Bootstrapping Without the Boot*

Jason Eisner and **Damianos Karakos**
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{eisner,damianos}@jhu.edu

Abstract

"Bootstrapping" methods for learning require a small amount of supervision to seed the learning process. We show that it is sometimes possible to eliminate this last bit of supervision, by trying many candidate seeds and selecting the one with the most plausible outcome. We discuss such "strapping" methods in general, and exhibit a particular method for strapping word-sense classifiers for ambiguous words. Our experiments on the Canadian Hansards show that our unsupervised technique is significantly more effective than picking seeds by hand (Yarowsky, 1995), which in turn is known to rival supervised methods.

1 Introduction

Some of NLP's most interesting problems have to do with unsupervised learning. Human language learners are able to discover word senses, grammatical genders, morphological systems, grammars, discourse registers, and so forth. One would like to build systems that discover the same linguistic patterns in raw text. For that matter, one would also like to discover patterns in bilingual text (for translation), in document collections (for categorization and retrieval), and in other data that fall outside the scope of humans' language learning.

There are relatively few successful methods for fully unsupervised learning from raw text. For example, the EM algorithm (Dempster et al., 1977) extracts the "wrong" patterns or gets stuck in local maxima.

One of the most promising avenues in recent years has been the use of "minimally supervised" methods. Such methods are initialized with some sort of "seed" that grows into a full classifier (or generative model). We say that a seed is "fertile" if it grows into a classifier (or model) that performs well on some desired criterion.

Ordinarily, it is up to a human to choose a seed that he or she intuitively expects to be fertile. While this may be easy when building a single classifier, it is prohibitive when building many classifiers. For example, we may wish to build

- word-sense classifiers for all words of a language (e.g., to get sharper lexical translation probabilities in a machine translation system)
- named-entity extractors for many languages
- new clusters or classifiers every day (for an evolving document collection)
- new clusters or classifiers every minute (for the document sets retrieved by *ad hoc* queries)
- many distinct classifiers that correspond to different views of the data[1]

Even when building a single classifier, a human may not know how to pick a good seed when working with an unfamiliar language or sublanguage, or when trying to induce less intuitive hidden variables, such as grammar rules or fine-grained senses. And there is no reason to expect humans to have good intuitions about seeds for mining non-linguistic data such as consumer purchasing records.

This paper considers how to remove this last element of supervision. Our idea is to guess a number of plausible seeds, build a classifier for each one, and then try to determine which of the seeds have grown successfully.

For example, to discover the two senses of the English word *drug*, we grow 200 classifiers (from different seeds) that attempt to partition instances of *drug* into two classes. We have no *direct* supervision about which of the resulting partitions corresponds to the true sense distinction. Instead, we rely on clues that tend to signal that a seed was fertile and led to a good partition. The clues are not specific to the word *drug*, but they may have been demonstrated to be good clues in general for successfully grown word sense disambiguators.

Demonstrated how? If we consider more than one clue, we may need some data to learn which clues to trust, and their relative weights. Our method is unsupervised in the conventional sense, as it obtains a classifier for *drug* with no supervision about *drug*. However, to learn what good classifiers generally look like[2] for this task, we first use

*We thank David Yarowsky for advice on the choice of data and for the *plant/tank* dataset.

[1] A word token or document can be characterized by a 20-bit vector, corresponding to its classifications by 20 different binary classifiers. These vectors are detailed abstract representations of the words or documents. They can be clustered, or all their bits can be included as potentially relevant features in another task.

[2] Ando and Zhang (2005) independently used this phrase, for a *semi-supervised, cross-task* learner that differs from our *unsupervised, cross-instance* learner. Both their work and ours try to transfer knowledge to a target problem from many artificial supervised "auxiliary problems," which are generated from unlabeled data (e.g., our pseudoword disambiguation problems). However, in their "structural learning," the target problem is *supervised* (if inadequately), and the auxiliary problems (supervised instances of a *different* task) are a source of useful *hidden features for the classifier*. In our "strapping," the target task is *unsupervised*, and the auxiliary problems (supervised instances

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 395–402, Vancouver, October 2005. ©2005 Association for Computational Linguistics

supervised data for a few *other* ambiguous words—or ambiguous pseudowords, a kind of artificial data where supervision comes for free. This supervision's effect on *drug* might be called *cross-instance learning*.

To take another metaphor, minimally supervised learning is often called "bootstrapping." Our goal is to allow a method to pull itself up by its own bootstraps[3] even when it has none. It places its stocking feet in anything handy, pulls on what it hopes to be sturdy straps, and checks to see how high it got.

We dub this family of methods "bootstrapping without the boot," or "strapping" for short. The name is meant to evoke "bagging" and "boosting"—other methods that train and combine multiple classifiers of the same form. However, we are careful to point out that strapping, unlike those theoretically motivated methods, is an *unsupervised* learning technique (in the sense explained above). The clusters or other hidden variables extracted by the winning classifier may or may not be the ones that one had hoped to find. Designing a strapping algorithm for a particular task requires more art than designing a supervised learner: one must invent not only appropriate features for classifying the data, but also appropriate clues for identifying "successful" classifiers.

2 Bootstrapping

To show where strapping might be useful, we briefly review a range of successful bootstrapping work. We consider different *tasks*. Given an *instance* of the task and a *seed* s for that instance, one bootstraps a classifier C_s that can classify *examples* of the task instance.

2.1 The Yarowsky algorithm

Yarowsky (1995) sparked considerable interest in bootstrapping with his successful method for word sense disambiguation. An instance of this task involves a homonymous word such as *drug*. A seed for the instance is a pair of words that are strongly associated, respectively, with the two senses of *drug*, such as (*trafficking*, *therapy*). An example is a token of *drug*.

For our purposes, a bootstrapping method can be regarded almost as a black box. However, we review the details of the Yarowsky algorithm to illustrate how bootstrapping is possible, and why some seeds are better than others. We will use these intuitions later in designing a method to strap the Yarowsky algorithm on a new instance—i.e., a method for *automatically* choosing seeds that discover a true sense distinction.

A learned classifier for the instance *drug* is an ordered decision list of contextual features (such as the presence of *dealer* nearby) that strongly indicate one or the other sense of *drug*. Given a sample token of *drug*, the classifier picks a sense according to the single highest-ranked feature that is present in the token's context.

To bootstrap a decision-list classifier from a seed, Yarowsky starts with all examples of *drug* that can be classified by using the seed words as the only features. These few examples are used as supervised data to train a longer decision list, which includes the seed words and any other features that suffice to distinguish these examples with high confidence. This longer decision list can now classify further examples, which are used to train a new and even longer decision list, and so on.

Yarowsky's method works if it can maintain high accuracy as it gradually increases its coverage. A precise classifier at iteration t tends to accurately classify new examples. This tends to produce a still-accurate classifier with greater coverage at iteration $t + 1$.

The method fails if the initial classifier is inaccurate (i.e., if the two seed words do not accurately pick out examples of the two senses). It may also fail if at some point, by bad luck on sparse data, the process learns some inappropriate features. If the classifier at iteration t is sufficiently polluted by bad features, the classifier at iteration $t + 1$ will start trying to distinguish examples that do *not* correspond to different senses, which may lead to even worse classifiers on subsequent iterations. However, some alternative seed may have escaped this bad luck by sprouting a different set of examples.

2.2 A Few Other Applications of Bootstrapping

Inspired by Yarowsky, Blum and Mitchell (1998) built a classifier for the task of web page classification.[4] They considered only one instance of this task, namely distinguishing course home pages from other web pages at a computer science department. Their seed consisted of 3 positive and 9 negative examples. Strapping a web page classifier would mean identifying seeds that lead to other "natural classes" of web pages. Strapping may be useful for unsupervised text categorization in general.

Riloff et al. (2003) learned lists of subjective nouns in English, seeding their method with 20 high-frequency, strongly subjective words. This seed set was chosen manually from an automatically generated list of 850 can-

of the *same* task) are a source of *clues for a meta-classifier* that chooses among classifiers grown from different seeds. In short, their auxiliary problems help train the target classifier directly, while ours help train only a simple meta-classifier that chooses among many unsupervised target classifiers. We use far fewer auxiliary problems but ours must be instances of the target task.

[3]The reference is to Baron Munchausen, a fictional 18th-century adventurer who rescued himself from a pit in this way. It is distinct from the "bootstrap" in non-parametric statistics.

[4]More precisely, they bootstrapped *two* Naive Bayes classifiers—one that looked at page content and the other that looked at links to the page. This "co-training" approach has become popular. It was also used by the Cucerzan and Yarowsky papers below, which looked at "internal" and "external" features of a phrase.

didate words. Strapping their method would identify subjective nouns in other languages, or other "natural classes" of English words.

Query expansion in IR searches for more documents "similar to" a designated relevant document. This problem too might be regarded as searching for a natural class—a small subset of documents that share some property of the original document—and approached using iterative bootstrapping. The seed would specify the original document *plus* one or two additional words or documents initially associated with the "relevant" and/or "irrelevant" classes. Strapping would guess various different seeds that extended the original document, then try to determine which seeds found a *cohesive* "relevant set."

Collins and Singer (1999) bootstrapped a system for classifying phrases in context. Again, they considered only one instance of this task: classifying English proper names as persons, organizations, or locations. Their seed consisted of 7 simple rules ("that *New York, California,* and *U.S.* are locations; that any name containing *Incorporated* is an organization; and that *I.B.M.* and *Microsoft* are organizations"). Strapping such a classifier would automatically discover named-entity classes in a different language, or other phrase classes in English.

Cucerzan and Yarowsky (1999) built a similar system that identified proper names as well as classifying them. Their seed consisted of a list of 40 to 300 names. Large seeds were not necessary for precision but did help recall.

Cucerzan and Yarowsky (2003) classified masculine vs. feminine nouns. They experimented with several task instances, namely different Indo-European languages. In each instance, their seed consisted of up to 30 feminine and 30 masculine words (e.g., *girl, princess, father*).

Many more papers along these lines could be listed. A rather different task is grammar induction, where a task instance is a corpus of text in some language, and the learned classifier is a parser. Following Chomsky (1981), we suggest that it may be possible to seed a grammar induction method with a small number of facts about the word order of the language: the basic clause order (SVO, SOV, etc.), whether pronominal subjects may be omitted (Chomsky's "*pro*-drop" parameter), etc. These facts can for example be used to construct a starting point for the inside-outside algorithm (Baker, 1979), which like other EM algorithms is highly sensitive to starting point. In a strapping method, one would guess a number of different seeds and evaluate the learned grammars on likelihood, entropy (Wang et al., 2002), correlation with semantics, or plausibility on other linguistic grounds that were not considered by the likelihood or the prior.

3 Strapping

Given a seed s for some task instance, let C_s denote the classifier grown from s. Let $f(s)$ denote the true fertility of a seed s, i.e., the performance of C_s measured against some set of correct answers for this instance. In general, we do not know the correct answers and hence do not know $f(s)$. That is why we are doing *unsupervised* learning.

Strapping relies on two *estimates* of $f(s)$. Let $g(s)$ be a quick estimate that considers only superficial features of the seed s. $h(s)$ is a more careful estimate that can be computed once C_s has been grown.

The basic method for strapping a classifier for a new task instance is very simple:

1. Quickly select a set S of candidate seeds such that $g(s)$ is high.
2. For each seed $s \in S$, learn a classifier C_s and measure $h(s)$.
3. Choose the seed $\hat{s} \in S$ that maximizes $h(\hat{s})$.
4. Return $C_{\hat{s}}$.

Variants on this method are obviously possible. For example, instead of returning a single classifier $C_{\hat{s}}$, one might use classifier combination to combine several classifiers C_s that have high $h(s)$.

It is clearly important that g and h be good estimates of f. Can data help us design g and h? Unfortunately, f is not known in an unsupervised setting. However, if one can get a few *supervised* instances of the same task, then one can select g and h so $g(s)$ and $h(s)$ approximate $f(s)$ for various seeds s for *those* instances, where $f(s)$ can be measured directly. The same g and h can then be used for *unsupervised* learning on all *new* task instances.

3.1 Selecting Candidate Seeds

The first step in strapping a classifier is to select a set S of seeds to try. For strapping to work, it is crucial that this set contain a fertile seed. How can this be arranged? Different strategies are appropriate for different problems and bootstrapping methods.

- Sometimes a simple heuristic $g(s)$ can help identify plausibly fertile seeds, as in the pseudocode above. In strapping the Yarowsky algorithm, we hope to find seeds $s = (x, y)$ such that x and y are strongly associated with different senses of the ambiguous target word. We choose $s = (x, y)$ such that x and y were never observed in the same sentence, but each of x and y has high pointwise mutual information with the ambiguous target word and appeared with it at least 5 times.
- If the space of possible seeds is small, it may be possible to try many or all of them. In grammar induction, for example, perhaps seeding with a few basic word order facts is enough. There are not so many basic word orders to try.

- Some methods have many fertile seeds—so many that a small random sample (perhaps filtered by $g(s)$) is likely to include at least one. We rely on this for the Yarowsky algorithm. If the target word is a true homonym, there exist many words x associated strongly with the first sense, and many words y associated strongly with the second sense. It is not difficult to stumble into a fertile seed $s = (x, y)$, just as it is not difficult for a human to think of one.[5]
- If fertile seeds are few and far between, one could abandon the use of a candidate set S selected by $g(s)$, and directly use general-purpose search methods to look for a seed whose predicted fertility $h(s)$ is high.

  For example, one could use genetic algorithms to breed a population of seeds with high $h(s)$. Or after evaluating several candidate seeds to obtain $h(s_1), h(s_2), \ldots h(s_k)$, one could perform a regression analysis that predicts $h(s)$ from superficial features of s, and use this regression function (a kind of $g(s)$ that is specific to the task instance) to pick s_{k+1}.

Strapping may be harder in cases like gender induction: it is hard to stumble into the kind of detailed seed used by Cucerzan and Yarowsky (2003). However, we suspect that fertile seeds exist that are much smaller than their lists of 50–60 words. While their large hand-crafted seed is sure to work, a handful of small seeds (each consisting of a *few* supposedly masculine and feminine words) might be likely to contain at least one that is fertile.[6] That would be sufficient, assuming we have a way to guess which seed in the handful is most fertile. That issue is at the core of strapping, and we now turn to it.

3.2 Clues for Evaluating Bootstrapped Classifiers

Once we have identified a candidate seed s and built the classifier C_s, we must evaluate whether C_s "looks like" the kind of classifier that tends to do well on our task.

This evaluation function $h(s)$ is task-specific. It may consider features of C_s, the growth trajectory of C_s, or the relation between C_s and other classifiers.

For concretness, we consider the Yarowsky method for word-sense disambiguation (WSD). How can we tell if a seed $s = (x, y)$ was fertile, without using even a small validation set to judge C_s? There are several types of *clues* to fertility, which may be combined into a meta-classifier that identifies fertile seeds.

Judge the result of classification with C_s: Even without a validation set, the result of running C_s on the training corpus can be validated in various ways, using independent plausibility criteria that were *not* considered by the bootstrapping learner.

- Is the classification reasonably balanced? (If virtually all examples of the target word are labeled with the same sense, then C_s has not found a sense distinction.)
- When a document contains multiple tokens of the target word, are all examples labeled with the same sense? This property tends to hold for correct classifiers (Gale et al., 1992a), at least for homonyms.
- True word senses usually correlate with document or passage topic. Thus, choose a measure of similarity between documents (e.g., the cosine measure in TF/IDF space). Does the target word tend to have the same sense in a document and in its nearby neighbors?
- True word senses may also improve performance on some task. Is the perplexity of a language model much reduced by knowing whether sense x or sense y (according to C_s) appeared in the current context? (This relates to the previous point.) Likewise, given a small bilingual text that has been automatically (and perhaps poorly) word-aligned, is it easier to predict how the target word will translate when we know its sense (according to C_s)?

Judge the internal structure of C_s: Does C_s look like a typical supervised decision list for word-sense disambiguation? For instance, does it contain many features with high log-likelihood ratios? (If a true sense distinction was discovered, we would expect *many* contextual features to correlate strongly with the predicted sense.)

Look at the process whereby C_s was learned: Does the bootstrapping run that starts from s look like a typical bootstrapping run from a fertile seed? For example, did it rapidly add many new examples with high confidence? Once new examples were classified, did their classifications remain stable rather than switching back and forth?

Judge the robustness of learning with seed s: Train several versions of C_s, as in ensemble methods (but unsupervised), by restricting each to a random subset of the data, or a subset of the available features. Do these versions tend to *agree* on how to classify the data? If not, seed s does not reliably find true (or even false) classes.

Judge the agreement of C_s with other classifiers: Are there several other classifiers $C_{s'}$ that agree strongly with C_s on examples that they both classify? If the sense

[5] Alignment methods in machine translation rely even more heavily on this property. While they begin with a small translation lexicon, they are sufficiently robust to the choice of this initial seed (lexicon) that it suffices to construct a single seed by crude automatic means (Brown et al., 1990; Melamed, 1997). Human supervision (or strapping) is unnecessary.

[6] This is particularly likely if one favors function words (in particular determiners and pronouns), which are strong indicators of gender. Cucerzan and Yarowsky used only content words because they could be extracted from bilingual dictionaries.

distinction is real, then many different seeds should be able to find it.

3.3 Training the Evaluation Function $h(s)$

Many of the above clues are necessary but not sufficient. For example, a learned classification may be robust without being a sense distinction. We therefore define $h(s)$ from a combination of several clues.

In general, $h(s)$ is a classifier or regression function that attempts to distinguish fertile from infertile seeds, given the clues. As mentioned earlier, we train its free parameters (e.g., coefficients for linear regression) on a few *supervised* instances of the task. These supervised instances allow us to measure the fertility $f(s)$ of various seeds, and thus to model the behavior of fertile versus infertile seeds. The presumption is that these behavior patterns will generalize to new seeds.

3.4 Training $h(s)$ on Artificial Data

Optionally, to avoid the need for any human annotation at all, the supervised task instances used to train $h(s)$ may be *artificial* instances, whose correct classifications are known without annotation.

In the case of word-sense disambiguation, one can automatically construct ambiguous *pseudowords* (Gale et al., 1992c; Schütze, 1998) by replacing all occurences of two words or phrases with their conflation. For example, *banana* and *wine* are replaced everywhere by *banana-wine*. The original, unconflated text serves as a supervised answer key for the artificial task of disambiguating *banana-wine*.

Traditionally, pseudowords are used as cheap test data to evaluate a disambiguation system. Our idea is to use them as cheap development data to tune a system. In our case, they tune a few free parameters of $h(s)$, which says what a good classifier for this task looks like. Pseudowords should be plausible instances of the task (Gaustad, 2001; Nakov and Hearst, 2003): so it is deliberate that *banana* and *wine* share syntactic and semantic features, as senses of real ambiguous words often do.

Cheap "pseudo-supervised" data are also available in some other strapping settings. For grammar induction, one could construct an artificial probabilistic grammar at random, and generate text from it. The task of recovering the grammar from the text then has a known answer.

4 Experiments

4.1 Unsupervised Training/Test Data

Our experiments focused on the original Yarowsky algorithm. We attempted to strap word-sense classifiers, using English data only, for English words whose French translations are ambiguous. This has obvious benefits for training an English-to-French MT system: separate parameters can be learned for the two senses of *drug*.[7]

Gale et al. (1992b) identified six such words in the Canadian Hansards, a parallel sentence-aligned corpus of parliamentary debate in English and French: *drug, duty, land, language, position, sentence*. We extracted all examples of each word from the 14-million-word English portion of the Hansards.[8] Note that this is considerably smaller than Yarowsky's (1995) corpus of 460 million words, so bootstrapping will not perform as well, and may be more sensitive to the choice of seed.

Because we are doing unsupervised learning, we both trained and tested these 6 words on the English Hansards. We used the French portion of the Hansards only to create a gold standard for evaluating our results.[9] If an English sentence containing *drug* is paired with a French sentence that contains exactly one of *médicament* or *drogue*, we take that as an infallible indicator of its sense.

4.2 Comparing Classifiers

Suppose binary classifier 1 assigns class "+" to a of n examples; binary classifier 2 assigns class "+" to b of the same n examples. Let e be the number of examples where the classifiers agree (both "+" or both "–").

An unsupervised classifier's polarity is arbitrary: classifier 1's "+" may correspond to classifier 2's "–". So we define the *overlap* as $E = \max(e, n - e)$, to reflect the best polarity.

To evaluate a learned classifier, we measure its overlap with the true classification. The statistical significance is the probability that this level of overlap would be reached by chance under independent classifications given the values a, b, n:

$$p = \sum_{\substack{\max(a+b-n,0) \le c \le \lfloor (a+b-E)/2 \rfloor \\ \text{or} \\ \lceil (a+b-(n-E))/2 \rceil \le c \le \min(a,b)}} \binom{a}{c}\binom{n-a}{b-c} \Big/ \binom{n}{b}$$

Also, we can measure the *agreement* between any two learned classifiers as $-(\log p)/n$. Note that a classifier that strongly favors one sense will have low agreement with other classifiers.

[7] To hedge against the possibility of misclassification, one could interpolate with non-sense-specific parameters.

[8] We are not certain that our version of the Hansards is identical to that in (Gale et al., 1992b).

[9] By contrast, Gale et al. (1992b) used the French portion as a source of training supervision. By contrast, we will assume that we do *not* have a large bilingual text such as the Hansards. We train only on the English portion of the Hansards, ignoring the French. This mimics the situation where we must construct an MT system with very little bilingual text. By first discovering word senses in unsupervised monolingual data (for either language), we can avoid incorrectly mixing up two senses of *drug* in our translation model.

4.3 Generating Candidate Seeds (via $g(s)$)

For each target word t, we chose candidate seeds $s = (x, y)$ with a high score $g(s)$, where $g(s) = \text{MI}(t, x) + \text{MI}(t, y)$, provided that $c(x, y) = 0$ and $c(t, x) \geq 5$ and $c(t, y) \geq 5$ and $1/9 < c(t, x)/c(t, y) < 9$.[10]

The set S of 200 seeds for t was constructed by repeatedly adding the top-scoring unused seed to S, except that to increase the variety of words, we disallowed a seed $s = (x, y)$ if x or y already appeared 60 times in S.

4.4 Hand-Picked Seeds

To compare, we chose two seeds by hand for each t.

The *casually* hand-picked seed was chosen by intuition from the list of 200 automatically generated seeds. This took about 2 minutes (per seed).

The *carefully* hand-picked seed was not limited to this list, and took up to 10 minutes to choose, in a data-guided fashion. We first looked at some supervised example sentences to understand the desired translational sense distinction, and then for each sense chose the highest-MI word that both met some stringent subjective criteria and appeared to retrieve an appropriate initial set of examples.

4.5 The Bootstrapping Classifier

Our approximate replication of Yarowsky's algorithm used only a small set of features:

- Original and lemmatized form of the word immediately preceding the target word t.
- Original and lemmatized form of the word immediately following t.
- Original and lemmatized form of the *content* words that appear in the same sentence as t.

We used the seed to provisionally classify any token of the target word that appeared in a sentence with exactly one of the two seed words. This formed our initial "training set" of disambiguated tokens. At each iteration of the algorithm, we trained a decision list on the current training set. We then used the decision list to reclassify all k tokens in the current training set, and also to augment the training set by classifying the *additional* $\max(50, k/10)$ tokens on which the decision list was most confident.[11]

[10] $c(x, y)$ counts the sentences containing both x and y. $\text{MI}(t, x) = \log c(t, x)c()/c(t)c(x)$ is pointwise mutual information.

[11] Such a token has some feature with high log-likelihood ratio, i.e., it strongly indicates one of the senses in the current training set. We smoothed using the method of (Yarowsky, 1996): when a feature has been observed with only one sense, its log-likelihood ratio is estimated as a linear function of the number of occurrences of the seen sense. Function words are smoothed with a different linear coefficient than content words, in order to discount their importance. We borrowed the actual coefficients from (Yarowsky, 1996), though we could have learned them.

4.6 Development Data (for tuning $h(s)$)

Before turning to the unsupervised Hansards, we tuned our fertility estimator $h(s)$ to identify good seeds on development data—i.e., on other, supervised task instances.

In the *supervised* condition, we used just 2 additional task instances, *plant* and *tank*, each with 4000 hand-annotated instances drawn from a large balanced corpus (Yarowsky, 1995).

In the *pseudo-supervised* condition, we used *no hand-annotated data*, instead constructing 10 artificial supervised task instances (section 3.4) from the English portion of the Hansards. To facilitate cross-instance learning, we tried to construct these pseudowords to behave something like our ambiguous test words.[12] Given a test word t, we randomly selected a seed (x, y) from its candidate list (section 4.3), excluding any that contained function words.[13] Our basic idea was to conflate x and y into a pseudoword x-y. However, to get a pseudoword with only two senses, we tried to focus on the particular senses of x and y that were selected by t. We constructed about 500 pseudoword tokens by using only x and y tokens that appeared in sentences that contained t, or in sentences resembling those under a TF-IDF measure. We repeated this process twice per test word to obtain 12 pseudowords. We then discarded the 2 pseudowords for which no seed beat baseline performance, reasoning that they were ill-chosen and unlike real ambiguous words.[14]

4.7 Clues to Fertility

For each seed s for each development or test target word, we measured a few clues $h_1(s), h_2(s) \ldots h_6(s)$ that we hoped might correlate with fertility. (In future work, we plan to investigate more clues inspired by section 3.2.)

- The *agreeability* of C_s with (some of) the other 199 classifiers:

$$\left(\frac{1}{199} \sum_{s' \neq s} \text{agr}(C_s, C_{s'})^{\gamma} \right)^{1/\gamma}$$

The agreement $\text{agr}(C_s, C_{s'})$ was defined in section 4.2. We tried 4 values for γ (namely 1, 2, 5, 10), each resulting in a different feature.

[12] We used collocates of t. Perhaps better yet would be words that are distributionally similar to t (appear in same contexts). Such words tend to be syntactically and semantically like t.

[13] For an unknown language or domain, a lexicon of function words could be constructed automatically (Katz, 1996).

[14] Thus we discarded *alcohol-trafficking* and *addicts-alcohol*; note that these were indeed ill-chosen (difficult) since both words unluckily corresponded to the *same* sense of *drug*. This left us with *bound-constituents, customs-pray, claims-value, claims-veterans, culture-unparliamentary, english-learn, competitive-party, financial-party, death-quote, death-page.*

- The *robustness* of the seed, defined by the agreement of C_s with 10 variant classifiers $C_s^{(k)}$ that were trained with the same seed but under different conditions:

$$\frac{1}{10}\sum_{k=1}^{10} \text{agr}(C_s, C_s^{(k)})$$

  We simply trained each classifier $C_s^{(k)}$ on a random subset of the n test examples, chosen by sampling n times with replacement.[15]

- The *confidence* of C_s on its own training data: its average confidence over the n training tokens, minus the *classifier skew*.

  The decision list's confidence on a token is the log-likelihood ratio of the single feature used to classify that token. It has the form $|\log(c/d)|$ (perhaps smoothed) and was previously used to select data while bootstrapping C_s. Subtracting the skew, $|\log(a/(n-a))|$,[16] gives a measurement ≥ 0. It corrects for confidence that arises from the classifier's overall bias, leaving only the added value of the relevant contextual feature.

4.8 Tuning $h(s)$ and Strapping New Classifiers

For each of the 2 words or 10 pseudowords t in our development set (see section 4.6), we ranked its 200 seeds s by their true fertility $f(s)$. We then ran support vector regression[17] to learn a single linear function, $h(s) = \vec{w} \cdot$ (clue vector for C_s), that predicts the fertilities of all $2 \cdot 200$ or $10 \cdot 200$ seeds.[18]

Then, for each of our 6 Hansards test instances (section 4.1), we used $h(s)$ to pick the *top*-ranked of 200 seeds.[19] It took about 3 hours total to strap classifiers for all 6 instances, using about 40 machines and unoptimized Perl code on the 14-million-word Hansards. For each of the 6 instances, this involved selecting 200 candidate seeds, bootstrapping 11 classifiers $C_s, C_s^{(1)}, \ldots C_s^{(10)}$ from each seed, and choosing a particular C_s to return.

4.9 Results

Our results are in Table 1. On both development and test instances of the task, $g(s)$ proposed seeds with a good range of fertilities. The correlation of predicted with actual fertility on test data averaged an outstanding 85%.

Despite having no knowledge of the desired senses, strapping significantly beat human selection in *all 24* of the possible comparisons between a hand-picked seed (casual or careful) and a strapped seed (chosen by an $h(s)$ tuned on supervised or pseudo-supervised instances).

The $h(s)$ tuned on annotated *plant/tank* actually chose the *very best* of the 200 seeds in 4 of the 6 instances. The $h(s)$ tuned on artificial pseudowords did nearly as well, in 2 of 6 instances identifying the very best seed, and in 5 of 6 instances ranking it among its top 3 choices.

We conclude that our unsupervised clues to fertility actually work. Furthermore, combining clues via regression was wise, as it tended to work better than any single clue. Somewhat better regression weights for the WSD task were learned from 2 out-of-domain hand-annotated words than from 10 in-domain artificial pseudowords.

5 Open Questions

The work reported here raises many interesting questions for future research.

In the WSD task, we have only considered word types with two unrelated senses (homonyms). A more general problem is to determine when a word type is ambiguous at all, and if so, how many coarse-grained or fine-grained senses it has. Strapping seems naturally suited to this problem, since it aims to discover when a sense distinction grown from some seed is a *true* sense distinction.

Then we would like to know how well strapping generalizes to additional bootstrapping scenarios. Our WSD strapping experiments were successful using only a subset of the techniques proposed in section 3. Generalizing to other tasks may require other techniques for selecting and evaluating candidate seeds, and perhaps combining the resulting classifiers.

An interesting question is whether strapping can be used in an active learning context. Active learning is a kind of bootstrapping method that periodically requires new seeds: it turns to the user whenever it gets confused. Perhaps some of these seeds can be guessed nondeterministically and the guesses evaluated automatically, with or without user confirmation.

Finally, there may be theoretical guarantees about strapping when something is known about the data. When $h(s)$ is trained to estimate $f(s)$ well on some supervised instances, there may be guarantees about how strapping will perform on unsupervised instances drawn

[15] We eliminated duplicates, perhaps unfortunately.

[16] As before, a and $n-a$ are the numbers of tokens that C_s classifies as "+" and "–" respectively. Thus the skew is the log-likelihood ratio of the decision list's "baseline" feature.

[17] We used cross-validation among the 10 development pseudowords to choose the options to SVM^{light} (Joachims, 1999): a linear kernel, a regularization parameter of 0.3, and a dependent variable of $10^{f(s)} \in [1, 10]$ rather than $f(s) \in [0, 1]$, which placed somewhat more emphasis on modeling the better seeds. Our development objective function was the average over the 10 pseudowords of the Spearman rank-order correlation between $h(s)$ and $f(s)$.

[18] We augmented the clue vector with binary clues of the form $t =$ *plant*, $t =$ *tank*, etc. The regression weight of such a clue is a learned bias term that models the inherent difficulty of the task instance t (which varies greatly by t). This allows the other regression features to focus on the quality of the seed *given* t.

[19] We do not have a clue $t = \ldots$ for this test instance. The resulting lack of a bias term may subtract a constant from the predicted fertilities—but that does not affect the ranking of seeds.

	drug		duty	land		language	position	sentence	
baseline / # examples	51.2 / 371		70.1 / 633	76.6 / 1379		87.5 / 1012	81.7 / 2949	50.8 / 501	
worst seed (of 200)	50.1 (200)	traffickers trafficking	50.0 (200)	50.1 (200)	claims farming	50.3 (200)	56.1 (200)	50.1 (200)	length life
casually selected (from 200)	56.5 (87)	food trafficking	73.4* (40)	76.2 (24)	farm veterans	86.4 (76)	81.7 (41)	80.6* (40)	page prison
carefully constructed	62.1* (75)	alcohol costs	82.1* (8.5)	76.6 (20)	farm strong	87.9 (25.5)	81.4 (56.5)	86.8* (27)	death quote
best/oracle seed (of 200)	76.1*† (1)	alcohol medical	86.2*† (1)	81.3*† (1)	acres courts	90.9*† (1)	88.3*† (1)	89.9*† (1)	reads served
strapping (unsupervised)									
most agreeable seed ($\gamma = 1$)	72.6*† (5)	abuse information	64.7 (47)	67.5 (36)	claims production	86.4 (79)	82.4 (36)	88.7*† (10)	life quote
most robust seed	**76.1***† (1)	alcohol medical	**86.2***† (1)	71.7 (29)	claims price	85.6 (93)	82.7 (21)	88.8*† (9)	commuted next
most confident seed	66.9* (32)	trafficking used	72.1* (42)	77.9*† (3)	claims courts	89.8*† (10)	84.4*† (8)	**89.9***† (1)	reads served
$h(s)$-picked (plant/tank)	**76.1***† (1)	alcohol medical	**86.2***† (1)	**81.3***† (1)	acres courts	90.3*† (7)	84.5*† (7)	**89.9***† (1)	reads served
$h(s)$-picked (10 pseudowd)	70.4*† (10)	alcohol found	**86.2***† (1)	78.9*† (2)	children farm	89.7*† (17)	83.7*† (16)	**89.9***† (1)	reads served
$h(s)$-picked, 2nd place	69.1* (13)	alcohol related	85.7*† (2)	77.8*† (4)	aboriginal acres	**90.9***† (1)	82.8 (19)	89.0*† (7)	prison quote
$h(s)$-picked, 3rd place	**76.1***† (1)	alcohol medical	84.2* (4)	77.1*† (5)	acres cities	87.5 (28)	**88.3***† (1)	88.6*† (15)	life reads
$h(s)$ rank of oracle seed	3		1	14		2	3	1	
Spearman rank-order corr.	0.863		0.905	0.718		0.825	0.842	0.937	

Table 1: [See section 4.9 for highlights.] Accuracy (as percentage) and rank (in parentheses) of bootstrapped classifiers for variously chosen seeds, some of which are shown. * denotes statistically significant agreement with the truth (section 4.2, $p < 0.01$). † denotes a seed having significantly better agreement with the truth than does the better of the hand-picked seeds (McNemar's test, $p < 0.03$). In each column, the best performance for an automatic or manual seed appears in **boldface**. The "most . . . " lines use no tuning, the "plant/tank" line tunes $h(s)$ on 2 supervised instances, and the subsequent lines tune $h(s)$ on 10 pseudoword instances. The last line gives the Spearman rank-order correlation between seeds' predicted fertilities $h(s)$ and their actual fertilities $f(s)$.

from the same source (cross-instance learning). Even in the fully unsupervised case, it may be possible to prove that if the data were generated from a particular kind of process (e.g., a Gaussian mixture), then a certain strapping algorithm can recover the hidden variables.

6 Conclusions

In this paper, we showed that it is sometimes possible—indeed, preferable—to eliminate the initial bit of supervision in "bootstrapping" algorithms such as the Yarowsky (1995) algorithm for word sense disambiguation. Our "strapping" approach tries many candidate seeds as starting points and evaluates them automatically. The evaluation function can be tuned if desired on other task instances, perhaps artificially constructed ones. It can then be used wherever human guidance is impractical.

We applied the method to unsupervised disambiguation of English words in the Canadian Hansards, as if for English-French translation. Our results (see section 4.9 for several highlights) show that our automatic "strapped" classifiers consistently outperform the classifiers bootstrapped from manually, knowledgeably chosen seeds.

References

R. K. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers Presented at the 97th meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June.

A. Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, July.

P. F. Brown, J. Cook, S.A. Della Pietra, V.G. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A statistical approach to machine translation. *CL*, 16(2).

N. Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP/VLC*.

S. Cucerzan and D. Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proc. of EMNLP/VLC*.

S. Cucerzan and D. Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proc. of HLT/NAACL*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–38.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992a. One sense per discourse. In *Proc. of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992b. Using bilingual materials to develop word sense disambiguation methods. In *Proc. of the 4th International Conf. on Theoretical and Methodological Issues in Machine Translation*.

W. A. Gale, K. W. Church, and D. Yarowsky. 1992c. Work on statistical methods for word sense disambiguation. In *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

T. Gaustad. 2001. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *Proc. of ACL-EACL*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*. MIT Press.

S. M. Katz. 1996. Distribution of context words and phrases in text and language modelling. *NLE*, 2(1):15–59.

I. Dan Melamed. 1997. A word-to-word model of translational equivalence. In *Proc. of ACL/EACL*, page 490.

P. Nakov and M. Hearst. 2003. Category-based pseudowords. In *HLT-NAACL'03*, pages 67–69, Edmonton, Canada.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proc. of CoNLL*, pages 25–32, May–June.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 23.

S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. 2002. The latent maximum entropy principle. In *Proc. of ISIT*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

D. Yarowsky. 1996. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. Ph.D. thesis, U. of Penn.