Using Diverse Information Sources to Retrieve Samples of Low-Density Languages

Andrew MacKinlay
Department of Computer Science and Software Engineering
University of Melbourne
Parkville VIC 3051 Australia

Abstract

Language samples are useful as an object of study for a diverse range of people. Samples of low-density languages in particular are often valuable in their own right, yet it is these samples which are most difficult to locate, especially in a vast repository of information such as the World Wide Web. We identify here some shortcomings to the more obvious approaches to locating such samples and present an alternative technique based on a search query using publicly available wordlists augmented with geospatial evidence, and show that the technique is successful for a number of languages.

1 Introduction

The utility of language samples to anyone with an interest in a given language is obvious: they can be valuable to linguists, language technologists and speakers of the language, among others. The World Wide Web (WWW) is vast repository of information with potentially large numbers of samples[1] of many languages, but locating these samples reliably is a non-trivial task. While language-specific search tools on search engines such as Google are useful for the languages they cover, for the vast majority of languages which have fewer speakers and a smaller online presence (low-density languages) they provide no information, and it is these lesser-known languages for which language resources are likely to be most useful to language researchers.

There are existing online resources which provide varying degrees of coverage for a large number of languages but the amount of data stored in these for even the best covered languages is generally quite limited. It is likely that there is a large number of documents on the WWW which for a variety of reasons have not made their way into these repositories.

When we expand our focus to include the entire Web, it essentially becomes a problem of language classification, but instead of looking at just a handful of documents as is often the norm in language classification tasks, the number of potential matches is all of the ∼8 billion documents on the WWW indexed by Google. This means the first stage of the classification will necessarily be a search to vastly cut down the number of potential matches.

There are of course other alternatives to finding web-based data for these low-density languages. The most obvious choice of a WWW query using the name of a language works in some cases, but there are of course complications. Languages often have more than one name, such as Adamawe Fulfulde (Niger-Congo, West Africa) which has roughly 33 distinct names including vastly different names such as Biira, Gapelta and Taareyo[2], and, language names can co-incide with words in high-density languages, such as Even (Altaic, Russia). Of course, webpages may not even mention the language in which they are written. The sparseness of these languages means that is crucial to have more than one approach to finding data in order to maximise recall.

The focus here is to investigate an alternative method for classifying and obtaining webpages with a substantial proportion of content written in a given low-density language based on a list of words known to be in the language, and show the additional classification steps required (specifically, analysis of locations mentioned in each document, and a word-unigram classifier) to produce reasonable precision in the documents returned. We have deliberately avoided alternative search methods such as the language-name based web searches mentioned

[1] In the context of this paper, we define a sample of a particular language as a webpage with a substantial proportion of content written in the language

[2] Source: *http://www.rosettaproject.org/*

Proceedings of the Australasian Language Technology Workshop 2005, pages 64–70,
Sydney, Australia, December 2005.

above to focus on the utility of a particular technique.

Section 2 describes the the components of the technique we use, section 3 shows a feasibility test and results for some low-density languages and in section 4 we discuss successes and shortcomings.

2 Method

2.1 Word Unigram-Based Querying and Classification

There are a number of features of a document which can be used to identify its language; our focus here on web-based data retrieval and the word-based nature of most search engines leads naturally to using the presence of particular word-unigrams as our first stage discriminator. For example, a document containing the word 'the', even with uniform prior probabilities for all languages, has a very strong chance of being at least partly in English. For many languages, we can pick a small set of words which will be entirely or partially included in any document in the target language.

Obviously a prerequisite for word-based language sample retrieval is a word list to act as a seed for the search process. For low-density languages, locating such a word-list can be non-trivial. Initially, when we are simply trying to identify candidate documents, the words need not be the most likely words of the language, although if such a list was available it would provide higher initial precision and recall. Assuming that we know very little about the language, a list of words known to be in the language will suffice. Such lists are already available online for some 1,400 languages at the Rosetta Project website. The website contains 'Swadesh lists' for many languages comprised of the translations of a fixed set of 100 or 200 English word-forms.

These Swadesh lists are the basis of the first step of the heuristic described here. To find potential language samples, we obtain a machine-readable version of the Swadesh list (manually converted from the online version due to the presence of idiosyncrasies in individual lists). All of the words are potential search queries; before performing the query, the first stage of filtering takes place, in which the words are compared against a list of stopwords from 17 high-density languages[3] and words that co-incide with these stopwords are removed to avoid too many erroneous matches. In stage 1 of the classification process, the remaining words are passed one-by-one as search queries using the Google Web API to give a list of possible language samples, and the ten top-ranked pages for that search term (this being the maximum allowed by the API) are retrieved and all non plain-text elements removed.

One problem with this search technique is caused by the the Google's page-ranking technique. Samples of low-density languages are unlikely to be among the ten highest ranked documents returned using the Web API, since these ranks are determined by a version of the PageRank algorithm (Brin and Page, 1998) which assigns a ranking based on the number of hypertext links to a webpage. This is probably the largest schortcoming of our technique. Ideally, it would be preferable to use a search engine which ranks pages based on the well-known TF-IDF metric used in information retrieval, however it is also important that the search engine has good coverage and an interface for automated querying. We were not able to locate a search engine which definitely fulfilled all of these criteria, not least because the details of the underlying ranking algorithms are generally trade secrets so determining the extent of TF-IDF usage is almost impossible. These considerations encouraged persisting with the default choice of Google.

The only way we found to alleviate the problem is based on the observation that if a query term produces just a few search results, there is a reasonable chance one of our target documents occurs in the top ten, but as the number of results becomes larger, the likelihood of this decreases. Therefore, if we group retrieved pages by the query term which produced them, and rank them in increasing order according to the number of hits produced by the query term, the most likely matches will be ordered first. This information can then be used as a heuristic to guide the manual refinement stage discussed below. Note that, while the motivation differs somewhat, in practice this is essentially the same as the inverse document frequency in the well-known TF-IDF metric used in information retrieval, since we are effectively weighting each term with a function which decreases monotonically as its document frequency increases.

[3]These are from some of the most widely spoken Roman-script languages including most crucially English, French, German and Spanish, the four most common languages for web content according to Google

2.2 Adding Geospatial Evidence

After fetching the query matches, the set of possible samples of the target language contains a high proportion of false positives, and in line with the assumption that we have very little information on the language, we assume we do not have a language classifier trained to distinguish between the documents. Clearly another layer of refinement is necessary, but it cannot depend on knowledge of the language.

For this, we use another piece of supporting evidence for the language of a document: the geospatial locations mentioned in it. Webpages frequently include local references and we would expect that, since lower-density languages tend to be confined to one geographic area, the locations mentioned in such documents will show similar geographic restrictions. Thus, the presence in a document of a reference to a location which we know to be in the area where the language in question is spoken provides reasonable evidence for the language of a document – not indisputable evidence, but certainly a sign that the document should be considered quite a likely candidate. This classification method will be denoted LLC for 'location lookup classifier'

To obviate the need for a location tagger trained specifically for the language, we take what amounts to a 'brute force' approach for stage 2 of the classification process. The target locations are taken to be any location in the countries where the language, according to the Rosetta Project site, is spoken. The list of cities and region in each country is easily obtainable from the UN LoCODES database.[4] The appropriate locations are extracted from the file, and we perform an uninformed linear search over each putative sample for any of the location names. Any documents which contain an appropriate location reference are tagged as such.

2.3 Further Refinement and Manual Analysis

One final layer of refinement is applied to the data to vastly cut down the manual analysis work. An existing language classifier trained on the high-density languages, such as van Noord's implementation TextCat[5] of the algorithm in Cavnar and Trenkle (1994), is used to test any documents which were candidate matches according to the previous two criteria. If the classifier can unambiguously determine the language of the document, it is almost certainly a sample of that language rather than of the target language. However, if there is any ambiguity, the document is deemed worthy of further examination by a human.

Applying the filters described so far cuts the number of potential samples from ∼4 billion to a much more manageable number. From the results obtained, the number of potential matches at this point is between zero and 600. At this point a manual analysis to find at least one matching document is quite feasible by looking at the set of search results ranked in order of the number of query-term matches.

There is an obvious difficulty of determining whether a document is in a language with which we are not familiar. In practice there are a large number of hints we can use: the URL or the presence of the name of the language, or the aforementioned lexical resources. At this point most of the potential matches are pathological webpages containing very few words, genuine incorrect languages that TextCat failed to recognise unambiguously, documents containing more than one language, samples of similar languages, and genuine samples of the target language. Of these only the samples of similar languages are any trouble to distinguish from genuine samples, and bilingual documents can be set aside for reference.

Once at least one potential sample is identified, we use it to train some sort of probabilistic classifier to identify other samples either in the remainder of the possible matches from stage 2 if the number was large, or in the residue of documents retrieved in stage 1 but discarded due to the absence of an appropriate location. We are simply trying to make the binary distinction between genuine and spurious samples, so any legitimate Bayesian classifier at this stage would suffice, as long as it could be trained on a single document and achieve reasonable performance. As noted by Dunning (1994) a word unigram-based classifier should produce acceptable performance in this case, given that we are only interested in samples of reasonable length, and these are the samples on which such a classifier can perform well. Additionally it has the advantage of being able to easily produce a list of the most likely words.

We simply obtain a word unigram frequency distribution from one document and produce a smoothed probability distribution from it using Witten-Bell discounting (Jurafsky and Martin,

[4] *http://www.unece.org/locode*
[5] *http://odur.let.rug.nl/~vannoord/TextCat/*

2000). Then, simplifying the method of Dunning (1994), we use the distibution to determine the logarithm of the probability of each test document according to the model as the sum of log probabilities of each word according to the model. This is then divided by the number of tokens in the document to give a normalised mean log probability per token for each document according to the trained model, which is essentially the cross entropy of each test document.

Since each document needs only to be classified as a positive or negative sample, they are classified as genuine or spurious on the basis of their cross entropy relative to a threshold. Any estimate of the appropriate cutoff value amounts to making arbitrary assumptions about the distribution. Instead we arbitrarily choose a cutoff point between the ranges of the minimum and maximum entropy values, which is empirically determined. In practice the value is used to rank the documents and it is a simple task for a human annotator to determine a reasonable cutoff in each case.

At this point we hopefully have a reasonably sized list of language samples. If this is the case, the same classifier is then trained on all of the documents and the most probable words are output. The number here can vary – ideally we would like 5-7 common words (probably stopwords of that language) after the stop-words from the high-density languages have been removed. This is a manageable number that can them form the basis of a new round of Google queries, but since we have the most common words of the language a query on the words in combination can be almost guaranteed to include any valid samples of the target language (albeit possibly not in the top 10), but is a more restrictive query, thus avoiding large numbers of spurious matches. So at this stage we perform a Google query on every possible combination of one or more of the common words. Since we have restricted the number of words, this is quite manageable: $2^n - 1$ for n words, giving 127 queries for 7 words.

The result of this query is new set of pages which should be even more likely to be language samples than in the previous iteration of the process. The previous refinement stages can then be run again in a similar fashion, and at this stage assuming no bottlenecks in the process, we have a set of sample documents as well a set of stopwords which can be used for further queries.

3 Results

3.1 Feasibility testing

The initial phase of testing was determining the feasibility of the method. To achieve this, it was necessary to choose languages for which there was a clear easily accessible gold-standard. The obvious choice was to use languages identifiable by TextCat. We chose four arbitrary medium density languages: Finnish, Polish, Rumanian and Icelandic. The method obviously differs slightly from that outlined above – we check to see whether TextCat exactly guesses the target language, rather than whether it is unsure of the language.

We evaluated the stages of the technique in several ways. In table 1 we evaluate the precision at each stage of the refinement process. It is clear that the technique of a Google query on the terms in a language works reasonably well in producing a reasonable proportion of genuine samples, and also that precisions ranging from acceptable to very impressive can be obtained through the combination of a web-query and geospatial location lookup. In Table 2 we show the precision and recall for the location lookup method relative to the refined set of language samples retrieved from the web-query. We included this to show that, working from the sample space of the URLs fetched in stage 1, the location-lookup method retrieves a reasonable proportion of the positive samples from all of the possible samples it receives. Note that the figures for precision are identical in tables 1 and 2 since we are looking at the same set of classifications each time.

It is also necessary to evaluate the validity of the technique of using word unigram cross entropy estimation and picking an arbitrary cutoff point. We selected a random correctly classified sample as training data, and for each of the genuine and spurious matches from the documents classified positively after a location lookup and web-query we calculated the cross entropy relative to this training data. This enabled evaluation of the precision and recall we would have achieved by selecting a particular cutoff value for the cross entropy. If we have minimum and maximum cross entropy values H_{min} and H_{max} respectively, and an absolute cross-entropy cutoff H_{thresh}, the factor k in ta-

Language	Icelandic	Finnish	Rumanian	Polish
Documents Retrieved in web query	1314	2090	1765	1921
True Samples Retrieved	438	912	385	673
Precision of Rosetta-based query	33.3%	43.6%	21.8%	35.0%
Positive Classifications by Location Lookup	153	261	340	481
Genuine Samples in Positives	146	176	267	389
Precision of Location Lookup/web query	95.4%	78.5%	67.4%	80.9%

Table 1: Precision of Rosetta-seeded web query and location lookup/web query combined

Language	Icelandic	Finnish	Rumanian	Polish
Genuine Samples Previously Retrieved	438	912	385	673
Positive Classifications	153	261	340	481
Genuine Samples in Positives	146	176	267	389
Recall	33.3%	19.3%	69.4%	57.8%
Precision	95.4%	78.5%	67.4%	80.9%
F-Score ($\alpha = 0.5$)	49.4%	30.0%	73.7%	67.4%

Table 2: Precision, Recall and F-Score of location lookup relative to retrieved query matches

ble 3 is calculated as:

$$k = \frac{H_{thresh} - H_{min}}{H_{max} - H_{min}}$$

Here the exemplifying language was Romanian; other languages give similar results. Clearly a high F-score can be obtained by careful selection of the cutoff point.

3.2 Genuine Low Density Languages

These results make it clear that the technique is at least worth pursuing. While we would only expect the results to get worse when we move into lower density languages, there is certainly enough accuracy that we can reasonably expect the technique to work in some cases.

In Table 4 the number of documents in each classification stage are shown with 'LLC +' denoting the number of documents positively classified by LLC and 'TC ?' denoting the number of documents ambiguous or unknown according to TextCat (which, as we have explained above, are the documents which are possible samples of the language). The results were obtained by checking whether there was at least one potential training document, and selecting an appropriate document from this set as training data for classifying genuine samples, then using all sufficiently uniform genuine samples as training for the purpose of extracting the most frequent words. Five or six infrequent words which did not coincide with stopwords for high-density languages were then selected for the second iteration.

As mentioned above, the number of genuine samples here was determined by manually examining documents in order of cross-entropy relative to a suitable training document and determining a cutoff point. Note that uncertainty figures are due to the presence of non-canonical documents in the word-list such as bilingual documents often containing little observable structure (such as many retrieved from a Pampangan web forum) and very short documents for which the language was unclear. In the second iteration, some of the documents retrieved were repetitions of pages already retrieved; the column labelled "New" only counts those that were unique to the second iteration.

While from these figures it may appear that the location lookup in the second iteration achieved very little, some additional experiments tended to indicate that no extra samples wore omitted by this technique.

4 Discussion and Further Work

While there are some encouraging results, certain aspects of the technique were not as useful as we had predicted. One aspect worth commenting on is that the crude location lookup was often less useful than we expected from our experiments with medium density languages. In many cases (notably the two where the most samples were retrieved for the genuine low-density languages), this stage of refinement re-

Cutoff k	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Samples Included	44	82	128	150	173	195	216	230	251	259
Precision	100.0	100.0	100.0	100.0	96.0	88.3	80.2	75.8	69.4	67.6
Recall	25.7	47.4	73.7	86.3	95.4	98.9	99.4	100.0	100.0	100.0
F-Score ($\alpha = 0.5$)	40.9	64.3	84.9	92.6	95.7	93.3	88.8	86.2	82.0	80.6

Table 3: Precision, Recall and F-Score (%) over Romanian documents for different cross entropy cutoffs

Language	1st Iteration				2nd Iteration				
	URLs	LLC +	TC ?	Genuine	URLs	LLC +	TC ?	Genuine	New
Mikir	1557	104	31	1	64	11	5	2	0
Fasu	1494	106	32	0	–	–	–	–	–
Hixkaryana	620	580	181	0	–	–	–	–	–
Hmong Daw	841	830	299	120±10	117	116	60	30±5	30±5
Mopon Maya	769	26	4	0	–	–	–	–	–
Pampangan	1839	1797	621	22 ±8	110	109	45	8	5
Warao	998	27	6	0	–	–	–	–	–
Fasu	1046	84	25	0	–	–	–	–	–
Tulu	852	80	18	0	–	–	–	–	–
Quiché	625	29	7	3	97	5	5	5	3

Table 4: Number of Documents at Each Stage of Classification

moved less than 10% of the possible samples, even though in subsequent stages many were positively identified as samples of other languages by TextCat and consequently removed. It would be possible to refine this aspect by applying the TF-IDF metric to the LLC method; this is an area for furher work.

The technique could also be augmented with other search techniques such as as the obvious web query for the name(s) of the language. Alternatively, one or more stages could be used in a more comprehensive language retrieval system.

4.1 Shortcomings and Solutions

While there is some promise to the technique, here we note a number of areas of weakness of this work-in-progress, with potential solutions where they can be identified:

1. There is no obvious starting point when there is no Rosetta wordlist or when the wordlist exists but the orthography is non-standard, either because no standard orthography exists or because the contributor failed to use it. In cases like these obviously another wordlist could be substituted, or if a training document is available we could render the first stage of the iteration unnecessary. While documents obtained in an *ad hoc* fashion could be used for this purpose, there are freely accessible repositories of possible training documents. The UN Declaration of Human Rights has been translated in over 300 languages [6], and while this does not approach the coverage of the Rosetta lists, the UN documents avoid the aforementioned problems with orthography in these lists.

2. The Google page-rank problem already outlined is an obvious issue. In the first iteration many possible matches are missed. We might expect that the second iteration would obtain some of these missed documents with its more precise queries, but the top-ranked among these tend to be the same documents previously retrieved, meaning that some proportion of the top-10 list will be wasted, and the number of douments retrieved is still limited. We have already discussed in Section 2.1 the desirability of a search engines with different underlying algorithms, and the difficulty of locating such a search engine. It is difficult to tell whether the tests in table 4 which produced no results did so because of the page-rank problem or because such a small amount of data is available.

3. There are of course other possible charac-

[6] Source: *http://www.unhchr.ch/udhr/navigate/alpha.htm*

teristics of the language which could render them unsuitable for this method. It will work best on isolating languages, as will probably be the case for any word-based search. For languages with large amounts of morphology, such as polysynthetic languages in the extreme case, the existence of large numbers of wordforms for any lemma means that this technique is less likely to work. Nonetheless, it is clear that technique works effectively on at least one agglutinative language (Finnish) and one inflectional one (Icelandic). Indeed, this possibly explains some of the variation between results. One way to alleviate the problem might be to replace word unigrams in the classifier with character n-grams, which are a well-studied method for language identification described by Dunning (1994), or even to use the documents as training data for a TextCat-like classifier (which is also n-gram based albeit with no reference to cross-entropy). The relative performance of such a technique would need to be evalutated empirically.

5 Conclusion

We have presented an alternative technique for locating samples of low-density languages based on web queries using publicly available wordlists, and refining the results using geospatial evidence and existing language identification algorithms. Despite the novelty of the technique presented and the crude underlying methods in certain parts, we have shown it can in certain cases be an effective means of retrieving samples of low-density languages, free of some of the shortcomings of more obvious techniques.

6 Acknowledgements

We thank Steven Bird and Baden Hughes for guidance and formulating the original idea, and the anonymous reviewers for many helpful comments which have contributed to this final paper.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

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.

Ted Dunning. 1994. Statistical identification of language. Technical Report MCCS 940-273, Computing Research Laboratory, New Mexico State University.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing.* Prentice Hall Series in Artificial Intelligence. Prentice-Hall.