 4, 10, 25, 200, 1000, >1000. Adding more buckets moves the evaluation closer to *Neighbor*, less is closer to *Random*.

Varying the Confounder Frequency

	Random	Buckets	Neighbor
Baseline	**91.5**	89.1	79.5
Erk-Jaccard	**93.9***	82.7*	68.1*
Erk-Cosine	**91.2**	81.8*	65.3*
Google	**74.3***	70.4*	59.4*
Backoff Erk	**96.6***	91.8*	80.8*
Backoff Goog	**92.7†**	89.7	79.8

Figure 4: Trained on two years of NYT data (Train x2). Accuracy of the models on the same NYT test documents, but with three different ways of choosing the confounders. * indicates statistical significance with the column's Baseline at the $p < 0.01$ level, † at $p < 0.05$. Random is overly optimistic, reporting performance far above more conservative (selective) confounder choices.

Baseline Details

	Train	Train x2	Train x10
Precision	96.1	95.5*	95.0†
Accuracy	78.2	82.0*	88.1*
Accuracy +50%	87.5	89.1*	91.7*

Figure 6: Results from the *buckets* confounder test set. Baseline precision, accuracy (the same as recall), and accuracy when you randomly guess the tests that Baseline does not answer. All numbers are statistically significant * with p-value < 0.01 from the number to their left.

larity scores perform similarly in their model. The Baseline achieves the highest accuracies (91.7% and 81.2%) with *Train x10*, outperforming the best Erk model by 5.2% and 13.1% absolute on buckets and nearest neighbor respectively. The backoff models improve the baseline by just under 1%. The Google n-gram backoff model is almost as good as backing off to the Erk smoothing model.

Finally, figure 6 shows the Baseline's precision and overall accuracy. Accuracy is the same as recall when the model does not guess between pseudo words that have the same conditional probabilities. *Accuracy +50%* (the full Baseline in all other figures) shows the gain from randomly choosing one of the two words when uncertain. Precision is extremely high.

8 Discussion

Confounder Choice: Performance is strongly influenced by the method used when choosing confounders. This is consistent with findings for WSD that corpus frequency choices alter the task (Gaustad, 2001; Nakov and Hearst, 2003). Our results show the gradation of performance as one moves across the spectrum from completely random to closest in frequency. The Erk model dropped 27%, Google 15%, and our baseline 12%. The overly optimistic performance on random data suggests using the nearest neighbor approach for experiments. Nearest neighbor avoids evaluating on 'easy' datasets, and our baseline (at 79.5%) still provides room for improvement. But perhaps just as important, the nearest neighbor approach facilitates the most reproducibile results in experiments since there is little ambiguity in how the confounder is selected.

Realistic Confounders: Despite its overoptimism, the random approach to confounder selection may be the correct approach in some circumstances. For some tasks that need selectional preferences, random confounders may be more realistic. It's possible, for example, that the options in a PP-attachment task might be distributed more like the random rather than nearest neighbor models. In any case, this is difficult to decide without a specific application in mind. Absent such specific motiviation, a nearest neighbor approach is the most conservative, and has the advantage of creating a reproducible experiment, whereas random choice can vary across design.

Training Size: Training data improves the conditional probability baseline, but does not help the smoothing model. Figure 5 shows a lack of improvement across training sizes for both jaccard and cosine implementations of the Erk model. The *Train x1* size is approximately the same size used in Erk (2007), although on a different corpus. We optimized argument cutoffs for each training size, but the model still appears to suffer from additional noise that the conditional probability baseline does not. This may suggest that observing a test argument with a verb in training is more reliable than a smoothing model that compares all training arguments against that test example.

High Precision Baseline: Our conditional probability baseline is very precise. It outperforms the smoothed similarity based **Erk** model and gives high results across tests. The only combination when Erk is better is when the training data includes just one year (one twelfth of the NYT section) and the confounder is chosen com-

Varying the Training Size

	Bucket Frequency			Neighbor Frequency		
	Train x1	Train x2	Train x10	Train x1	Train x2	Train x10
Baseline	87.5	89.1	**91.7**	78.4	79.5	**81.2**
Erk-Jaccard	86.5*	82.7*	83.1*	66.8*	68.1*	65.5*
Erk-Cosine	82.1*	81.8*	81.1*	66.1*	65.3*	65.7*
Google	-	-	70.4*	-	-	59.4*
Backoff Erk	92.6*	91.8*	92.6*	79.4*	80.8*	**81.7***
Backoff Google	88.6	89.7	91.9†	78.7	79.8	81.2

Figure 5: Accuracy of varying NYT training sizes. The left and right tables represent two confounder choices: choose the confounder with frequency buckets, and choose by nearest frequency neighbor. Trainx1 starts with year 2001 of NYT data, Trainx2 doubles the size, and Trainx10 is 10 times larger. * indicates statistical significance with the column's Baseline at the $p < 0.01$ level, † at $p < 0.05$.

pletely randomly. These results appear consistent with Erk (2007) because that work used the BNC corpus (the same size as one year of our data) and Erk chose confounders randomly within a broad frequency range. Our reported results include every (v_d, n) in the data, not a subset of particular semantic roles. Our reported 93.9% for Erk-Jaccard is also significantly higher than their reported 81.4%, but this could be due to the random choices we made for confounders, or most likely corpus differences between Gigaword and the subset of FrameNet they evaluated.

Ultimately we have found that complex models for selectional preferences may not be necessary, depending on the task. The higher computational needs of smoothing approaches are best for backing off when unseen data is encountered. Conditional probability is the best choice for seen examples. Further, analysis of the data shows that as more training data is made available, the seen examples make up a much larger portion of the test data. Conditional probability is thus a very strong starting point if selectional preferences are an internal piece to a larger application, such as semantic role labeling or parsing.

Perhaps most important, these results illustrate the disparity in performance that can come about when designing a pseudo-word disambiguation evaluation. It is crucially important to be clear during evaluations about how the confounder was generated. We suggest the approach of sorting nouns by frequency and using a neighbor as the confounder. This will also help avoid evaluations that produce overly optimistic results.

9 Conclusion

Current performance on various natural language tasks is being judged and published based on pseudo-word evaluations. It is thus important to have a clear understanding of the evaluation's characteristics. We have shown that the evaluation is strongly affected by confounder choice, suggesting a nearest frequency neighbor approach to provide the most reproducible performance and avoid overly optimistic results. We have shown that evaluating entire documents instead of subsets of the data produces vastly different results. We presented a conditional probability baseline that is both novel to the pseudo-word disambiguation task and strongly outperforms state-of-the-art models on entire documents. We hope this provides a new reference point to the pseudo-word disambiguation task, and enables selectional preference models whose performance on the task similarly transfers to larger NLP applications.

Acknowledgments

This work was supported by the National Science Foundation IIS-0811974, and the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, ndings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reect the view of the AFRL. Thanks to Sebastian Padó, the Stanford NLP Group, and the anonymous reviewers for very helpful suggestions.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Empirical Methods in Natural Language Processing*, pages 59–68, Honolulu, Hawaii.

Lou Burnard. 1995. *User Reference Guide for the British National Corpus*. Oxford University Press, Oxford.

Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1):43–69.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

Tanja Gaustad. 2001. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *39th Annual Meeting of the Association for Computational Linguistics - Student Research Workshop*.

David Graff. 2002. English Gigaword. *Linguistic Data Consortium*.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Maria Lapata, Scott McDonald, and Frank Keller. 1999. Determinants of adjective-noun plausibility. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Preslav I. Nakov and Marti A. Hearst. 2003. Category-based pseudowords. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 67–69, Edmonton, Canada.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Hinrich Schutze. 1992. Context space. In *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 113–120.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *International Conference on Computational Linguistics (COLING)*.

Beat Zapirain, Eneko Agirre, and Llus Mrquez. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*, Singapore.