NADA: A Robust System for Non-Referential Pronoun Detection

Shane Bergsma and David Yarowsky

Dept. of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
sbergsma@jhu.edu, yarowsky@cs.jhu.edu

Abstract. We present NADA: the Non-Anaphoric Detection Algorithm. NADA is a novel, publicly-available program that accurately distinguishes between the referential and non-referential pronoun *it* in raw English text. Like recent state-of-the-art approaches, NADA uses very large-scale web N-gram features, but NADA makes these features practical by compressing the N-gram counts so they can fit into computer memory. NADA therefore operates as a fast, stand-alone system. NADA also improves over previous web-scale systems by considering the entire sentence, rather than narrow context windows, via long-distance lexical features. NADA very substantially outperforms other state-of-the-art systems in non-referential detection accuracy.

1 Introduction

Virtually all anaphora resolution systems search for the referents of an anaphor among the preceding noun phrases (NPs). However, not every expression that looks like an anaphor actually has an NP antecedent. In English, the pronoun *it* can refer both to NPs and also to more abstract entities, such as discourse segments [6, 31]. *It* can also be used as a syntactic placeholder, as in −ref below:

+**ref** *It* is able to maintain a stable price.
−**ref** *It* is important to maintain a stable price.

The pronoun *It* in +ref is a *nominal* pronoun; it refers back to some previously-mentioned entity. *It* in −ref is a *pleonastic* or *expletive* pronoun, a dummy subject without an antecedent. For the purposes of information extraction or retrieval, we would like to know *who* is able to maintain a stable price in +ref, and might apply a pronoun resolution system to find the antecedent. On the other hand, we need to know that there is no antecedent for *It* in −ref; it would be wasteful and potentially harmful to try to find one.

In this paper, we investigate detectors that distinguish nominal *it* from both pleonastic *it* and from *it* referring to abstract entities, such as discourse segments. We call the latter cases *non-referential it*, following convention. This is the pragmatic division since, as mentioned, virtually all anaphora resolution systems only attempt to resolve pronouns with NP antecedents (but see [6]).

The referential/non-referential ambiguity is quite common in text. For example, there are over 7000 instances of *it* in the (one-million-token) Wall Street Journal portion of the Penn Treebank, of which 26% are non-referential (§4). Unfortunately, non-referential detection is not performed terribly well by today's coreference resolution systems. In experiments on a portion of the Treebank, the state-of-the-art Charniak and Elsner pronoun resolution system [7] achieves 73% accuracy in deciding if an *it* is non-referential, only marginally above the majority-class baseline. Thus there is need for attention to this problem.

Non-referential pronoun detection is structurally similar to other lexical disambiguation problems like word-sense disambiguation and spelling correction [2]; and, as in these tasks, supervised machine learning approaches have been adopted. We aimed to build a machine-learned non-referential detector that incorporates many of the features used in prior machine learning approaches to this problem [12, 4, 25]. However, the ±ref examples show why non-referential detection is a difficult disambiguation task: the decision depends on specific lexical items (i.e. *able* vs. *important*) rather than parts of speech. Hence, the information needed for robust detection cannot be derived from limited hand-annotated data. We thus look to leverage unlabeled data to improve non-referential detection, in particularly building on the use of web-scale N-gram data by Bergsma et al. [1].

We present a fast, freely-available non-referential detector for English that very substantially outperforms other state-of-the-art approaches. We call our system NADA (Non-Anaphoric Detection Algorithm). The NADA system and source code are available for download online through Google Code at:

`http://code.google.com/p/nada-nonref-pronoun-detector/`

To maximize adoption of NADA, we ensured: (1) it does not require parsed/tagged input, rather raw (tokenized) text, (2) it is fast, classifying *it* at speeds up to 20K sentences/sec., and (3) it works well out-of-domain.

2 Related Work

While the issue of non-referential pronouns has long been acknowledged [18, 17], the problem has been side-stepped in various ways in past pronoun resolution research. Ge et al. [14] report that non-referentials "are excluded from computing the precision." Kehler et al. [19] only consider pronouns that were "ACE markables," so that "certain problems... such as non-referential pronouns and pronouns that refer to eventualities, did not have to be dealt with."[1] Yang et al. [33] only evaluate on "pronouns with non-empty candidate sets." Systems

[1] ACE (Automatic Content Extraction) is a NIST program that conducted a series of evaluations on information extraction systems. These evaluations resulted in the availability of new coreference data for researchers, but, unfortunately, this data only includes coreference annotations for a subset of entity types, including people, organizations, and facilities. Furthermore, assuming the availability of the ACE markables "unrealistically simplifies the coreference resolution task" [30].

that do detect non-referentials as part of a fully-automatic pronoun resolution system include [24, 8, 7].

Early attempts to handle non-referentials were rule-based, and focused on English *it* [27, 21]. While these approaches performed well in their original domains, later work has found these systems to perform poorly on new data [12, 4, 1]. Recent work has focused on training non-referential classifiers using machine learning [12, 4, 25, 1]. Instances of *it* are classified on the basis of various features, with feature weights learned from a labeled training set. A recent departure from this approach is by Charniak and Elsner [7] who perform non-referential *it* detection jointly with pronoun resolution via inference in an unsupervised generative model. Recent work has also moved to identify non-referential pronouns in other languages, including French [10], Arabic [16] and Spanish [29].

A related line of work aims to identify all noun phrases that have an antecedent in text, but these systems typically classify all pronouns as referential and ignore non-referential *it* [26, 11].

Bergsma et al. [1] detect non-referential pronouns using counts from web-scale N-gram data (we describe this approach in detail in §3.2). NADA improves on the Bergsma et al. work in several ways. First, we integrate their web count features with lexical (indicator) features that are inspired by previous non-referential classifiers. Secondly, because the original Bergsma et al. approach relies on the very large Google N-gram corpus, their approach cannot easily be made into a stand-alone, publicly-available system. For example, Miltsakaki [23] used the Bergsma et al. approach to detect non-referential *it* in the *Antelogue* pronoun resolution system, but usage of *Antelogue* therefore required separate access to the huge Google corpus. Since our objective is to develop a publicly-available, stand-alone system that works on a variety of texts, we adapt the Bergsma et al. approach to ensure the needed counts can fit into computer memory.

To our knowledge, up until now there have been no publicly-available, state-of-the-art non-referential pronoun detectors. Potential users of non-referential technology usually re-implement systems when needed, often adopting the simpler rule-based approaches as baselines. The Charniak and Elsner [7] system is publicly-available (and we compare to it), but requires (expensive) syntactic parses of entire documents as input. Since NADA is simpler, faster, and more accurate, we expect it to be adopted both as a component of larger coreference resolution systems and as a competitive comparison system for future non-referential detection research.

3 Supervised Non-referential Detection

We build on previous supervised approaches to detecting non-referential pronouns [12, 4, 25, 1]. For each instance of the pronoun *it*, we create a feature vector, $\bar{x}$, to encode information about the pronoun's context. Our feature vector consists of two types of features: (1) *Lexical features*: binary-valued features that indicate the presence or absence of a particular string at a given position in the input (§3.1) and (2) *Web count features*: real-valued features that give the log-

count of relevant N-grams derived from the input, with the counts taken from an auxiliary web-scale N-gram corpus (§3.2). The feature vector is given as input to a classifier and the output is a decision, y, as to whether the instance is referential or not. We take 'y=1' to denote the non-referential class and 'y=0' to denote the referential class.

We use a (regularized) logistic regression model as our supervised classifier. Logistic regression has been shown to perform well on a range of NLP tasks. In binary logistic regression, the features are weighted with a set of learned weight parameters, $\bar{w}$. The probability of a positive class is the logistic function:

$$\Pr(y = 1) = \frac{e^{\bar{w}\cdot\bar{x}}}{1 + e^{\bar{w}\cdot\bar{x}}}$$

We predict non-referential if $\Pr(y = 1) > 0.5$ (equivalently, $\bar{w} \cdot \bar{x} > 0$), otherwise we predict referential.

We assume N labeled training examples $\{(y^1, \bar{x}^1), ..., (y^N, \bar{x}^N)\}$ are available to train the classifier. The weight parameters $\bar{w}$ are set at training time in order to maximize performance on the training corpus.

While the supervised learning paradigm has been very successful in NLP, there are some important caveats. First of all, adding new features only improves classification accuracy if there's sufficient training data from which to learn good feature weights; the more features one uses, the more training data one needs. Secondly, many supervised systems perform poorly when used outside of their training domain. However, recent work has shown that the use of web N-gram features in supervised classifiers can alleviate both problems: systems combining lexical and count features tend to do better than using lexical features alone, especially with less training data and when operating on new domains [3].

3.1 Lexical Features

The optimum amount of context to encode in a feature representation remains an open question for this task. In [12], the "vectors convey information obtained from the paragraph in which the instance appears," while Bergsma et al. [1] use "context patterns that together span from four-[tokens]-to-the-left to four-[tokens]-to-the-right of the pronoun." We therefore tested how humans perform with varying amounts of context, comparing their decisions to the labels in the BBN Pronoun Coreference Corpus [32]. Given only four tokens on either side of the *it* pronoun, our subject achieved 85% accuracy on 200 instances. When subsequently given the entire sentence, the subject achieved 95%, which is broadly similar to inter-annotator agreement given the full discourse [1]. We therefore extract our lexical features from the entire sentence.

Consider the referential example "*The EU team says* **it** *was able to address its concerns in full.*" We first normalize the text by converting all digits to '0' and replacing multi-character capitalized words with special named-entity tokens (e.g. *EU* → *NE*). Lexical features then encode the following specific attributes of the pronoun's normalized context, via binary indicator features:

1. All 3-grams to 5-grams that span the confusable pronoun, e.g. *team-says-***it**, *says-***it**-*was*, **it**-*was-able*, *NE-team-says-***it**, etc.
2. Tokens conjoined with their positions, from two tokens before, to 5 tokens after the pronoun, e.g. $team_{-2}$, $says_{-1}$, was_{+1}, ..., its_{+5}.
3. Any token within the 20 tokens on the right, e.g. was_{right}, $able_{right}$, to_{right}, $address_{right}$, its_{right}, $concerns_{right}$, in_{right}, $full_{right}$.
4. Any token within 10 tokens on the left that is on the list {*that, this, and, said, says,* NE*, it, It, its, itself*}, e.g. NE_{left}, $says_{left}$.

These specific features were developed over the course of extensive development experiments on BBN data. We aimed to encapsulate many of the features used in prior machine learning approaches to this problem [12, 4, 25], but omitted those that did not prove effective in development experiments.[2] Given sufficient training data, such lexical features provide excellent discriminators for this task. For example, the system learns the presence of other forms of the third-person-neutral pronoun (e.g. *its/itself*) are associated with referential *it*: the *it*-to-classify is likely part of this coreference chain. The presence of preceding *NE* tags also indicates referential; these entities are often the antecedents of referential *it*. Prepositions immediately preceding the pronoun are also indicative of referential, as noted by Paice and Husk [27], while following complementizers such as *that* or *to* are indicative of non-referential, even when they occur many tokens after the pronoun.

3.2 Web Count Features

We also use features derived from web-scale unlabeled data following Bergsma et al. [1]. The Bergsma et al. approach first converts the context around *it* into patterns, e.g. "*it is able to*" → "_ *is able to*." Then, the Google N-gram data [5] is queried to determine which words fill the patterns. For referential cases, fillers like "**he** is able to" or "**China** is able to" are common, while for non-referentials (e.g., "_ *is important to*"), the word *it* is usually the most common filler. Rather than using a single pattern, Bergsma et al. gather fillers for all 4-and-5-gram patterns spanning the *it* token. The patterns are generalized using both stemming and various rules for irregular verbs and common contractions. Features are created with the counts of different filler classes (e.g., count(*word-it*)-in-pattern, count(*they/them*), count(*all-pronouns*), etc.). Each count feature is associated with a feature dimension (i.e., a particular element of the $\bar{x}$ vector) depending on the count's filler class, its pattern length, and the position of the filler slot in the pattern. Since each feature dimension has a corresponding weight (i.e., the corresponding element in the $\bar{w}$ vector), counts are ultimately weighted depending on their filler-class, pattern-length, and filler-position.

[2] Notably, we did not observe a performance improvement using part-of-speech tags, and thus only use lexical features in the final system. One benefit is that our system can operate on raw sentences; it thus runs as a convenient, stand-alone program.

Sequence of Steps in Compression	Resulting Size
1. All N-grams in the Google N-gram corpus (unzipped)	93 GB
2. Extract N-grams of length-4 only	33 GB
3. Extract N-grams containing *it*, *they*, *them* only	500 MB
4. Lower-case, truncate tokens to four characters, replace special tokens (e.g. named entities, pronouns, digits) with symbols, etc.	189 MB
5. Encode tokens and values, store only changes from previous line	44 MB
6. `gzip` resulting file	33 MB

Table 1: Steps taken to compress the huge Google N-gram corpus (33 GB just for 4-grams) by several orders of magnitude to a small file that can be quickly loaded into NADA's memory (44 MB without zipping).

Since these counts were shown to be effective, we want to use them in our own system, but the Google N-gram corpus is too large to include directly in a stand-alone system. We therefore took steps to compress the counts so they could be quickly loaded and accessed in a computer program's working memory.

Recent work has proposed ways to compress the Google N-gram corpus to fit it into memory for language modeling tasks [15, 28]. We could have potentially used these approaches to retrieve counts for our task, however, there are properties of our problem that will allow us to use significantly less memory than, e.g., the compression to 10 GB achieved in the recent Pauls and Klein paper [28]. Nevertheless, some of our lossless compression steps below (token/value encoding, trie-like compression) were inspired by recent work in this area.

Steps in Data Compression Table 1 shows the sequence of steps and their effect on the data size. The first steps are *lossy* compression steps in that they might result in loss of information. Note that, as in [1], we sum the counts of any N-grams that map to the same tokens as a result of our processing. Only using 4-grams (Step 2) and also truncating tokens to four characters (Step 4) were motivated by the analysis in [1], which showed only a small impact on accuracyr after these steps. The largest reduction is achieved by Step 3: only keeping N-grams that contain the words *it*, *they*, or *them* (regardless of capitalization); this filters 98% of the 4-grams. Unfortunately, as a result of this filtering we can now only make use of two of the five original filler classes used as features in [1]; however there is precedent for this in the SUMLM system in [2] and in Antelogue [23]. During development, we found that overall accuracy degrades by roughly 1% with these steps compared to the original Bergsma et al. system, which seems a reasonable price to pay for a practical system. Finally, Step 5 (and, of course, Step 6) is a *lossless* compression of the remaining N-gram data; we describe this step in more detail below. The following details are likely not relevant to all readers of this paper, however they might provide some insight for others trying to incorporate large-scale statistics into small-scale programs.

More Detailed Steps in Lossless Compression After Step 4, our N-grams are roughly 9.5 million lines that look like this:

```
make _ clea that 659127 0
make _ clea the 8500 152
make _ clea then 105 0
make _ clea ther 2224 0
make _ clea thes 427 0
make _ clea this 3194 42
make _ clea thor 118 0
make _ clea thos 89 0
make _ clea thou 392 0
make _ clea thro 827 0
```

Each line begins with the 4-gram (with the filler-position marked by a '_'), followed by the count of *it* and then the count of *they/them*.[3]

Our first step was to replace the tokens in each N-gram with a fixed numerical encoding (converting from a text file to a binary file). After the Step 4 processing, we found there were only 28047 unique tokens (types) in our data. These 28047 types can be stored in a separate list (the *token list*), and looked up when needed using a 15-bit index (since $2^{15} > 28047$). Furthermore, rather than storing the filler '_' in the same way as the other tokens, we instead add another flag bit to each encoded token to indicate whether or not that token is preceded-by-the-filler. In this way we replace the four original tokens with three 16-bit (2-byte) integers. Clearly, we now don't require any delimiters as in the original text file. Also, the 2-byte encoding aligns nicely with the 2-byte types native to C/C++, the language in which we coded NADA.

A slightly-more-complicated trick works to replace the values (i.e. the counts of the *it* and *they/them* fillers). To reduce the value space further, we only keep the two most significant digits in each value (e.g. 3194 42 is mapped to 3200 42).[4] After this quantization, we found there were only 65354 unique **pairs** of values, which could together be stored in a separate list (the *values list*) and looked up, when needed with a 16-bit integer index ($2^{16} > 65354$). So we can replace every pair of values with a single 2-byte integer.

For those keeping track, each N-gram needs six bytes for tokens, two bytes for values, and since there are 9.5 million N-grams in our data, we can store the whole shebang in $8 * 9.5 = 76$ MB. This is indeed how we store things in NADA's memory. We hash the 2-byte encoded-value as the hash value and look it up with the 6-byte encoded-tokens as the hash key. As we run NADA on text, we have separate, small and very fast hashes for the token list (TL) and values list (VL). We use the TL to encode the input text (as hash keys). We look up the encoded token keys in our big hash of encoded values, and then use the VL to decode the retrieved values back to pairs of counts, which of course provide the feature values in our classifier.

[3] Although it may not seem so based on this sample, this is indeed more compact than storing these two counts separately.

[4] Actually, this step is *lossy* but the resulting effect on NADA is negligible.

However, this is still wasteful, since one can observe from the sample above that most tokens in the N-grams don't change from line-to-line. So, to make the data even smaller in our final file *on disk* (which makes it faster to load from disk into memory) for each N-gram we only store the changes from the previous N-gram (having special flags to indicate when changes happen and which tokens have changed). This is broadly similar to the idea behind the *trie* data structure: essentially, we re-use the space for some tokens among many different N-grams.

This processing results in a 44 MB file (uncompressed) of pattern counts, which easily fits into computer memory. Thus our system ships and operates with all the data it needs.[5]

4 Experimental Details

We evaluate NADA in comprehensive experiments on news and other domains (§5). The NADA classifier is trained using L2-regularized logistic regression via the LIBLINEAR package [13]. We optimize the classifier's regularization parameter for development accuracy (the percentage of examples classified correctly). As a logistic regression classifier, NADA returns a probability of an instance being non-referential. To turn this probability into classification decisions, we threshold the probability at 0.5, predicting *non-referential* if the probability exceeds this threshold, referential otherwise. We report final accuracy, precision, recall, and F-score (F1) for classifications on held-out test data.

Unlabeled N-gram Data: All N-gram counts are taken from the web-scale Google N-gram Corpus [5]. The feature values are the logarithm of the counts. We add one to all counts for smoothing. If a count is unavailable, or if a context pattern spans beyond the sentence, we indicate so with binary indicator features.

Labeled Data: We use a much more extensive set of labeled data for our experiments than has been used in past research, namely, the BBN Pronoun Coreference Corpus [32]. The BBN corpus provides the antecedents of all *referential* pronouns in the Wall Street Journal (WSJ) portion of the Penn Treebank [22]. If a pronoun is labeled with an antecedent, we mark the instance as referential, otherwise as non-referential. There are 7195 instances of *it* in this corpus, of which 26% are non-referential. We train all classifiers on the first 3195 instances of *it*. We use another 1000 BBN examples as development data, and the last 3000 BBN examples as a final held-out test set.

We also evaluate on data from the publicly-available ItBank corpus [1]. We use the 709 instances from (separate) WSJ data as another in-domain dataset, called 'WSJ-2.' We also use another 1928 instances from ItBank as an out-of-domain set, called 'ItBank.' This set includes *it* instances from both Science News articles and articles from the Slate portion of the American National Corpus. Taking a system trained on the WSJ and testing it on this out-of-domain data is a good test of the robustness of our proposed detector.

[5] Note it would have been possible, but of dubious additional value here, to use variable-length encoding of the token and value indices to save even more space [9].

Features	BBN	WSJ-2	ItBank
Majority Class	72.5	74.9	67.7
Lexical	82.9	82.5	78.7
Web Counts	83.3	85.6	83.1
Lexical+Counts (final NADA system)	**86.0**	**86.2**	**85.1**

Table 2: Accuracy (%) on various datasets of majority-class (referential) and classifiers with different features. Web count features are better, especially out-of-domain, but **lexical and count-based features can be combined effectively on this task**.

System	Prec.	Rec.	F1	Acc.
Paice-Husk	56.9	40.8	47.5	75.2
Charniak-Elsner	51.2	**64.9**	57.3	73.4
NADA	**81.6**	63.4	**71.3**	**86.0**

Table 3: Classification performance of several comparison systems. NADA achieves superior classification results (%) on BBN, yielding a 47% error-rate-reduction relative to Charniak and Elsner [7].

5 Results

We first evaluate the speed of NADA, and find it to be very fast. It takes NADA just over two minutes to identify non-referential pronouns in 1.7 million English sentences in Europarl [20] (a speed of approximately 12 thousand sentences per second). On news text, NADA is even faster, tagging the 49 thousand sentences in the WSJ portion of the Treebank in 2.5 seconds, at a speed of roughly 20 thousand sentences per second. Of course, many of these sentences are passed unmodified if they do not contain an instance of *it*; on the 6511 WSJ sentences containing an *it*, NADA tags around 3900 sentences per second.

We then assess the value of the different features, alone and in combination, on the three test sets (Table 2). The system with *Lexical* features can be regarded as a reasonable approximation of a range of previous supervised approaches [12, 4, 25] (but trained and evaluated on more data), while the *Web Counts* system is a reasonable (but practical) approximation of the Bergsma et al. [1] approach. On the BBN data, the system with lexical features performs almost as well as the system with count features. On out-of-domain data, however, the lexical features perform 3-4% worse (confirming the trend observed in [3] on a range of other tasks). The classifier using both lexical and count features is best, performing between 85.1% and 86.2% on all data. This shows, for the first time, that the two dominant machine learning paradigms for this task (i.e., indicator features and count features) can be integrated effectively into a single system.

Our integrated approach is also superior to competing approaches in the literature. We reimplemented the rule-based approach of Paice and Husk [27]. We also tested the joint non-referential-detection/pronoun-resolution system of Charniak and Elsner [7]. This system is publicly available, so we compare directly to its output; we mark a pronoun as non-referential when the probability of being

non-referential exceeds a fixed threshold (set by the system designers). Unlike NADA, the Charniak and Elsner system requires full parses of entire documents, so we charitably provide it with the gold-standard Treebank parses for the BBN test data. Our proposed system does a very substantially better job than both these approaches (Table 3).

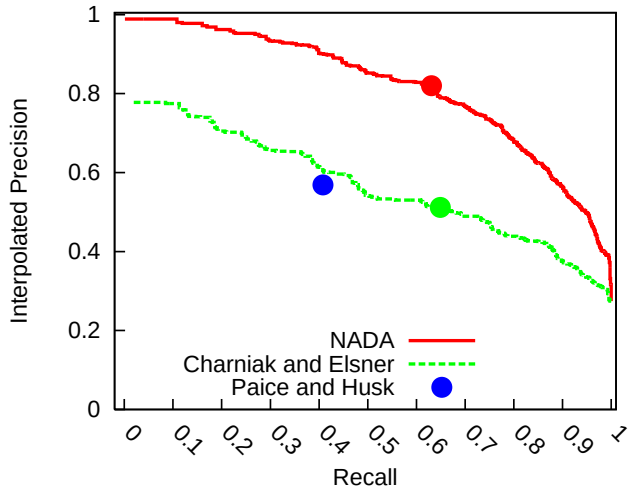

Fig. 1: Performance on BBN, showing superiority of NADA to Charniak-Elsner and Paice-Husk. NADA curve produced by thresholding output probability (• for P>0.5). Charniak-Elsner curve produced by thresholding probability of most probable antecedent (• where probability exceeds their (fixed) probability of being non-referential).

As mentioned, NADA returns a probability of an instance being non-referential, which we threshold at 0.5 to produce the above classification results. By lowering this threshold, we can increase the recall of NADA (at the expense of precision), while raising the threshold increases the precision (at the expense of recall). We can do the same for the Charniak and Elsner system, which also produces probabilistic output. Figure 1 provides a precision-recall curve for both detectors over a range of thresholds. For comparison, the plot also includes the fixed Paice and Husk performance. Note in particular that NADA obtains close to 100% precision on the 10-20% most-confident non-referentials. In fact, many of these instances are incorrectly 'resolved' by the Charniak-Elsner system, and this could be prevented by using our system as a reliable coreference preprocessor.

6 Conclusion

We presented NADA, a new system for detecting non-referential instances of the English pronoun *it*. NADA is a supervised system based on two kinds of

features: lexical and web-scale N-gram counts. We showed how to compress the N-gram counts to make NADA an efficient, stand-alone program. An extensive empirical evaluation showed that NADA outperforms all previous similar systems. Furthermore, NADA offers a number of important additional advantages: (1) it is publicly-available, (2) it operates on raw (tokenized) text, without requiring any special preprocessing, (3) it is very fast, and (4) it performs well on different domains. We hope many other groups also find NADA useful in their work.

References

1. Shane Bergsma, Dekang Lin, and Randy Goebel. Distributional identification of non-referential pronouns. In *Proc. ACL-08: HLT*, pages 10–18, 2008.
2. Shane Bergsma, Dekang Lin, and Randy Goebel. Web-scale N-gram models for lexical disambiguation. In *IJCAI*, pages 1507–1512, 2009.
3. Shane Bergsma, Emily Pitler, and Dekang Lin. Creating robust supervised classifiers via web-scale N-gram data. In *Proc. ACL*, pages 865–874, 2010.
4. Adriane Boyd, Whitney Gegg-Harrison, and Donna Byron. Identifying non-referential *it*: A machine learning approach incorporating linguistically motivated patterns. In *Proc. ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 40–47, 2005.
5. Thorsten Brants and Alex Franz. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13, 2006.
6. Donna Byron. Resolving pronominal reference to abstract entities. In *Proc. ACL*, pages 80–87, 2002.
7. Eugene Charniak and Micha Elsner. EM works for pronoun anaphora resolution. In *Proc. EACL*, pages 148–156, 2009.
8. Colin Cherry and Shane Bergsma. An Expectation Maximization approach to pronoun resolution. In *Proc. CoNLL*, pages 88–95, 2005.
9. Kenneth Church, Ted Hart, and Jianfeng Gao. Compressing trigram language models with Golomb coding. In *Proc. EMNLP-CoNLL*, pages 199–207, 2007.
10. Laurence Danlos. Automatic recognition of French expletive pronoun occurrences. In *Proc. IJCNLP*, pages 73–78, 2005.
11. Pascal Denis and Jason Baldridge. Joint determination of anaphoricity and coreference using integer programming. In *Proc. NAACL-HLT*, pages 236–243, 2007.
12. Richard Evans. Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 16(1):45–57, 2001.
13. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.
14. Niyu Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170, 1998.
15. David Guthrie and Mark Hepple. Storing the web in memory: Space efficient language models with constant time retrieval. In *Proc. EMNLP*, pages 262–272, 2010.
16. Souha Mezghani Hammami, Rahma Sallemi, and Lamia Hadrich Belguith. A bayesian classifier for the identification of non-referential pronouns in Arabic. In *Proc. INFOS, Special Track On Natural Language Processing and Knowledge Mining*, 2010.

17. Graeme Hirst. *Anaphora in Natural Language Understanding: A Survey*. Springer Verlag, 1981.
18. Jerry Hobbs. Resolving pronoun references. *Lingua*, 44(311):339–352, 1978.
19. Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proc. HLT-NAACL*, pages 289–296, 2004.
20. Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit X*, pages 79–86, 2005.
21. Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.
22. Mitchell P. Marcus, Beatrice Santorini, and Mary Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.
23. Eleni Miltsakaki. Antelogue: Pronoun resolution for text and dialogue. In *Proc. Coling 2010: Demonstrations*, pages 41–44, 2010.
24. Ruslan Mitkov, Richard Evans, and Constantin Orasan. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *Proc. CICLing*, pages 168–186, 2002.
25. Christoph Müller. Automatic detection of nonreferential *It* in spoken multi-party dialog. In *Proc. EACL*, pages 49–56, 2006.
26. Vincent Ng and Claire Cardie. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proc. COLING*, pages 730–736, 2002.
27. C. D. Paice and G. D. Husk. Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun "it". *Computer Speech and Language*, 2:109–132, 1987.
28. Adam Pauls and Dan Klein. Faster and smaller N-Gram language models. In *Proc. ACL*, pages 258–267, 2011.
29. Luz Rello, Pablo Suárez, and Ruslan Mitkov. A machine learning method for identifying impersonal constructions and zero pronouns in Spanish. In *Proc. Procesamiento del Lenguaje Natural*, pages 281–287, 2010.
30. Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. Conundrums in noun phrase coreference resolution: making sense of the state-of-the-art. In *Proc ACL-IJCNLP*, pages 656–664, 2009.
31. Bonnie Lynn Webber. Discourse deixis: reference to discourse segments. In *Proc. ACL*, pages 113–122, 1988.
32. Ralph Weischedel and Ada Brunstein. BBN pronoun coreference and entity type corpus. LDC2005T33, 2005.
33. Xiaofeng Yang, Jian Su, and Chew Lim Tan. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proc. ACL*, 2005.