Language ID in the Context of Harvesting Language Data off the Web

Fei Xia
University of Washington
Seattle, WA 98195, USA
fxia@u.washington.edu

William D. Lewis
Microsoft Research
Redmond, WA 98052, USA
wilewis@microsoft.com

Hoifung Poon
University of Washington
Seattle, WA 98195, USA
hoifung@cs.washington.edu

Abstract

As the arm of NLP technologies extends beyond a small core of languages, techniques for working with instances of language data across hundreds to thousands of languages may require revisiting and recalibrating the tried and true methods that are used. Of the NLP techniques that has been treated as "solved" is language identification (language ID) of written text. However, we argue that language ID is far from solved when one considers input spanning not dozens of languages, but rather hundreds to thousands, a number that one approaches when harvesting language data found on the Web. We formulate language ID as a coreference resolution problem and apply it to a Web harvesting task for a specific linguistic data type and achieve a much higher accuracy than long accepted language ID approaches.

1 Introduction

A large number of the world's languages have been documented by linguists; it is now increasingly common to post current research and data to the Web, often in the form of language snippets embedded in scholarly papers. A particularly common format for linguistic data posted to the Web is "interlinearized text", a format used to present language data and analysis relevant to a particular argument or investigation. Since interlinear examples consist of orthographically or phonetically encoded language data aligned with an English translation, the "corpus" of interlinear examples found on the Web, when taken together, constitute a significant multilingual, parallel corpus covering hundreds to thousands of the world's languages. Previous work has discussed methods for harvesting interlinear text off the Web (Lewis, 2006), enriching it via structural projections (Xia and Lewis, 2007), and even making it available to typological analyses (Lewis and Xia, 2008) and search (Xia and Lewis, 2008).

One challenge with harvesting interlinear data off the Web is language identification of the harvested data. There have been extensive studies on language identification (language ID) of written text, and a review of previous research on this topic can be found in (Hughes et al., 2006). In general, a language ID method requires a collection of text for training, something on the order of a thousand or more characters. These methods work well for languages with rich language resources; for instance, Cavnar and Trenkle's N-gram-based algorithm achieved an accuracy as high as 99.8% when tested on newsgroup articles across eight languages (Cavnar and Trenkle, 1994). However, the performance is much worse (with accuracy dropping to as low as 1.66%) if there is very little language data for training and the number of languages being evaluated reaches a few hundred.

In this paper, we treat the language ID of harvested linguistic data as a coreference resolution problem. Our method, although narrowly focused on this very specific data type, makes it possible to collect small snippets of language data across hundreds of languages and use the data for linguistic search and bootstrapping NLP tools.

2 Background

2.1 Interlinear glossed text (IGT)

In linguistics, the practice of presenting language data in interlinear form has a long history, going back at least to the time of the structuralists. Interlinear Glossed Text, or *IGT*, is often used to present data and analysis on a language that the reader may not know much about, and is frequently included in scholarly linguistic documents. The canonical form of an IGT consists

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 870–878,
Athens, Greece, 30 March – 3 April 2009.

of three lines: a line for the language in question (i.e., the *language line*), an English gloss line, and an English translation. Table 1 shows the beginning of a linguistic document (Baker and Stewart, 1996) which contains two IGTs: one in lines 30-32, and the other in lines 34-36. The line numbers are added for the sake of convenience.

1: THE ADJ/VERB DISTINCTION: **EDO** EVIDENCE
2:
3: Mark C. Baker and Osamuyimen **Thompson** Stewart
4: McGill University
....
27: The following shows a similar minimal pair from **Edo**,
28: a **Kwa** language spoken in Nigeria (Agheyisi 1990).
29:
30: (2) a. *Èmèrí mòsé.*
31: Mary be.beautiful(V)
32: 'Mary is beautiful.'
33:
34: b. *Èmèrí *(yé) mòsé.*
35: Mary be.beautiful(A)
36: 'Mary is beautiful (A).'
...

Table 1: A linguistic document that contains IGT: words in boldface are potential language names

2.2 The Online Database of Interlinear text (ODIN)

ODIN, the Online Database of INterlinear text, is a resource built from data harvested from scholarly documents (Lewis, 2006). It was built in three steps: (1) crawling the Web to retrieve documents that may contain IGT, (2) extracting IGT from the retrieved documents, and (3) identifying the language codes of the extracted IGTs. The identified IGTs are then extracted and stored in a database (the ODIN database), which can be easily searched with a GUI interface.[1]

ODIN currently consists about 189,000 IGT instances extracted from three thousand documents, with close to a thousand languages represented. In addition, there are another 130,000 additional IGT-bearing documents that have been crawled and are waiting for further process. Once these additional documents are processed, the database is expected to expand significantly.

ODIN is a valuable resource for linguists, as it can be searched for IGTs that belong to a particular language or a language family, or those that contain a particular linguistic construction (e.g., passive, wh-movement). In addition, there have been some preliminary studies that show the benefits of using the resource for NLP. For instance, our previous work shows that automatically enriched IGT data can be used to answer typological questions (e.g., the canonical word order of a language) with a high accuracy (Lewis and Xia, 2008), and the information could serve as prototypes for prototype learning (Haghighi and Klein, 2006).

[1] http://odin.linguistlist.org

3 The language ID task for ODIN

As the size of ODIN increases dramatically, it is crucial to have a reliable module that automatically identifies the correct language code for each new extracted IGT to be added to ODIN. The current ODIN system uses two language identifiers: one is based on simple heuristics, and the other on Cavnar and Trenkle's algorithm (1994). However, because the task here is very different from a typical language ID task (see below), both algorithms work poorly, with accuracy falling below 55%. The focus of this paper is on building new language identifiers with a much higher accuracy.

3.1 The data set

A small portion of the IGTs in ODIN have been assigned the correct language code semi-automatically. Table 2 shows the size of the data set. We use it for training and testing, and all results reported in the paper are the average of running 10-fold cross validation on the data set unless specified otherwise.

Table 2: The data set for the language ID task

# of IGT-bearing documents	1160
# of IGT instances	15,239
# of words on the language lines	77,063
# of languages	638

3.2 The special properties of the task

The task in hand is very different from a typical language ID task in several respects:

- Large number of languages: The number of languages in our data set is 638 and that of the current ODIN database is close to a thousand. As more data is added to ODIN, the number of languages may reach several thousand as newly added linguistic documents could refer to any of approximately eight thousand living or dead languages.

- The use of language code: When dealing with only a few dozen languages, language names might be sufficient to identify languages. This is not true when dealing with a large number of languages, because some languages have multiple names, and some language names refer to multiple languages (see Section 4.2). To address this problem, we use language codes, since we can (mostly) ensure that each language code maps to exactly one language, and each language maps to exactly one code.

- Unseen languages: In this data set, about 10% of IGT instances in the test data belong to some languages that have never appeared in the training data. We call it the *unseen language problem*. This problem turns out to be the major obstacle to existing language ID methods.

- Extremely limited amount of training data per language: On average, each language in the training data has only 23 IGTs (116 word tokens in the language lines) available, and 45.3% of the languages have no more than 10 word tokens in the training data.

- The length of test instances: The language lines in IGT are often very short. The average length in this data set is 5.1 words. About 0.26% of the language lines in the data set are totally empty due to the errors introduced in the crawling or IGT extraction steps.

- Encoding issues: For languages that do not use Roman scripts in their writing system, the authors of documents often choose to use Romanized scripts (e.g., pinyin for Chinese), making the encoding less informative.

- Multilingual documents: About 40% of documents in the data set contain IGTs from multiple languages. Therefore, the language ID prediction should be made for each individual IGT, not for the whole document.

- Context information: In this task, IGTs are part of a document and there are often various cues in the document (e.g., language names) that could help predict the language ID of specific IGT instances.

Hughes and his colleagues (2006) identified eleven open questions in the domain of language ID that they believed were not adequately addressed in published research to date. Interestingly, our task encounters eight out of the eleven open questions. Because of these properties, existing language ID algorithms do not perform well when applied to the task (see Section 6).

4 Using context information

Various cues in the document can help predict the language ID of IGTs, and they are represented as features in our systems.

4.1 Feature templates

The following feature templates are used in our experiments.

(F1): The nearest language that precedes the current IGT.

(F2): The languages that appear in the neighborhood of the IGT or at the beginning or the end of a document.[2] Another feature checks the most frequent language occurring in the document.

(F3): For each language in the training data, we build three token lists: one for word unigrams, one for morph unigrams and the third for character ngrams ($n \leq 4$). These word lists are compared with the token lists built from the language line of the current IGT.

(F4): Similar to (F3), but the comparison is between the token lists built from the current IGT with the ones built from other IGTs in the same document. If some IGTs in the same document share the same tokens, they are likely to belong to the same language.

Here, all the features are binary: for features in F3 and F4, we use thresholds to turn real-valued features into binary ones. F1-F3 features can be calculated by looking at the documents only, whereas F4 features require knowing the language codes of other IGTs in the same document.

4.2 Language table

To identify language names in a document and map language names to language codes, we need a language table that lists all the (language code,

[2] For the experiments reported here, we use any line within 50 lines of the IGT or the first 50 or the last 50 lines of the document.

language name) pairs. There are three existing language tables: (1) ISO 639-3 maintained by SIL International,[3] (2) the 15th edition of the Ethnologue,[4] and (3) the list of ancient and dead languages maintained by LinguistList.[5] [6] We merged the three tables, as shown in Table 3.

Table 3: Various language name tables

Language table	# of lang codes	# of lang (code, name) pairs
(1) ISO 639-3	7702	9312
(2) Ethnologue v15	7299	42789
(3) LinguistList table	231	232
Merged table	7816	47728

The mapping between language names and language codes is many-to-many. A language code often has several alternate names in addition to the primary name. For instance, the language code *aaa* maps to names such as Alumu, Tesu, Arum, Alumu-Tesu, Alumu, Arum-Cesu, Arum-Chessu, and Arum-Tesu. While most language names map to only one language code, there are exceptions. For instance, the name *Edo* can map to either *bin* or *lew*. Out of 44,071 unique language names in the merged language table, 2625 of them (5.95%) are ambiguous.[7]

To identify language names in a document, we implemented a simple language name detector that scans the document from left to right and finds the longest string that is a language name according to the language table. The language name is then mapped to language codes. If a language name is ambiguous, all the corresponding language codes are considered by later stages. In Table 1, the language names identified by the detector are in boldface. The detector can produce false positive (e.g., *Thompson*) because a language name can have other meanings. Also, the language table is by no means complete and the detector is not able to recognize any language names that are missing from the table.

[3] http://www.sil.org/iso639-3/download.asp

[4] http://www.ethnologue.com/codes/default.asp#using

[5] http://linguistlist.org/forms/langs/GetListOfAncientLgs.html

[6] While ISO 639-3 is supposed to include all the language codes appearing in the other two lists, there is a lag in the adoption of new codes, which means the ISO 639-3 list continues to be somewhat out-of-date with the lists from which it is compiled since these other lists change periodically.

[7] Among the ambiguous names, 1996 names each map to two language codes, 407 map to three codes, 130 map to four codes, and so on. The most ambiguous name is *Miao*, which maps to fourteen language codes.

5 Formulating the language ID task

The language ID task here can be treated as two different learning problems.

5.1 As a classification problem

The language ID task can be treated as a classification problem. A classifier is a function that maps a training/test instance x to a class label y, and y is a member of a pre-defined label set C. For language ID, the training/test instance corresponds to a document (or an IGT in our case), and C is the set of language codes. We call this approach the *classification (CL) approach*.

Most, if not all, of previous language ID methods, fall into this category. They differ with respect to the underlying learning algorithms and the choice of features or similarity functions. When applying a feature-based algorithm (e.g., Maximum entropy) and using the features in Section 4.1, the feature vectors for the two IGTs in Table 1 are shown in Table 4. Each line has the format "*instance_name true_lang_code feat_name1 feat_name2 ...*", where feat_names are the names of features that are present in the instance. Take the first IGT as an example, its true language code is *bin*; the nearest language name (*nearLC*) is *Edo* whose language code is *bin* or *lew*; the languages that appear before the IGT includes Edo (*bin* or *lew*), Thompson (*thp*), and so on. The presence of *LMw1_bin* and *LMm1_bin* means that the overlap between the word/morph lists for *bin* and the ones built from the current IGT is higher than some threshold. The feature vector for the second IGT looks similar, except that it includes a F4 feature *IIw1_bin*, which says that the overlap between the word list built from the other IGTs in the same document with language code bin and the word list built from the current IGT is above a threshold. Note that language codes are part of feature names; therefore, a simple feature template such as nearest language (*nearLC*) corresponds to hundreds or even thousands of features (*nearLC_xxx*).

The *CL* approach has several major limitations. First, it cannot handle the *unseen language problem*: if an IGT in the test data belongs to a language that does not appear in the training data, this approach cannot classify it correctly. Second, the lack of parameter tying in this approach makes it unable to generalize between different languages. For instance, if the word *German* appears right before an IGT, the IGT is likely to be German. The

igt1 bin nearLC_bin nearLC_lew prev50_bin prev50_lew prev50_thp ... LMw1_bin LMm1_bin ...
igt2 bin nearLC_bin nearLC_lew prev50_bin prev50_lew prev50_thp ... LMw1_bin LMm1_bin ... IIw1_bin ...

Table 4: Feature vectors for the IGTs in Table 1 when using the *CL* approach (Edo: bin/lew, Thompson: thp, Kwa: etu/fip/kwb)

same is true if the word *German* is replaced by another language name. But this property cannot be leveraged easily by the *CL* approach without modifying the learning algorithm. This results in a proliferation of parameters, making learning harder and more prone to overfitting.

5.2 As a coreference resolution problem

A different way of handling the language ID task is to treat it as a coreference resolution problem: a mention is an IGT or a language name appearing in a document, an entity is a language code, and finding the language code for an IGT is the same as linking a mention (i.e., an IGT) to an entity (i.e., a language code).[8] We call this approach the *CoRef* approach. The major difference between the *CL* approach and the *CoRef* approach is the role of language code: in the former, language code is a class label to be used to tag an IGT; and in the latter, language code is an entity which an IGT can be linked to.

The language ID task shares many similarities with a typical coreference resolution task. For instance, language names are similar to proper nouns in that they are often unambiguous. IGT instances are like pronouns in that they often refer to language names appearing in the neighborhood. Once the language ID task is framed as a *CoRef* problem, all the existing algorithms on *CoRef* can be applied to the task, as discussed below.

5.2.1 Sequence labeling using traditional classifiers

One common approach to the *CoRef* problem processes the mentions sequentially and determine for each mention whether it should start a new entity or be linked to an existing mention (e.g., (Soon et al., 2001; Ng and Cardie, 2002; Luo, 2007)); that is, the approach makes a series of decisions, one decision per (mention, entity) pair. Applying this to the language ID task, the (mention, entity) pair would correspond to an (IGT, lang_code) pair, and each decision would have two possibilities: *Same* when the IGT belongs to the language or *Diff* when the IGT does not. Once the decisions are made for all the pairs, a post-processing procedure would check all the pairs for an IGT and link the IGT to the language code with which the pair has the highest confidence score.

Using the same kinds of features in Section 4.1, the feature vectors for the two IGTs in Table 1 are shown in Table 5. Comparing Table 4 and 5 reveals the differences between the *CL* approach and the *CoRef* approach: the *CoRef* approach has only two class labels (*Same* and *Diff*) where the *CL* approach has hundreds of labels (one for each language code); the *CoRef* approach has much fewer number of features because language code is not part of feature names; the *CoRef* approach has more training instances as each training instance corresponds to an (IGT, lang_code) pair.

igt1-bin same nearLC prev50 LMw1 LMm1 ...
igt1-lew diff nearLC prev50 ...
igt1-thp diff prev50 ...
...
igt2-bin same nearLC prev50 LMw1 LMm1 IIw1 ...
igt2-lew diff nearLC prev50 ...
igt2-thp diff prev50 ...
...

Table 5: Feature vectors for the IGTs in Table 1 when using the *CoRef* approach with sequence labeling methods

5.2.2 Joint Inference Using Markov Logic

Recently, joint inference has become a topic of keen interests in both the machine learning and NLP communities (e.g., (Bakir et al., 2007; Sutton et al., 2006; Poon and Domingos, 2007)). There have been increasing interests in formulating coreference resolution in a joint model and conducting joint inference to leverage dependen-

[8] There are minor differences between the language ID and coreference resolution tasks. For instance, each entity in the language ID task must be assigned a language code. This means that ambiguous language names will evoke multiple entities, each with a different language code. These differences are reflected in our algorithms.

cies among the mentions and entities (e.g., (Wellner et al., 2004; Denis and Baldridge, 2007; Poon and Domingos, 2008)). We have built a joint model for language ID in *Markov logic* (Richardson and Domingos, 2006).

Markov logic is a probabilistic extension of first-order logic that makes it possible to compactly specify probability distributions over complex relational domains. A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by $P(x) = (1/Z) \exp\left(\sum_i w_i f_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith clause, $f_i = 1$ if the ith clause is true, and $f_i = 0$ otherwise. Conditional probabilities can be computed using Markov chain Monte Carlo (e.g., MC-SAT (Poon and Domingos, 2006)). The weights can be learned using pseudo-likelihood training with L-BFGS (Richardson and Domingos, 2006). Markov logic is one of the most powerful representations for joint inference with uncertainty, and an implementation of its existing learning and inference algorithms is publicly available in the Alchemy package (Kok et al., 2007).

To use the features defined in Section 4.1, our MLN includes two evidence predicates: the first one is $\texttt{HasFeature}(\texttt{i}, \texttt{l}, \texttt{f})$ where f is a feature in $F1$-$F3$. The predicate is true iff the IGT-language pair $(\texttt{i}, \texttt{l})$ has feature f. The second predicate is $\texttt{HasRelation}(\texttt{i1}, \texttt{i2}, \texttt{r})$ where r is a relation that corresponds to a feature in $F4$; this predicate is true iff relation $\texttt{r}$ holds between two IGTs $\texttt{i1}, \texttt{i2}$. The query predicate is $\texttt{IsSame}(\texttt{i}, \texttt{l})$, which is true iff IGT $\texttt{i}$ is in language $\texttt{l}$. Table 6 shows the predicates instantiated from the two IGTs in Table 1.

The language ID task can be captured in our MLN with just three formulas:

$$\texttt{IsSame}(\texttt{i}, \texttt{l})$$

$$\texttt{HasFeature}(\texttt{i}, \texttt{l}, +\texttt{f}) \Rightarrow \texttt{IsSame}(\texttt{i}, \texttt{l})$$

$$\texttt{HasRelation}(\texttt{i1}, \texttt{i2}, +\texttt{r}) \wedge \texttt{IsSame}(\texttt{i1}, \texttt{l}) \Rightarrow \texttt{IsSame}(\texttt{i2}, \texttt{l})$$

The first formula captures the default probability that an IGT belongs to a particular language.

```
IsSame(igt1, bin)
HasFeature(igt1, bin, nearLC)
HasFeature(igt1, bin, prev50)
HasFeature(igt1, bin, LMw1)
...
HasFeature(igt1, lew, nearLC)
HasFeature(igt1, lew, prev50)
...
IsSame(igt2, bin)
HasFeature(igt2, bin, nearLC)
HasFeature(igt2, bin, prev50)
HasFeature(igt2, bin, LMw1)
...
HasRelation(igt1, igt2, IIw1)
...
```

Table 6: The predicates instantiated from the IGTs in Table 1

The second one captures the conditional likelihoods of an IGT being in a language given the features. The third formula says that two IGTs probably belong to the same language if they have a certain relation r.

The plus sign before f and r in the formulas signifies that the MLN will learn a separate weight for each individual feature f and relation r. Note that there is no plus sign before i and l, allowing the MLN to achieve parameter tying by sharing the same weights for different instances or languages.

5.2.3 The advantage of the *Coref* approach

Both methods of the *CoRef* approach address the limitations of the *CL* approach: both can handle the *unseen language problem*, and both do parameter tying in a natural way. Not only does parameter tying reduce the number of parameters, it also makes it possible to accumulate evidence among different languages and different IGTs.

6 Experiments

In this section, we compare the two approaches to the language ID task: the *CL* approach and the *CoRef approach*. In our experiments, we run 10-fold cross validation (90% for training and 10% for testing) on the data set in Table 2 and report the average of language ID accuracy.

The two approaches have different upper bounds. The upper bound of the *CL* approach is the percentage of IGTs in the test data that belong to a *seen* language. The upper bound of the *CoRef* approach is the percentage of IGTs in the test data that belong to a language whose language name appears in the same document. For the data set in Table 2, the upper bounds are 90.33% and

Table 7: The performance of the *CL* approach (# of classes: about 600, # of training instances=13,723)

	Upper bound of *CL* approach	TextCat	MaxEnt classifier using context information			
			F1	F1-F2	F1-F3	F1-F4 (cheating)
# of features	N/A	N/A	769	5492	8226	8793
w/o the language filter	90.33	51.38	49.74	61.55	64.19	66.47
w/ the language filter	*88.95*	60.72	56.69	64.95	67.03	69.20

97.31% respectively. When the training data is much smaller, the upper bound of the *CL* approach would decrease tremendously, whereas the upper bound of the *CoRef* approach remains the same.

6.1 The *CL* approach

As mentioned before, most existing language ID algorithm falls into this category. We chose TextCat,[9] an implementation of Cavnar-Trenkle's algorithm (1994), as an example of these algorithms. In order to take advantage of the context information, we trained several classifiers (e.g., decision tree, Naive Bayes, and maximum entropy) using the Mallet package (McCallum, 2002) and a SVM classifier using the libSVM package (Chang and Lin, 2001).

The result is in Table 7. The first column shows the upper bound of the *CL* approach; the second column is the result of running TextCat;[10] the rest of the table lists the result of running a MaxEnt classifier with different feature sets.[11] F4 features require knowing the language code of other IGTs in the document. In the F1-F4 cheating experiments, the language codes of other IGTs come from the gold standard. We did not implement beam search for this because the difference between the cheating results and the results without F4 features is relatively small and both are much worse than the results in the *CoRef* approach.

In Table 7, the first row shows the number of features; the second row shows the accuracy of the two classifiers; the last row is the accuracy when a post-processing filter is added: the filter takes the ranked language list produced by a classifier, throws away all the languages in the list that do not appear in the document, and then outputs the highest ranked language in the remaining list.

There are several observations. First, applying the post-processing filter improves performance, albeit it also lowers the upper bound of algorithms as the correct language names might not appear in the document. Second, the MaxEnt classifier has hundreds of classes, thousands of features, and millions of model parameters. This will cause severe sparse data and overfitting problems.

6.2 The *CoRef* approach

For the *CoRef* approach, we built two systems as described in Section 5: the first system is a MaxEnt classifier with beam search, and the second one is a MLN for joint inference.[12] The results are in Table 8.[13]

In the first system, the values of F4 features for the test data come from the gold standard in the F1-F4 cheating experiments, and come from beam search in the non-cheating experiments.[14] In the second system, the predicate $\texttt{HasRelation}(\texttt{i1}, \texttt{i2}, \texttt{r})$ instantiated from the test data is treated as evidence in the F1-F4 cheating experiments, and as query in the F1-F4 non-cheating experiments.

The results for the two systems are very similar since they use same kinds of features. However, with Markov logic, it is easy to add predicates and formulas to allow joint inference. Therefore, we believe that Markov logic offers more potential to incorporate arbitrary prior knowledge and leverage further opportunities in joint inference.

Tables 7-8 show that, with the same kind of features and the same amount of training data, the *CoRef* approach has higher upper bound, fewer model parameters, more training instances, and much higher accuracy than the *CL* approach. This study shows that properly formulating a task into a learning problem is very important.

[9] http://odur.let.rug.nl/ vannoord/TextCat/

[10] We varied the lexicon size (m) – an important tuned parameter for the algorithm – from 100 and 800 and observed a minor change to accuracy. The numbers reported here are with lexicon size set to 800.

[11] The MaxEnt classifier slightly outperforms other classifiers with the same feature set.

[12] For learning and inference, we used the existing implementations of pseudo-likelihood training and MC-SAT in Alchemy with default parameters.

[13] No language filter is needed since the approach links an IGT to only the language names appearing in the document.

[14] It turns out that for this task the size of beam does not matter much and simply using the top choice by the MaxEnt classifier for each IGT almost always produces the best results, so that is the setting used for this table and Table 9.

Table 8: The performance of the *CoRef* approach (# of classes=2, # of training instances=511,039)

	Upper bound of CoRef approach	F1	F1-F2	F1-F3	F1-F4 (cheating)	F1-F4 (Non-cheating)
# of features	N/A	2	12	17	22	22
Sequence labeling	*97.31*	54.37	66.32	83.49	90.26	85.10
Markov logic model	*97.31*	54.98	65.94	83.44	90.37	84.70

Table 9: The performance of the *CoRef* approach with less training data (the upper bound of the *Coref* approach remains 97.31%)

% of training data used	F1	F1-F2	F1-F3	F1-F4 (cheating)	F1-F4 (non-cheating)	Upper bound of the *CL* approach
0.1%	54.37	54.84	65.28	81.21	70.15	1.66
0.5%	54.37	62.78	76.74	87.17	80.24	21.15
1.0%	54.37	60.58	76.09	87.24	81.20	28.92
10%	54.37	62.13	77.07	87.20	83.08	54.45

6.3 Experiments with much less data

Table 8 shows that the *CoRef* approach has very few features and a much larger number of training instances; therefore, it is likely that the approach would work well even with much less training data. To test the idea, we trained the model with only a small fraction of the original training data and tested on the same test data. The results with the first system are in Table 9. Notice that the upper bound of the *CoRef* approach remains the same as before. In contrast, the upper bound for the *CL* model is much lower, as shown in the last column of the table. The table shows when there is very little training data, the *CoRef* approach still performs decently, whereas the *CL* approach would totally fail due to the extremely low upper bounds.

6.4 Error analysis

Several factors contribute to the gap between the best *CoRef* system and its upper bound. First, when several language names appear in close range, the surface positions of the language names are often insufficient to determine the prominence of the languages. For instance, in pattern "*Similar to L1, L2 ...*", $L2$ is the more prominent than $L1$; whereas in pattern "*L1, a L2 language, ...*", $L1$ is. The system sometimes chooses a wrong language in this case.

Second, the language name detector described in Section 4.2 produces many false negative (due to the incompleteness of the language table) and false positive (due to the fact that language names often have other meanings).

Third, when a language name is ambiguous, choosing the correct language code often requires knowledge that might not even be present in the document. For instance, a language name could refer to a list of related languages spoken in the same region, and assigning a correct language code would require knowledge about the subtle differences among those languages.

7 Conclusion and future work

In this paper we describe a language identification methodology that achieves high accuracy with a very small amount of training data for hundreds of languages, significantly outperforming existing language ID algorithms applied to the task. The gain comes from two sources: by taking advantage of context information in the document, and by formulating the task as a coreference resolution problem.

Our method can be adapted to harvest other kinds of linguistic data from the Web (e.g., lexicon entries, word lists, transcriptions, etc.) and build other ODIN-like resources. Providing a means for rapidly increasing the amount of data in ODIN, while at the same time *automatically* increasing the number of languages, can have a significant positive impact on the linguistic community, a community that already benefits from the existing search facility in ODIN. Likewise, the increased size of the resulting ODIN database could provide sufficient data to bootstrap NLP tools (e.g., POS taggers and parsers) for a large number of low-density languages, greatly benefitting both the fields of linguistics and NLP.

Acknowledgements This work has been supported, in part, by the NSF grants BCS-0748919 and BCS-0720670 and ONR grant N00014-08-1-0670. We would also like to thank three anonymous reviewers for their valuable comments.

References

Mark C. Baker and Osamuyimen Thompson Stewart. 1996. Unaccusativity and the adjective/verb distinction: Edo evidence. In *Proceedings of the Fifth Annual Conference on Document Analysis and Information Retrieval (SDAIR)*, Amherst, Mass.

G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar, and S. Vishwanathan (eds). 2007. *Predicting Structured Data*. MIT Press.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. of the Conference on Human Language Technologies (HLT/NAACL 2007)*, pages 236–243, Rochester, New York, April.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 881–888, Sydney, Australia, July. Association for Computational Linguistics.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. 2006. Reconsidering language identification for written language resources. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC2006)*, pages 485–488, Genoa, Italy.

S. Kok, P. Singla, M. Richardson, P. Domingos, M. Sumner, H Poon, and D. Lowd. 2007. The Alchemy system for statistical relational AI. Technical report, Dept. of CSE, Univ. of Washington.

William Lewis and Fei Xia. 2008. Automatically Identifying Computationally Relevant Typological Features. In *Proc. of the Third International Joint Conference on Natural Language Processing (IJCNLP-2008)*, Hyderabad, India.

William Lewis. 2006. ODIN: A Model for Adapting and Enriching Legacy Infrastructure. In *Proc. of the e-Humanities Workshop, held in cooperation with e-Science 2006: 2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam.

Xiaoqiang Luo. 2007. Coreference or not: A twin model for coreference resolution. In *Proc. of the Conference on Human Language Technologies (HLT/NAACL 2007)*, pages 73–80, Rochester, New York.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Vincent Ng and Claire Cardie. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 104–111, Philadelphia.

H. Poon and P. Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. of AAAI-06*.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI)*, pages 913–918, Vancouver, Canada. AAAI Press.

H. Poon and P. Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proc. of the 13th Conf. on Empirical Methods in Natural Language Processing (EMNLP-2008)*.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, pages 107–136.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4).

Charles Sutton, Andrew McCallum, and Jeff Bilmes (eds.). 2006. *Proc. of the HLT/NAACL-06 Workshop on Joint Inference for Natural Language Processing*.

B. Wellner, A. McCallum, F. Peng, and M. Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proc. of the 20th Conference on Uncertainty in AI (UAI 2004)*.

Fei Xia and William Lewis. 2007. Multilingual structural projection across interlinear text. In *Proc. of the Conference on Human Language Technologies (HLT/NAACL 2007)*, pages 452–459, Rochester, New York.

Fei Xia and William Lewis. 2008. Repurposing Theoretical Linguistic Data for Tool Development and Search. In *Proc. of the Third International Joint Conference on Natural Language Processing (IJCNLP-2008)*, Hyderabad, India.