Language Identification for Creating Language-Specific Twitter Collections

Shane Bergsma† Paul McNamee†,♮ Mossaab Bagdouri⋆ Clayton Fink♮ Theresa Wilson†
†Human Language Technology Center of Excellence, Johns Hopkins University
‡Johns Hopkins University Applied Physics Laboratory, Laurel, MD
⋆Department of Computer Science, University of Maryland, College Park, MD
sbergsma@jhu.edu, mcnamee@jhu.edu, mossaab@umd.edu, clayton.fink@jhuapl.edu, taw@jhu.edu

Abstract

Social media services such as Twitter offer an immense volume of real-world linguistic data. We explore the use of Twitter to obtain authentic user-generated text in low-resource languages such as Nepali, Urdu, and Ukrainian. Automatic language identification (LID) can be used to extract language-specific data from Twitter, but it is unclear how well LID performs on short, informal texts in low-resource languages. We address this question by annotating and releasing a large collection of tweets in nine languages, focusing on confusable languages using the Cyrillic, Arabic, and Devanagari scripts. This is the first publicly-available collection of LID-annotated tweets in non-Latin scripts, and should become a standard evaluation set for LID systems. We also advance the state-of-the-art by evaluating new, highly-accurate LID systems, trained both on our new corpus and on standard materials only. Both types of systems achieve a huge performance improvement over the existing state-of-the-art, correctly classifying around 98% of our gold standard tweets. We provide a detailed analysis showing how the accuracy of our systems vary along certain dimensions, such as the tweet-length and the amount of in- and out-of-domain training data.

1 Introduction

Twitter is an online social-networking service that lets users send and receive short texts called tweets. Twitter is enormously popular; more than 50 million users log in daily and billions of tweets are sent each month.† Tweets are publicly-available by de-

fault and thus provide an enormous and growing free resource of authentic, unedited text by ordinary people. Researchers have used Twitter to study how human language varies by time zone (Kiciman, 2010), census area (Eisenstein et al., 2011), gender (Burger et al., 2011), and ethnicity (Fink et al., 2012). Twitter also provides a wealth of user dialog, and a variety of dialog acts have been observed (Ritter et al., 2010) and predicted (Ritter et al., 2011).

Of course, working with Twitter is not all roses and rainbows. Twitter is a difficult domain because unlike, for example, news articles, tweets are short (limited to 140 characters), vary widely in style, and contain many spelling and grammatical errors. Moreover, unlike articles written by a particular news organization, a corpus constructed from Twitter will contain tweets in many different languages. This latter point is particularly troubling because the majority of language-processing technology is predicated on knowing which language is being processed. We are pursuing a long-term effort to build social media collections in a variety of low-resource languages, and we need robust language identification (LID) technology. While LID is often viewed as a solved problem (McNamee, 2005), recent research has shown that LID can be made arbitrarily difficult by choosing domains with (a) informal writing, (b) lots of languages to choose from, (c) very short texts, and (d) unbalanced data (Hughes et al., 2006; Baldwin and Lui, 2010). Twitter exhibits all of these properties. While the problem of LID on Twitter has been considered previously (Tromp and Pechenizkiy, 2011; Carter et al., 2013), these studies have only targeted five or six western European languages, and not the diversity of languages and writing systems that we would like to process.
Our main contribution is the release of a large collection of tweets in nine languages using the Cyrillic, Arabic, and Devanagari alphabets. We test different methods for obtaining tweets in a given target language (§2). We then use an online crowdsourcing platform to have these tweets annotated by fluent speakers of that language (§3). We generate over 18,000 triple-consensus tweets, providing the first publicly-available collection of LID-annotated tweets in non-Latin scripts. The annotated corpus is available online at: http://apl.jhu.edu/~paulmac/lid.html. We anticipate our multilingual Twitter collection becoming a standard evaluation set for LID systems.

We also implement two LID approaches and evaluate these approaches against state-of-the-art competitors. §4.1 describes a discriminative classifier that leverages both the tweet text and the tweet metadata (such as the user name, location, and landing pages for shortened URLs). §4.2 describes an efficient tool based on compression language models. Both types of systems achieve a huge improvement over existing state-of-the-art approaches, including the Google Compact Language Detector (part of the Chrome browser), and a recent LID system from Lui and Baldwin (2011). Finally, we provide further analysis of our systems in this unique domain, showing how accuracy varies with the tweet-length and the amount of in-domain and out-of-domain training data. In addition to the datasets, we are releasing our compression language model tool for public use.

2 Acquiring Language-Specific Tweets

We use two strategies to collect tweets in specific languages: (§2.1) we collect tweets by users who follow language-specific Twitter sources, and (§2.2) we use the Twitter API to collect tweets from users who are likely to speak the target language.

2.1 Followers of Language-Specific Sources

Our first method is called the Sources method and involves a three-stage process. First, Twitter sources for the target language are manually identified. Sources are Twitter users or feeds who: (a) tweet in the target language, (b) have a large number of followers, and (c) act as hubs (i.e., have a high followers-to-following ratio). Twitter sources are typically news or media outlets (e.g. BBC News), celebrities, politicians, governmental organizations, but they may just be prominent bloggers or tweeters.

Once sources are identified, we use the Twitter API (dev.twitter.com) to query each source for its list of followers. We then query the user data for the followers in batches of 100 tweets. For users whose data is public, a wealth of information is returned, including the total number of tweets and their most recent tweet. For users who had tweeted above a minimum number of times, and whose most-recent-tweet tweet was in the character set for the target language, we obtained their most recent 100-200 tweets and added them to our collection.2

While we have used the above approach to acquire data in a number of different languages, for the purposes of our annotated corpus (§3), we select the subsets of users who exclusively follow sources in one of our nine target languages (Table 1). We also filter tweets that do not contain at least one character in the target’s corresponding writing system (we plan to address romanized tweets in future work).

2.2 Direct Twitter-API Collection

While we are most interested in users who follow news articles, we also tested other methods for obtaining language-specific tweets. First, we used the Twitter API to collect tweets from locations where we expected to get some number of tweets in the target language. We call this method the Twit-API collection method. To geolocate our tweets, the Twitter API’s geotag method allowed us to collect tweets within a specified radius of a given set of coordinates in latitude and longitude. To gather a sample of tweets in our target languages, we queried for tweets from cities with populations of at least 200,000 where speakers of the target language are prominent (e.g., Karachi, Pakistan for Urdu; Tehran, Iran for Farsi; etc.). We collected tweets within a radius of 25 miles of the geocoordinates. We also used the Search API to persistently poll for tweets in future work).

Tromp and Pechenizkiy (2011) also manually identified language-specific Twitter feeds, but they use tweets from these sources directly as gold standard data, while we target the users who simply follow such sources. We expect our approach to obtain more-authentic and less-edited user language.
identification code returned by the API for each
tweet; we filter all our geolocated Urdu tweets that
are not marked as Urdu.

We also obtained tweets through an information-
retrieval approach that has been used elsewhere for
creating minority language corpora (Ghani et al.,
2001). We computed the 25 most frequent unique
words in a number of different languages (that is,
words that do not occur in the vocabularies of other
languages). Unfortunately, we found no way to en-
force that the Twitter API return only tweets con-
taining one or more of our search terms (e.g., re-
turned tweets for Urdu were often in Arabic and did
not contain our Urdu search terms). There is a lack
of documentation on what characters are supported
by the search API; it could be that the API cannot
handle certain of our terms. We thus leave further
investigation of this method for future work.

3 Annotating Tweets by Language

The general LID task is to take as input some piece
of text, and to produce as output a prediction of what
language the text is written in. Our annotation and
prediction systems operate at the level of individual
tweets. An alternative would have been to assume
that each user only tweets in a single language, and
to make predictions on an aggregation of multiple
tweets. We operate on individual tweets mainly be-
cause (A) we would like to quantify how often users
switch between languages and (B) we are also inter-
ested in domains and cases where only tweet-sized
amounts of text are available. When we do have
multiple tweets per user, we can always aggregate
the scores on individual predictions (§6 has some ex-
perimental results using prediction aggregation).

Our human annotation therefore also focuses on
validating the language of individual tweets. Tweets
verified by three independent annotators are ac-
cepted into our final gold-standard data.

3.1 Amazon Mechanical Turk

To access annotators with fluency in each language,
we crowdsourced the annotation using Amazon Me-
chanical Turk (mturk.com). AMT is an online la-
bor marketplace that allows requesters to post tasks
for completion by paid human workers. Crowd-
sourcing via AMT has been shown to provide high-
quality data for a variety of NLP tasks (Snow et al.,
2008; Callison-Burch and Dredze, 2010), including
multilingual annotation efforts in translation (Zaidan
and Callison-Burch, 2011b), dialect identification
(Zaidan and Callison-Burch, 2011a), and building
bilingual lexicons (Irvine and Klementiev, 2010).

3.2 Annotation Task

From the tweets obtained in §2, we took a random
sample in each target language, and posted these
tweets for annotation on AMT. Each tweet in the
sample was assigned to a particular AMT job; each
job comprised the annotation of 20 tweets. The job
description requested workers that are fluent in the
target language and gave an example of valid and
invalid tweets in that language. The job instructions
asked workers to mark whether each tweet was writ-
ten for speakers of the target language. If the tweet
combines multiple languages, workers were asked
to mark as the target language if “most of the text is
in [that language] excluding URLs, hash-tags, etc.”
Jobs were presented to workers as HTML pages with
three buttons alongside each tweet for validating the
language. For example, for Nepali, a Worker can
mark that a tweet is ‘Nepali’, ‘Not Nepali’, or ‘Not
sure.’ We paid $0.05 per job and requested that each
job be completed by three workers.

3.3 Quality Control

To ensure high annotation quality, we follow our
established practices in only allowing our tasks to
be completed by workers who have previously com-
pleted at least 50 jobs on AMT, and who have had at
least 85% of their jobs approved. Our jobs also dis-
play each tweet as an image; this prevents workers
from pasting the tweet into existing online language
processing services (like Google Translate).

We also have control tweets in each job to allow
us to evaluate worker performance. A positive con-
trol is a tweet known to be in the target language;
a negative control is a tweet known to be in a dif-
ferent language. Between three to six of the twenty
tweets in each job were controls. The controls are
taken from the sources used in our Sources method
(§2.1); e.g., our Urdu controls come from sources
like BBC Urdu’s Twitter feed. To further validate
the controls, we also applied our open-domain LID
system (§4.2) and filtered any Source tweets whose
predicted language was not the expected language. Our negative controls are validated tweets in a language that uses the same alphabet as the target (e.g., our negative controls for Ukrainian were taken from our LID-validated Russian and Bulgarian sources).

We collect aggregate statistics for each Worker over the control tweets of all their completed jobs. We conservatively discard any annotations by workers who get below 80% accuracy on either the positive or negative control tweets.

3.4 Dataset Statistics

Table 1 gives the number of triple-validated ‘Gold’ tweets in each language, grouped into those using the Arabic, Devanagari and Cyrillic writing systems. The Arabic data is further divided into tweets acquired using the Sources and Twit-API methods. Table 1 also gives the Purity of the acquired results; that is, the percentage of acquired tweets that were indeed in the target language. The Purity is calculated as the number of triple-verified gold tweets divided by the total number of tweets where the three annotators agreed in the annotation (thus triply-marked either Yes, No, or Not sure).

For major languages (e.g. Arabic and Russian), we can accurately obtain tweets in the target language, perhaps obviating the need for LID. For the Urdu sets, however, a large percentage of tweets are not in Urdu, and thus neither collection method is reliable. An LID tool is needed to validate the data. A native Arabic speaker verified that most of our invalid Urdu tweets were Arabic. Ukrainian is the most glaringly impure language that we collected, with less than 15% of our intended tweets actually in Ukrainian. Russian is widely spoken in Ukraine and seems to be the dominant language on Twitter, but more analysis is required. Finally, Marathi and Bulgarian also have significant impurities.

The complete annotation of all nine languages cost only around $350 USD. While not insignificant, this was a small expense relative to the total human effort we are expending on this project. Scaling our approach to hundreds of languages would only cost on the order of a few thousand dollars, and we are investigating whether such an effort could be supported by enough fluent AMT workers.

4 Language Identification Systems

We now describe the systems we implemented and/or tested on our annotated data. All the approaches are supervised learners, trained from a collection of language-annotated texts. At test time, the systems choose an output language based on the information they have derived from the annotated data.

4.1 LogR: Discriminative LID

We first adopt a discriminative approach to LID. Each tweet to be classified has its relevant information encoded in a feature vector, $\bar{x}$. The annotated training data can be represented as $N$ pairs of labels and feature vectors: $\{(y^1, \bar{x}^1), ..., (y^N, \bar{x}^N)\}$. To train our model, we use (regularized) logistic regression (a.k.a. maximum entropy) since it has been shown to perform well on a range of NLP tasks and its probabilistic outputs are useful for downstream processing (such as aggregating predictions over multiple tweets). In multi-class logistic regression, the probability of each class takes the form of exponential functions over features:

$$p(y = k | \bar{x}) = \frac{\exp(\bar{w}_k \cdot \bar{x})}{\sum_j \exp(\bar{w}_j \cdot \bar{x})}$$

For LID, the classifier predicts the language $k$ that has the highest probability (this is also the class with highest weighted combination of features, $\bar{w}_k \cdot \bar{x}$). The training procedure tunes the weights to optimize for correct predictions on training data, subject to a tunable L2-regularization penalty on the weight vector norm. For our experiments, we train and test our logistic regression classifier ($LogR$) using the efficient LIBLINEAR package (Fan et al., 2008).
We use two types of features in our classifier:

**Character Features** encode the character N-grams in the input text; characters are the standard information source for most LID systems (Cavnar and Trenkle, 1994; Baldwin and Lui, 2010). We have a unique feature for each unique N-gram in our training data. N-grams of up-to-four characters were optimal on development data. Each feature value is the (smoothed) log-count of how often the corresponding N-gram occurs in that instance. Prior to extracting the N-grams, we preprocess each tweet to remove URLs, hash-tags, user mentions, punctuation and we normalize all digits to 0.

**Meta features** encode user-provided information beyond the tweet text. Similar information has previously been used to improve the accuracy of LID classifiers on European-language tweets (Carter et al., 2013). We have features for the tokens in the Twitter user name, the screen name, and self-reported user location. We also have features for prefixes of these tokens, and flags for whether the name and location are in the Latin script. Our meta features also include features for the hash-tags, user-mentions, and URLs in the tweet. We provide features for the protocol (e.g. http), hostname, and top-level domain (e.g. .com) of each link in a tweet. For shortened URLs (e.g. via bit.ly), we query the URL server to obtain the final link destination, and provide the URL features for this destination link.

**4.2 PPM: Compression-Based LID**

Our next tool uses compression language models, which have been proposed for a variety of NLP tasks including authorship attribution (Pavelec et al., 2009), text classification (Teahan, 2000; Frank et al., 2000), spam filtering (Bratko et al., 2006), and LID (Benedetto et al., 2002). Our method is based on the prediction by partial matching (PPM) family of algorithms and we use the PPM-A variant (Cleary et al., 1984). The algorithm processes a string and determines the number of bits required to encode each character using a variable-length context. It requires only a single parameter, the maximal order, \(n\); we use \(n = 5\) for the experiments in this paper. Given training data for a number of languages, the method seeks to minimize cross-entropy and thus selects the language which would most compactly encode the text we are attempting to classify.

We train this method both on our Twitter data and on large collections of other material. These materials include corpora obtained from news sources, Wikipedia, and government bodies. For our experiments we divide these materials into two sets: (1) just Wikipedia and (2) all sources, including Wikipedia. Table 2 gives the sizes of these sets.

### Table 2: Size of other PPM training materials.

| Language  | Wikip. | All   |
|-----------|--------|-------|
| Arabic    | 372 MB | 1058 MB |
| Farsi     | 229 MB | 798 MB |
| Urdu      | 30 MB  | 50 MB  |
| Hindi     | 235 MB | 518 MB |
| Nepali    | 31 MB  | 31 MB  |
| Marathi   | 32 MB  | 66 MB  |
| Russian   | 563 MB | 564 MB |
| Bulgarian | 301 MB | 518 MB |
| Ukrainian | 461 MB | 463 MB |

**4.3 Comparison Systems**

We compare our two new systems with the best-available commercial and academic software.

**TextCat**: TextCat\(^3\) is a widely-used stand-alone LID program. It is an implementation of the N-gram-based algorithm of Cavnar and Trenkle (1994), and supports identification in “about 69 languages” in its downloadable form. Unfortunately, the available models do not support all of our target languages, nor are they compatible with the standard UTF-8 Unicode character encoding. We therefore modified the code to process UTF-8 characters and re-trained the system on our Twitter data (§5).

**Google CLD**: Google’s Chrome browser includes a tool for language-detection (the Google *Compact Language Detector*), and this tool is included as a library within Chrome’s open-source code. Mike McCandless ported this library to its own open source project.\(^4\) The CLD tool makes predictions using text 4-grams. It is designed for detecting the language of web pages, and can take meta-data hints from the domain of the webpage and/or the declared webpage

\(^3\)http://odur.let.rug.nl/vannoord/TextCat/
\(^4\)http://code.google.com/p/chromium-compact-language-detector/
encoding, but it also works on stand-alone text. We use it in its original, unmodified form. While there are few details in the source code itself, the training data for this approach was apparently obtained through Google’s internal data collections.



| Dataset       | Train | Development | Test  |
|---------------|-------|-------------|-------|
| Arabic        | 2254  | 1171        | 1191  |
| Devanagari    | 2099  | 991         | 962   |
| Cyrillic      | 2243  | 1133        | 1146  |

Table 3: Number of tweets used in experiments, by writing system/classification task

Lui and Baldwin ’11: Lui and Baldwin (2011) recently released a stand-alone LID tool, which they call langid.py. They compared this system to state-of-the-art LID methods and found it “to be faster whilst maintaining competitive accuracy.” We use this system with its provided models only, as the software readme notes “training a model for langid.py is a non-trivial process, due to the large amount of computations required.” The sources of the provided models are described in Lui and Baldwin (2011). Although many languages are supported, we restrict the system to only choose between our data’s target languages (§5).

5 Experiments

The nine languages in our annotated data use one of three different writing systems: Arabic, Devanagari, or Cyrillic. We therefore define three classification tasks, each choosing between three languages that have the same writing system. We divide our annotated corpus into training, development and test data for these experiments (Table 3). For the Arabic data, we merge the tweets obtained via our two collection methods (§2); for Devanagari/Cyrillic, all tweets are obtained using the Sources method. We ensure that tweets by a unique Twitter user occur in at most only one of the sets. The proportion of each language in each set is roughly the same as the proportions of gold tweets in Table 1. All of our Twitter-trained systems learn their models from this training data, while all hyperparameter tuning (such as tuning the regularization parameter of the LogR classifier) is done on the development set. Our evaluation metric is Accuracy: what proportion of tweets in each held-out test set are predicted correctly.

6 Results

For systems trained on the Twitter data, both our LogR and PPM system strongly outperform TextCat, showing the effectiveness of our implemented approaches (Table 4). Meta features improve LogR on each task. For systems trained on external data, PPM strongly outperforms other systems, making fewer than half the errors on each task. We also trained PPM on both the relatively small number of Twitter training samples and the much larger number of other materials. The combined system is as good or better than the separate models on each task.

We get more insight into our systems by seeing how they perform as we vary the amount of training data. Figure 1 shows that with only a few hundred annotated tweets, the LogR system gets over 90% accuracy, while performance seems to plateau shortly afterwards. A similar story holds as we vary the amount of out-of-domain training data for the PPM system; performance improves fairly linearly as exponentially more training data is used, but eventually begins to level off. Not only is PPM an effective system, it can leverage a lot of training ma-
Figure 1: The more training data the better, but accuracy levels off: learning curve for \( \text{LogR} \)-chars (note log-scale).

Figure 2: Accuracy of PPM classifier using varying amounts of Wikipedia training text (also on log-scale).

Figure 3: The longer the tweet, the better: mean accuracy of \( \text{LogR} \) by average length of tweet, with tweets grouped into five bins by length in characters.

Table 5: The benefits of aggregating predictions by user: Mean accuracy of \( \text{LogR} \)-chars as you make predictions on multiple Devanagari tweets at a time.

| Number of Tweets | 1     | 2     | 3     | 4     |
|------------------|-------|-------|-------|-------|
| Accuracy         | 97.0  | 98.7  | 98.8  | 98.9  |

In Figure 3, we show how the accuracy of our systems varies over tweets grouped into bins by their length. Performance on short tweets is much worse than those closer to 140 characters in length.

We also examined aggregating predictions over multiple tweets by the same user. We extracted all users with \( \geq 4 \) tweets in the Devanagari test set (87 users in total). We then averaged the predictions of the \( \text{LogR} \) system on random subsets of a user’s test tweets, making a single decision for all tweets in a subset. We report the mean accuracy of running this approach 100 times with random subsets of 1, 2, 3, and all 4 tweets used in the prediction. Even with only 2 tweets per user, aggregating predictions can reduce relative error by almost 60% (Table 5).

Encouraged by the accuracy of our systems on annotated data, we used our PPM system to analyze a large number of un-annotated tweets. We trained PPM models for 128 languages using data that includes Wikipedia (February 2012), news (e.g., BBC News, Voice of America), and standard corpora such as Europarl, JRC-Acquis, and various LDC releases. We then made predictions in the TREC Tweets2011 Corpus.\(^7\)

We observed 65 languages in roughly 10 million tweets. We calculated two other proportions using auxiliary data: \(^8\) (1) the proportion of Wikipedia articles written in each language, and (2) the proportion of speakers that speak each language. We use these proportions to measure a language’s relative representation on Twitter: we divide the tweet-proportion by the Wikipedia and speaker proportions. Table 6 shows some of the most over-represented Twitter languages compared to Wikipedia. E.g., Indonesian is predicted to be 9.9 times more relatively common on Twitter than Wikipedia. Note these are predictions only; some English tweets may be falsely marked as other languages due to English impurities in our training sources. Nevertheless, the good representation of languages with otherwise scarce electronic resources shows the potential of using Twitter to build language-specific social media collections.

\(^7\)http://trec.nist.gov/data/tweets/ This corpus, developed for the TREC Microblog track (Soboroff et al., 2012), contains a two-week Twitter sample from early 2011. We processed all tweets that were obtained with a “200” response code using the twitter-corpus-tools package.

\(^8\)From http://meta.wikimedia.org/wiki/List_of_Wikipedias_by_speakers_per_article
| Language     | Num. Tweets | % of Tweets/ Tot. Tweets | Wikip. Speakers | Tweets/ Speakers |
|--------------|-------------|--------------------------|----------------|-----------------|
| Indonesian   | 1055        | 9.0                      | 9.9            | 3.1             |
| Thai         | 238         | 2.0                      | 5.7            | 1.9             |
| Japanese     | 2295        | 19.6                     | 5.0            | 8.8             |
| Korean       | 446         | 3.8                      | 4.0            | 3.2             |
| Swahili      | 46          | 0.4                      | 3.4            | 0.4             |
| Portuguese   | 1331        | 11.4                     | 3.2            | 2.8             |
| Marathi      | 58          | 0.5                      | 2.9            | 0.4             |
| Malayalam    | 30          | 0.3                      | 2.2            | 0.4             |
| Nepali       | 23          | 0.2                      | 2.1            | 0.8             |
| Macedonian   | 61          | 0.5                      | 1.9            | 13.9            |
| Bengali      | 25          | 0.2                      | 1.9            | 0.1             |
| Turkish      | 174         | 1.5                      | 1.7            | 1.1             |
| Arabic       | 162         | 1.4                      | 1.6            | 0.3             |
| Chinese      | 346         | 3.0                      | 1.4            | 0.2             |
| Spanish      | 696         | 5.9                      | 1.4            | 0.7             |
| Telugu       | 39          | 0.3                      | 1.4            | 0.3             |
| Croatian     | 79          | 0.7                      | 1.3            | 6.1             |
| English      | 2616        | 22.3                     | 1.2            | 2.1             |

Table 6: Number of tweets (1000s) and % of total for languages that appear to be over-represented on Twitter (vs. proportion of Wikipedia and proportion of all speakers).

7 Related Work

Researchers have tackled language identification using statistical approaches since the early 1990s. Cavnar and Trenkle (1994) framed LID as a text categorization problem and made their influential TextCat tool publicly-available. The related problem of identifying the language used in speech signals has also been well-studied; for speaker LID, both phonetic and sequential information may be helpful (Berkling et al., 1994; Zissman, 1996). Insights from LID have also been applied to related problems such as dialect determination (Zaidan and Callison-Burch, 2011a) and identifying the native language of non-native speakers (Koppel et al., 2005).

Recently, LID has received renewed interest as a mechanism to help extract language-specific corpora from the growing body of linguistic materials on the web (Xia et al., 2009; Baldwin and Lui, 2010). Work along these lines has found LID to be far from a solved problem (Hughes et al., 2006; Baldwin and Lui, 2010; Lui and Baldwin, 2011); the web in general has exactly the uneven mix of style, languages, and lengths-of-text that make the real problem quite difficult. New application areas have also arisen, each with their own unique challenges, such as LID for search engine queries (Gottron and Lipka, 2010), or person names (Bhargava and Kondrak, 2010).

The multilinguality of Twitter has led to the development of ways to ensure language purity. Ritter et al. (2010) use “a simple function-word-driven filter...to remove non-English [Twitter] conversations,” but it’s unclear how much non-English survives the filtering and how much English is lost. Tromp and Pechenizkiy (2011) and Carter et al. (2013) perform Twitter LID, but only targeting six common European languages. We focus on low-resource languages, where training data is scarce. Our data and systems could enable better LID for services like indigenoustweets.com, which aims to “strengthen minority languages through social media.”

8 Conclusions

Language identification is a key technology for extracting authentic, language-specific user-generated text from social media. We addressed a previously unexplored issue: LID performance on Twitter text in low-resource languages. We have created and made available a large corpus of human-annotated tweets in nine languages and three non-Latin writing systems, and presented two systems that can predict tweet language with very high accuracy.9 While challenging, LID on Twitter is perhaps not as difficult as first thought (Carter et al., 2013), although performance depends on the amount of training data, the length of the tweet, and whether we aggregate information across multiple tweets by the same user. Our next step will be to develop a similar approach to handle romanized text. We also plan to develop tools for identifying code-switching (switching languages) within a tweet.

Acknowledgments

We thank Chris Callison-Burch for his help with the crowdsourcing. The first author was supported by the Natural Sciences and Engineering Research Council of Canada. The third author was supported by the BOLT program of the Defense Advanced Research Projects Agency, Contract No. HR0011-12-C-0015

9The annotated corpus and PPM system are available online at http://apl.jhu.edu/~paulmac/lid.html
References

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In Proc. HLT-NAACL, pages 229–237.

Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. 2002. Language trees and zipping. Physical Review Letters, 88(4):2–5.

Kay Berkling, Takayuki Arai, and Etienne Barnard. 1994. Analysis of phoneme-based features for language identification. In Proc. ICASSP, pages 289–292.

Aditya Bhargava and Grzegorz Kondrak. 2010. Language identification of names with SVMs. In Proc. HLT-NAACL, pages 693–696.

Andrej Bratko, Gordon V. Cormack, Bogdan Filipic, Thomas R. Lynam, and Blaz Zupan. 2006. Spam filtering using statistical data compression models. JMLR, 6:2673–2698.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In Proc. EMNLP, pages 1301–1309.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon’s mechanical turk. In Proc. NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk, pages 1–12.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog Language Identification: Overcoming the Limitations of Short, Unedited and Idiomatic Text. Language Resources and Evaluation Journal. (forthcoming).

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In Proc. Symposium on Document Analysis and Information Retrieval, pages 161–175.

John G. Cleary, Ian, and Ian H. Witten. 1984. Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications, 32:396–402.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In Proc. ACL, pages 1365–1374.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. JMLR, 9:1871–1874.

Clayton Fink, Jonathon Kopecky, Nathan Bos, and Max Thomas. 2012. Mapping the Twitterverse in the developing world: An analysis of social media use in Nigeria. In Proc. International Conference on Social Computing, Behavioral Modeling, and Prediction, pages 164–171.

Eibe Frank, Chang Chui, and Ian H. Witten. 2000. Text categorization using compression models. In Proc. DCC-00, IEEE Data Compression Conference, Snowbird, US, pages 200–209. IEEE Computer Society Press.

Rayid Ghani, Rosie Jones, and Dunja Mladenic. 2001. Automatic web search query generation to create minority language corpora. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’01, pages 432–433, New York, NY, USA. ACM.

Thomas Gottron and Nedim Lipka. 2010. A comparison of language identification approaches on short, query-style texts. In Proc. ECIR, pages 611–614.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew Mackinlay. 2006. Reconsidering language identification for written language resources. In Proc. LREC, pages 485–488.

Ann Irvine and Alexandre Klementiev. 2010. Using Mechanical Turk to annotate lexicons for less commonly used languages. In Proc. NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk, pages 108–113.

Emre Kiciman. 2010. Language differences and metadata features on Twitter. In Proc. SIGIR 2010 Web N-gram Workshop, pages 47–51.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author’s native language by mining a text for errors. In Proc. KDD, pages 624–628.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In Proc. IJCNLP, pages 553–561.

Paul McNamee. 2005. Language identification: a solved problem suitable for undergraduate instruction. J. Comput. Sci. Coll., 20(3):94–101.

D. Pavelec, L. S. Oliveira, E. Justino, F. D. Nobre Neto, and L. V. Batista. 2009. Compression and stylometry for author identification. In Proc. IJCNN, pages 669–674, Piscataway, NJ, USA. IEEE Press.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In Proc. HLT-NAACL, pages 172–180.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In Proc. EMNLP, pages 583–593.

Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In Proc. EMNLP, pages 254–263.

Ian Soboroff, Dean McCullough, Jimmy Lin, Craig Macdonald, Iadh Ounis, and Richard McCreadie. 2012. Evaluating real-time search over tweets. In Proc. ICWSM.
William John Teahan. 2000. Text classification and segmentation using minimum cross-entropy. In *Proc. RIAO*, pages 943–961.

Erik Tromp and Mykola Pechenizkiy. 2011. Graph-based n-gram language identification on short texts. In *Proc. 20th Machine Learning conference of Belgium and The Netherlands*, pages 27–34.

Fei Xia, William Lewis, and Hoifung Poon. 2009. Language ID in the context of harvesting language data off the web. In *Proc. EACL*, pages 870–878.

Omar F. Zaidan and Chris Callison-Burch. 2011a. The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content. In *Proc. ACL*, pages 37–41.

Omar F. Zaidan and Chris Callison-Burch. 2011b. Crowdsourcing translation: Professional quality from non-professionals. In *Proc. ACL*, pages 1220–1229.

Marc A. Zissman. 1996. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31–44.