Repurposing Theoretical Linguistic Data for Tool Development and Search

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

William D. Lewis[*]
Microsoft Research
Redmond, WA 98052-6399
wilewis@microsoft.com

Abstract

For the majority of the world's languages, the number of linguistic resources (e.g., annotated corpora and parallel data) is very limited. Consequently, supervised methods, as well as many unsupervised methods, cannot be applied directly, leaving these languages largely untouched and unnoticed. In this paper, we describe the construction of a resource that taps the large body of linguistically analyzed language data that has made its way to the Web, and propose using this resource to bootstrap NLP tool development.

1 Introduction

Until fairly recently, most NLP research has focused on the ten or so majority languages of the world, the canonical *high density* languages. *Low density*, or *resource poor languages* (RPLs), have more recently captured the interest of NLP research, mostly because of recent advances in computational technologies and computing power. As indicated by their name, RPLs suffer from a lack of resources, namely data. Supervised learning techniques generally require large amounts of annotated data, something that is nonexistent or scare for most RPLs. A greater number of RPLs, however, have raw data that is available, and the amount and availability of this raw data is increasing every day as more of it makes its way to the Web. Likewise, advances in un- and semi-supervised learning techniques have made raw data more readily viable for tool development. Still, however, such techniques often require "seeds", or "prototypes" (c.f., (Haghighi and Klein, 2006)) which are used to prune search spaces or direct learners.

An important question is how to create such seeds for the hundreds to thousands of RPLs. We describe the construction of a resource that taps the large body of linguistically analyzed language data that has made its way to the Web, and propose using this resource as a means to bootstrap NLP tool development. Interlinear Glossed Text, or IGT, a semi-structured data type quite common to the field of linguistics, is used to present data and analysis for a language and is generally embedded in scholarly linguistic documents as part of a larger analysis. IGT's unique structure — effectively each instance consists of a bitext between English and some target language — can be easily enriched through alignment and projection (e.g., (Yarowsky and Ngai, 2001), (Hwa et al., 2002)). The reader will note that the IGT instance in Example (1) consists of a bitext between some target language on the first line, or the *target line* (in this case in Welsh), and a third line in English, the *translation line*. The canonical IGT form, which this example is representative of, has intervening linguistic annotations and glosses on a second line, the *gloss line*. Because the gloss line aligns with words and morphemes on the target line, and contains glosses that are similar to words on the translation line, it can serve as a bridge between the target and translation lines; high word alignment accuracy between the three lines can be achieved without requiring parallel data or bilingual dictionaries (Xia and Lewis, 2007). Furthermore, the gloss line provides additional information about the target language data, such as a variety of grammatical annotations, including verbal and tense markers (e.g., 3sg), case markers, etc., all of which can provide useful knowledge about the language.

(1)

Rhoddodd	yr	athro	lyfr	i'r	bachgen	ddoe
gave-3sg	the	teacher	book	to-the	boy	yesterday

"The teacher gave a book to the boy yesterday"
(Bailyn, 2001)

ODIN, the Online Database of INterlinear text (Lewis, 2006), is a resource built over the past few years from data harvested from scholarly documents. Currently, ODIN has over 41,581 instances of IGT for 944 languages, and the number of IGT instances is expected to double or triple in the near-term as new methods for collecting data are brought online. Although the number of instances per language varies, e.g., the maximum currently is 2,891 instances (for

[*]The work described in this document was done while Lewis was faculty at the University of Washington.

Table 1: The numbers of languages in ODIN

Range of IGT instances	# of languages	# of instances	% of instances
1000-2891	10	15019	36.11
500-999	11	8111	19.50
250-499	18	6274	15.08
100-249	22	3303	7.94
50-99	38	2812	6.76
25-49	60	2089	5.02
10-24	127	1934	4.65
1-9	658	2039	4.91

Japanese), and the overall number per language may appear small, it is still possible to harvest significant value from IGT for targeted RPLs. In this paper, we present the ODIN database and methods used to create it. We also present methods we have employed to enrich IGT in order to make it more readily useful for bootstrapping NLP tools. Because the canon of knowledge embodied in the hundred or so years of linguistic analysis remains virtually untapped by the NLP community, we provide a bridge between the communities by providing linguistic data in a way that NLP researchers will find useful. Likewise, because IGT is a common linguistic data type, we provide a search facility over these data, which has already been found to be quite useful to the theoretical linguistics community.

2 Building ODIN

ODIN currently has 41,581 IGT instances for 944 languages. Table 1 shows the number of languages that fall into buckets defined by the number of IGT instances for each language. For instance, the fourth row ("bucket") says that 22 languages each have 100 to 249 IGT instances, and the 3,303 instances in this bucket account for 7.94% of all instances. ODIN is built in three steps, as described below.[1]

2.1 Crawling for IGT documents

Because a large number of instances of IGT exist on the Web,[2] we have focused on searching for these instances. The major difficulty with locating documents that contain IGT, however, is reducing the size of the search space. We decided very early in the development of ODIN that unconstrained Web crawling was too time and resource intensive a process to be feasible, mostly due to the Web's massive size. We discovered that highly focused metacrawls were far more fruitful. Metacrawling essentially involves throwing queries against an existing search engine, such as Google, Yahoo or MSN Live, and crawling only the pages returned by those queries. We found that the most successful queries were those that used strings contained within IGT itself, e.g. grammatical annotations, or *grams*, such as 3sg, NOM, ACC, etc. In addition, we found precision increased when we included two or more search terms per query, with the most successful queries being those which combined grams and language names. Thus, for example, although NOM alone returned a large number of linguistic documents, NOM combined with ACC (or any other high frequency term), or a language name, returned a far less noisy and far more relevant set of documents.

Other queries we have developed include: queries by language names and language codes (drawn from the Ethnologue database (Gordon, 2005), which contains about 40,000 language names and their variants), by linguists' names and the languages they work on (drawn from the Linguist List's linguist database), by linguistically relevant terms (drawn from the SIL linguistic glossary), and by particular words or morphemes found in IGT and their grammatical markup. Table 2 shows the statistics for the most successful crawls and their related search term "types". Calculated from the top 100 queries for each type, the table presents the most successful query types, the average number of documents returned for each, the average number of documents in which IGT was actually found, and the average number of IGT instances netted by each query. The most relevant measure of success is the number of IGT instances returned (the obvious focus of our crawling); in turn, the most successful query types are those which contain a combination of grams and language names.[3]

[1]The work of creating ODIN, in some ways, speaks to the need of standardizing IGT (perhaps along with other linguistic data types) such that both humans and machines can more readily consume the data. Some recent efforts to develop standards for encoding IGT (e.g., (Hughes et al., 2003), (Bickel et al., 2004)) have met with limited success, however, since they have not been widely recognized and even less frequently adopted. Over time it is our hope that these or other standards will see wider use thus eliminating the need for much of the work proposed here.

[2]Although we have no direct data about the total number of IGT instances that exist on the Web, we hypothesize that the total supply is at least several hundred thousand instances. Given that ODIN contains 41,581 instances which have been extracted from approximately 3,000 documents, and given that we have located at least 60,000 more documents that might contain IGT, we feel our estimate to be reasonable.

[3]Note that target documents are often returned by multiple queries. For instance, the documents returned by "NOM+ACC+Icelandic" will also be returned by the individual query terms "NOM", "ACC", and "Icelandic".

Table 2: The Most Successful Query Types

Query **Type**	Avg # docs	Avg # docs w/ IGT	Avg # IGTs
Gram(s)	1184	239	50
Language name(s)	1314	259	33
Both grams and names	1536	289	77
Language words	1159	193	0

2.2 IGT detection

After crawling, the next step is to identify IGT instances in the retrieved documents. This is a difficult task for which machine learning methods are well suited.

2.2.1 Difficulty in IGT detection

The canonical form of IGT, as presented in Section 1, consists of three parts and each part is on a single line. However, many IGT instances do not follow the canonical format for several reasons. First, when IGT examples appear in a group, very often the translation or glosses are dropped for some examples in the group because the missing parts can be recovered from the context, resulting in *two-part* IGT. In other cases, some IGT examples include multiple target transcriptions (e.g., one part in the native script, and another in a latin transliteration) or even, in rare cases, multiple translations.

Second, dictated by formatting constraints, long IGT examples may need to be wrapped one or more times, and there are no conventions on how wrapping should be done, nor how many times it can be done. For short IGT examples, sometimes linguists put the translation to the right of the target line rather than below it. As a result, each part of IGT examples may appear on multiple lines and multiple parts can appear on a single line.

Third, most IGT-bearing documents on the Web are in PDF, and the PDF-to-text conversion tools will sometimes corrupt IGT instances (most often on the target line). In some instances, some words or morphemes on the target line are inadvertently dropped in the conversion, or are displaced up or down a line. Finally, an IGT instance could fall into multiple categories. For instance, a two-part IGT instance could have a corrupted target line. All of this makes the detection task difficult.

2.2.2 Applying machine learning methods

The first system that we designed for IGT detection used regular expression "templates", effectively looking for text that resembled IGT. An example is shown in (2), which matches any three-line instance (e.g., the IGT instance in (1)) such that the first line starts with an example number (e.g., *(1)*) and the third line starts with a quotation mark.

(2)
```
\s*\(\d+\).*\n
\s*.*\n
\s*\[‘’"].*\n
```

Unfortunately, this approach tends to over-select when applied to the documents crawled from the Web. Further, many true IGT instances do not match any of hand-written templates due to the issues mentioned in the previous section. As a result, both precision and recall are quite low (see Table 4).

Given the irregular structure of IGT instances, a statistical system is likely to outperform a rule-based system. In our second system, we treat the IGT detection task as a sequence labeling problem, and apply machine learning methods to the task: first, we train a learner and use it to tag each line in a document with a tag in a pre-defined tag set; then we convert the best tag sequence into a span sequence. A span is a (start, end) pair, which indicates the beginning and ending line numbers of an IGT instance.

Among all the tagging schemes we experimented with (including the standard BIO tagging scheme), the following 5-tag scheme works the best on the development set: The five tags are BL (any blank line), O (outside IGT that is not a BL), B (the first line in an IGT), E (the last line in an IGT), I (inside an IGT that is not a B, E, or BL).

For machine learning, we use four types of features:

F_1**:** The words that appear on the current line. These are the features typically used in a text classification task.

F_2**:** Sixteen features that look at various cues for the presence of an IGT. For example, whether the line starts with a quotation, whether the line starts with an example number (e.g., (1)), and whether the line contains a large portion of hyphenated or non-English tokens.

F_3**:** In order to find good tag sequences, we include features for the tags of the previous two lines.

F_4**:** The same features as in F_2, but they are checked against the neighboring lines. For instance, if a feature f_5 in F_2 checks whether the current line contains a citation, f_5^{+1} checks whether the next line contains a citation.

After the lines in a document are tagged by the learner, we identify IGT instances by finding all the spans in the document that match the "$B\ [I \mid BL]^*\ E$" pattern; that is, the span starts with a B, ends with an E, and has zero or more I or BL in between.[4]

[4]Other heuristics for converting tag sequences to span sequences produce similar results.

Table 3: Data sets for the IGT detection experiments

	# files	# lines	# IGTs
Training data	41	39127	1573
Dev data	10	8932	447
Test data	10	14592	843

2.2.3 Experimental results

To evaluate the two detectors, we randomly selected 61 ODIN documents and manually marked the occurrence of IGT instances. The files were then split into training, development, and test sets, and the size of each set is shown in Table 3. The annotation speed was about four thousand lines per hour. Each file in the development and test sets was annotated independently by two annotators, and the inter-annotator agreement (f-score) on IGT boundary was 93.74% when using *exact match* (i.e., two spans match *iff* they are identical). When *partial match* (i.e., two spans match *iff* they overlap) was used, the f-score increased to 98.66%.

We used four machine learning algorithms implemented in Mallet (McCallum, 2002): decision tree, Naive Bayes, maximum entropy (MaxEnt), and conditional random field (CRF).[5] Table 4 shows the MaxEnt model's performance on the development set with different combinations of features: the highest f-score for exact match in each group is marked in boldface.[6] In addition to exact and partial match results, we also list the number of spans produced by the system (cf. the span number in the gold standard is 447) and the classification accuracy (i.e., the percent of lines receiving correct labels). The results for CRF are very similar to those for MaxEnt, and both outperform decision tree and Naive Bayes.

Several observations are in order. First, as expected, the machine learning approach outperforms the regular expression approach. Second, although F_2 contains only sixteen features, it works much better than F_1, which uses all the words occurring in the training data. Third, F_4 works much better than F_3 in capturing contextual information, mainly because F_4 allows the learner to take into account the information that appears on both the preceding lines and the succeeding lines.[7] Last, adding F_1 and F_3 to the $F_2 + F_4$ system offers a modest but statistically significant gain.

Table 5 shows the results on the test data. The performance of MaxEnt on this data set is slightly worse than on the development set mainly because the test set contains much more corrupted data (due to pdf-to-text conversion) than both the training and development sets.[8] Nevertheless, the machine learning approach outperforms the regex approach significantly, reducing the error rate by 52.3%. In addition, the partial match results are much better than exact match results, indicating that many span errors could be potentially fixed by postprocessing.

2.3 Manual review and language ID

About 45% of IGT instances in the current ODIN database were manually checked to verify IGT boundaries and to identify the language names of the target lines. Subsequently, we trained several language ID algorithms with the labeled data, and used them to label the remaining 55% of the IGT instances in ODIN automatically.

The language ID task in this context is different from a typical language ID task in several ways. First, the number of languages in IGT is close to a thousand or even more. In contrast, the amount of training data for many of the languages is very limited; for instance, hundreds of languages have less than 10 sentences, as shown in Table 1. Second, some languages in the test data might never occur in the training data, a problem that we shall call the *unknown language problem*. Third, the target sentences in IGT are very short (e.g., a few words), making the task more challenging. Fourth, for languages that do not use a latin-based writing system, the target sentences are often transliterated, making the character encoding scheme less informative. Last, the context, such as the language names occurring in the document, provides important cues for the language ID of IGT instances.

Given these properties, applying common language ID algorithms directly will not produce satisfactory results. For instance, Cavnar and Trenkle's N-gram-based algorithm yields an accuracy of as high as 99.8% when tested on newsgroup articles in eight languages (Cavnar and Trenkle, 1994).[9]

[5]For the first three methods, we implemented beam search to find the best tag sequences; and for CRF, we used features in F1, F2, and F4, as the model itself incorporates the information about previous tags already.

[6]F_4 is an extension of F_2, so every combination with F_4 should include F_2 as well. Also, F_3 should not be used alone. Therefore, Table 4 in fact lists all the possible feature combinations.

[7]The window for F_4 is set empirically to [-2,3]; that is, F_4 uses the information from the preceding two lines and the succeeding three lines.

[8]The corruption not only affects the target lines, but also the layout of IGT (e.g., the indentation of the three lines). As a result, features in F_2 and F_4 are not as effective as for the development set. Since the regex template approach uses fewer layout features, its performance is not affected as much.

[9]The accuracy ranges from 92.9% to 99.8% depending on the article length and a model parameter called *profile*

Table 4: Performance on the development set (the span number in the gold standard is 447)

Features	System span num	Classification accuracy	Exact match			Partial match		
			prec	recall	**fscore**	prec	recall	fscore
Regex templates	269	N/A	68.40	41.16	**51.40**	99.26	59.73	74.58
F_1	130	81.50	68.46	19.91	30.85	97.69	28.41	44.02
F_2	405	93.28	58.27	52.80	**55.40**	95.56	86.58	90.85
$F_1 + F_3$	180	80.26	61.67	24.83	35.40	81.11	32.66	46.57
$F_1 + F_2$	420	94.42	63.09	59.28	61.13	93.81	88.14	90.88
$F_2 + F_3$	339	92.68	75.81	57.49	65.39	93.21	70.69	80.40
$F_2 + F_4$	456	96.91	80.92	82.55	**81.73**	93.64	95.53	94.57
$F_1 + F_2 + F_3$	370	93.39	75.14	62.20	68.05	93.51	77.40	84.70
$F_1 + F_2 + F_4$	444	97.00	84.68	84.11	84.40	95.95	95.30	95.62
$F_2 + F_3 + F_4$	431	97.79	86.77	83.67	**85.19**	97.68	94.18	95.90
$F_1 + F_2 + F_3 + F_4$	431	98.00	90.02	86.80	**88.38**	97.22	93.74	95.44

Table 5: Performance on the test set (the span number in the gold standard is 843)

Features	System span num	Classification accuracy	Exact match			Partial match		
			prec	recall	**fscore**	prec	recall	fscore
Regex templates	587	N/A	74.95	52.19	**61.54**	98.64	68.68	80.98
F_2	719	92.45	57.02	48.64	52.50	94.02	80.19	86.56
$F_2 + F_4$	849	95.66	75.50	76.04	75.77	93.76	94.42	94.09
$F_2 + F_3 + F_4$	831	95.95	77.14	76.04	76.58	95.19	93.83	94.50
$F_1 + F_2 + F_3 + F_4$	830	96.83	82.29	81.02	**81.65**	96.51	95.02	95.76

However, when we ran the same algorithm on the IGT data, the accuracy was only 50.2%.[10] In contrast, a heuristic approach that predicts the language ID according to the language names occurring in the document yields an accuracy of 65.6%.

Because the language name associated with an IGT instance almost always appears somewhere in the document, we propose to treat the language ID task as a reference resolution problem, where IGT instances are the *mentions* and the language names appearing in the document are the *entities*. A language identifier simply needs to link the mentions to the entities, allowing us to apply any good resolution algorithms such as (Soon et al., 2001; Ng, 2005; Luo, 2007) and to provide an elegant solution to the unknown language problem. More detail on this approach will be reported elsewhere.

3 Using ODIN

We see ODIN being used in a number of different ways. In another study (Lewis and Xia, 2008), we demonstrated a method for using ODIN to discover interesting and computationally relevant typological features for hundreds of the world's languages automatically. In this section we present two more uses for ODIN's data: bootstrapping NLP tools (specifically taggers), and providing search over ODIN's data (as a kind of large-scale multi-lingual search).

length.

[10]The setting for our preliminary experiments is as follows: there are 10,415 IGT instances over 549 languages in the training data, and 3064 instances in the test data. The language names of about 12.2% of IGT instances in the test data never appear in the training data.

3.1 IGT for bootstrapping NLP tools

Since the target line in IGT data does not come with annotations (e.g., POS tags), it is first necessary to enrich it. Once enriched, the data can be used as a bootstrap for tools such as taggers.

3.1.1 Enriching IGT

In a previous study (Xia and Lewis, 2007), we proposed a three-step process to enrich IGT data: (1) parse the English translation with an English parser and convert English phrase structures (PS) into dependency structures (DS) with a head percolation table (Magerman, 1995), (2) align the target line and the English translation using the gloss line, and (3) project the syntactic structures (both PS and DS) from English onto the target line. For instance, given the IGT example in Ex (1), the enrichment algorithm will produce the word alignment in Figure 1 and the syntactic structures in Figure 2.

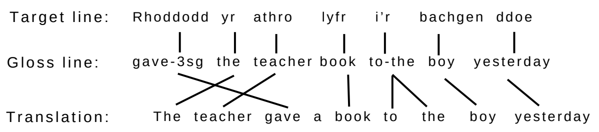
Target line: Rhoddodd yr athro lyfr i'r bachgen ddoe
Gloss line: gave-3sg the teacher book to-the boy yesterday
Translation: The teacher gave a book to the boy yesterday

Figure 1: Aligning the target line and the English translation with the help of the gloss line

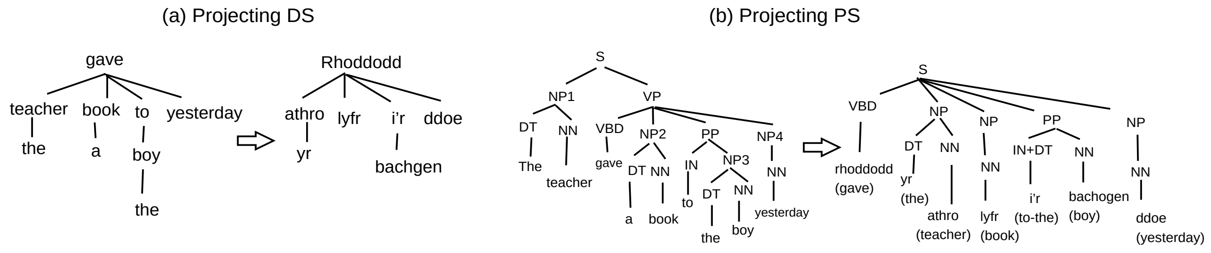

Figure 2: Projecting syntactic structure from English to the target language

We evaluated the algorithm on a small set of 538 IGT instances for several languages. On average, the accuracy of the English DS (i.e., the percentage of correct dependency links in the DS) is 93.48%; the f-score of the word alignment links between the translation and target lines is 94.03%, and the accuracy of the target DS produced by the projection algorithm is 81.45%. When we replace the automatically generated English DS and word alignment with the ones in the gold standard, the accuracy of target DS increases significantly, from 81.45% to 90.64%. The details on the algorithms and the experiments can be found in (Xia and Lewis, 2007).

3.1.2 Bootstrapping NLP tools

The enriched data produced by the projection algorithms contains (1) the English DS and PS produced by an English parser, (2) the word alignment among the three parts of IGT data, and (3) the target DS and PS produced by the projection algorithm. From the enriched data, various kinds of information can be extracted. For instance, the target syntactic structures form small monolingual treebanks, from which grammars in various formalisms can be extracted (e.g., (Charniak, 1996)). The English and target syntactic structures form parallel treebanks, from which transfer rules and translation lexicon can be extracted and used for machine translation (e.g., (Meyers et al., 2000; Menezes, 2002; Xia and McCord, 2004)).

There are many ways of using the enriched data to bootstrap NLP tools. Suppose we want to build a POS tagger. Previous studies on unsupervised POS tagging can be divided into several categories according to the kind of information available to the learner. The first category (e.g., (Kupiec, 1992; Merialdo, 1994; Banko and Moore, 2004; Wang and Schuurmans, 2005)) assumes there is a lexicon that lists the allowable tags for each word in the text. The common approach is to use the lexicon to initialize the emission probability in a Hidden Markov Model (HMM), and run the Baum-Welch algorithm (Baum et al., 1970) on a large amount of unlabeled data to re-estimate transition and emission probability. The second category uses unlabeled data only (e.g., (Schütze, 1995; Clark, 2003; Biemann, 2006; Dasgupta and Ng, 2007)). The idea is to cluster words based on morphological and/or distributional cues. Haghighi and Klein (2006) showed that adding a small set of prototypes to the unlabeled data can improve tagging accuracy significantly.

The tagged target lines in the enriched IGT data can be incorporated in each category of work mentioned above. For instance, the frequency collected from the data can be used to bias initial transition and emission probabilities in an HMM model; the tagged words in IGT can be used to label the resulting clusters produced by the word clustering approach; the frequent and unambiguous words in the target lines can serve as prototype examples in the prototype-driven approach (Haghighi and Klein, 2006). Finally, we can apply semi-supervised learning algorithms (e.g., self-training (Yarowsky, 1995), co-training (Blum and Mitchell, 1998), and transductive support vector machines (Vapnik, 1998)), using the tagged sentences as seeds.

3.2 Search

One focus of ODIN is and has always been search: how can linguists find the data that they are interested in and how can the data be encoded in such a way as to accommodate the variety of queries that a linguist might ask. We currently allow four types of search queries: search by language name and code, search by language family, search by concept/gram, and search by linguistic constructions. The first allows the user to specify a language name or ISO code to search for, and allows the user to view documents that contain instances of IGT in that language, as well as the instances themselves. The second allows the user to specify a language family (families as specified in the Ethnologue), and returns similar results, except grouped by language. The third allows the user to select from a list of known grams, all of which have been mapped to a conceptual space used by linguists (the GOLD ontology, (Farrar and Langendoen, 2003)).[11]

The final query type, the Construction Search is the most powerful and most innovative of the query facilities currently provided by ODIN. Rather than limiting search to just the content and markup natively contained within IGT, Construction Search searches over *enriched* content. For instance, a search for relative clauses can look for either the POS tag sequences that contain a noun followed by an appropriate relativizer, or the parse trees that contain an NP node with an NP child and a clause child. Currently, 15 construction queries have been implemented, with some 40 additional queries being evaluated and built. Note that currently construction queries are performed on the English translation, not on the target language data. As syntactic projection becomes more reliable, we will allow construction queries on the target language data and even queries on both the English and the target (e.g., for comparative linguistic analyses). For example, a query could be something like *Find examples where the target line uses imperfective aspect and is in active voice and the English translation uses passive voice.*

4 Conclusion and Future Directions

In this paper, we introduce Interlinear Glossed Text (IGT), a data type that has been rarely tapped by the NLP community, and describe the process of creating ODIN, a database of IGT data. We show that using machine learning methods can significantly improve the performance of IGT detection. We then demonstrate how IGT instances can be enriched and discuss several ways of using enriched data to bootstrap NLP tools such as POS taggers. Finally, we review the four types of linguistic search that are currently implemented in ODIN. All of the above show the value of ODIN as a resource for both NLP researchers and linguists. In the future, we plan to improve the IGT detection and language ID algorithms and will apply them to all the crawled documents. We expect the size of ODIN to grow dramatically. We also plan to use the enriched data to bootstrap taggers and parsers, starting with the ideas outlined in Section 3.1.2.

Acknowledgements This work has been supported, in part, by the Royalty Research Fund at the University of Washington. We would also like to thank Dan Jinguji for providing the preliminary results on language ID expriments, and three anonymous reviewers for their valuable comments.

[11]Most gram-to-concept mapping has been done by hand. We are currently exploring methods to use machine learning to enhance our ability to identify and map additional unknown grams (to be discussed elsewhere).

References

John Frederick Bailyn. 2001. Inversion, dislocation and optionality in Russian. In Gerhild Zybatow, editor, *Current Issues in Formal Slavic Linguistics*.

Michele Banko and Robert C. Moore. 2004. Part of Speech Tagging in Context. In *Proc. of the 20th International Conference on Computational Linguistics (Coling 2004)*, pages 556–561, Geneva, Switzerland.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statistics*, 41(1):164–171.

Balthasar Bickel, Bernard Comrie, and Martin Haspelmath. 2004. The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses (revised version). Technical report, Max Planck Institute for Evolutionary Anthropology and the Department of Linguistics of the University of Leipzig.

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, pages 7–12, Sydney, Australia, July.

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proc. of the Workshop on Computational Learning Theory (COLT-1998)*.

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.

Eugene Charniak. 1996. Treebank Grammars. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-1996)*.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proc. of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2003)*.

Sajib Dasgupta and Vincent Ng. 2007. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language*

Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 218–227.

Scott Farrar and D. Terence Langendoen. 2003. A linguistic ontology for the Semantic Web. *GLOT International*, 7(3):97–100.

Raymond G. Gordon, editor. 2005. *Ethnologue: Languages of the World*. SIL International, Dallas, TX, fifteenth edition.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT/NAACL 2006)*, pages 320–327, New York City, USA.

Baden Hughes, Steven Bird, and Cathy Bow. 2003. Interlinear text facilities. In *E-MELD 2003*, Michigan State University.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the ACL*, Philadelphia, Pennsylvania.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6.

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.

David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, Cambridge, Massachusetts, USA.

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

Arul Menezes. 2002. Better contextual translation using machine learning. In *Proc. of the 5th conference of the Association for Machine Translation in the Americas (AMTA 2002)*.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2).

Adam Meyers, Michiko Kosaka, and Ralph Grishman. 2000. Chart-based transfer rule application in machine translation. In *Proc. of the 18th International Conference on Computational Linguistics (COLING 2000)*.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 157–164, Ann Arbor, Michigan.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proc. of the EACL*, pages 141–148.

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).

V. Vapnik. 1998. *Statistical learning theory*. Wiley-Interscience.

Qin Iris Wang and Dale Schuurmans. 2005. Improved Estimation for Unsupervised Part-of-Speech Tagging. In *Proc. of IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE 2005)*.

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 Michael McCord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proc. of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

David Yarowsky and Grace Ngai. 2001. Inducing Multilingual POS Taggers and NP Bracketers via Robust Projection across Aligned Corpora. In *Proc. of the 2001 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-2001)*, pages 200–207.

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