Bootstrapped Language Identification For Multi-Site Internet Domains

Uwe F. Mayer
eBay Research Labs
2145 Hamilton Ave
San Jose, CA 95125

ABSTRACT

We present an algorithm for language identification, in particular of short documents, for the case of an Internet domain with sites in multiple countries with differing languages. The algorithm is significantly faster than standard language identification methods, while providing state-of-the-art identification. We bootstrap the algorithm based on the language identification based on the site alone, a methodology suitable for any supervised language identification algorithm. We demonstrate the bootstrapping and algorithm on eBay email data and on Twitter status updates data. The algorithm is deployed at eBay as part of the back-office development data repository.

Categories and Subject Descriptors

I.2.7 [**Computing Methodologies**]: Artificial Intelligence – Natural Language Processing.

General Terms

Algorithms, Performance, Design.

Keywords

Language identification, large data, statistical model, boosting.

1. INTRODUCTION

In today's interconnected global Internet community it is quite common that Internet-based businesses (e.g., eBay, Twitter) operate sites in different languages (e.g., ebay.com, ebay.de, ebay.fr, etc., or twitter.com, twitter.com/?lang=de, twitter.com/?lang=fr, etc.). Many such sites have user-contributed content data, private or public, such as web postings, reviews, blogs, or emails. Usually, but not always, either the site which the user employs to generate the content or more often the site on which it is to appear (or which the recipient uses in case of a message) dictates the language used by the creator of the content. Typically one may expect that knowing the site language alone allows language identification of user-provided content with high precision. On the other hand this approach is not expected to be perfect, and one can employ standard language identification algorithms, which even if not specifically trained for the corpus at hand, also achieve usually still high precision. The task is to improve over either approach to determine the language of the user-created documents.

The general idea, of course, is to combine the language identification based on the site language or language from a user profile with some other supervised algorithm. In particular in a web setting one usually has lots of data, however obtaining suitable accurate labels constitutes the bottleneck. Our approach is to use data labeled by the site or the user profile alone for training, and we build a custom model using a supervised algorithm. Then we combine the language identification based on the site or user profile language with the newly trained supervised model to create a final model. The method of combination could be as simple as a linear model.

Most algorithms for language identification, and in particular most modern ones, employ a statistical approach, often not word (dictionary) based, but rather character or n-gram based [1,2,3]. However, these algorithms often perform not optimally if the documents to be classified are short [4].

The archetypical example of shorts text on the web nowadays are Twitter status updates (so-called tweets), which by design are limited to 140 characters. On the other hand, tweets are not the only short prevalent documents on the web. This research grew out of the need to accurately identify the language of user-to-user email communications on eBay. We found that these often also are rather short. This may well be due to the limited setting of what the messages are usually about (e.g., "do you have this also in a different color?" or "would you ship the item to Europe?", or the complete email could also just be the single word "yes"). It turns out that on these user-to-user email messages a dictionary-based approach does indeed perform very well. We then also apply our methodology to Twitter status updates, which besides being short, are also well known to be difficult to classify due to the vast amount of Twitter-specific slang and abbreviations.

Our specific approach is to look only at the first two and the last two words of the documents (emails, tweets), and use the site or user profile determined language to build frequency tables on a training set. In the email setting, the intuition of looking at the opening word of a message was the observation that many users start their text with one of only relatively few and very language-specific terms, such as hello, bonjour or goedendag. Then again, starting an email with the name of the recipient is also very frequent, which leads to a dilution of the signal captured by the first word. This is why we also consider the second word, which then in turn tends to be language-specific. An analogous reasoning applies to the end of the messages, common endings are strongly indicative of the language, such as good-bye, ciao, or Tschüß. However, if the email is ended with the name of the sender, then the second-to-last word carries this signal.

This approach also works quite well for tweets, which start often with common opening words or abbreviations and often finish with common ending words or abbreviations. For example in English common opening words or abbreviations are I, lol, you, my, …, while common ending words or abbreviations are you, now, me, smh, …, and similarly for other languages.

Normalizing these frequency tables across the languages for each word separately leads to estimated language probabilities, and summing up these probabilities with suitable weights creates a dictionary-based language identification model, with weights on the opening and closing words only. Combination with the site or user profile language identification is easy, the indicator variable is just another (binary) probability weighted into the sum.

Because in the end only the first two and the last two words of a document need to be investigated, the algorithm is computationally very lightweight during scoring, and correspondingly very fast. Additionally our method needs no human-labeled training data, rather it bootstraps itself from the (hopefully relatively strong) learner given by language identification based on the site or user profile alone. This is also where the nature of today's high-volume web sites helps our approach: the hosting domain generally captures vast amounts of such trivially machine-labeled data for free. Furthermore, as a consequence of being built on large amounts of data, the resulting models are very stable. On the eBay data where we have access to a longer period of data, the model still performs exceptionally well one year after fitting. The algorithm is deployed at eBay as part of the back-office development data repository.

With the investigation of the Twitter data, which has the relatively weaker language identifier given by the user profile language, we conclude that the (a-priori) precision of the site or user profile language determination is a key ingredient to our approach: the stronger the site or user profile language learner, the better the algorithm and bootstrapping.

Applications of language identification are not subject of this paper, but of course could include the obvious ones, from spam filtering, fraud detection, sentiment analysis, to customer service queue routing, among others.

2. EBAY MEMBER TO MEMBER EMAILS

The specific data leading to the development of the algorithm were the (non-public) emails between sellers and buyers (or interested members, in case of a pre-sale communication). These are all called member-to-member emails (or messages), in short, M2M emails. The eBay sites that are part of this study use one of the following seven languages: English, German, Italian, French, Spanish, Dutch, or Polish. This glosses over subtle differences, for example, British English, Australian English, and American English are simply identified as English, similarly for French being spoken in France, parts of Canada, parts of Switzerland, and parts of Belgium, the same holds for German being used in Germany, Austria, and parts of Switzerland, and finally Dutch for the Netherlands and parts of Belgium.

M2M messages on eBay usually are generated within the MyMessages facility, which wraps each message with a template. What the template precisely contains is not important, with the exception that the template header before the message ends on "Dear recipient-user-name", and the footer template after the message starts with a signature of the form "- sender-user-name". Hence many eBay members do not include the recipient name as part of their message, and neither do they include their own sender name as a signature. This is important for our algorithm, as otherwise if one looks just at the beginning and the ending of a typical email message one would generally have to discard (and identify!) user names in order to derive a generalizable model. On the other hand, as explained in the introduction, we consider the leading two words and the ending two words, just in case a greeting by name or a signature user name is still present in the message.

A key feature of the eBay M2M data is that it contains many short documents; one- and two-word documents are not uncommon.

Obviously, the strongest indicator of the email language is the site. In our setting we extract the site from the email template, thus avoiding a user database lookup to find the registered site of the user. However, in particular because of a small percentage of template-free emails this results in missing values for some of the records. Also, some users write emails in a language different from their site language, which combined with the missing values ultimately gives a site-language accuracy of about 96% in identifying the language of emails. More precisely, in case the sites of the sender and the recipient differ, this is based on the site of the recipient, which has proven to give slightly better language identification performance. This percentage was established on a random un-biased sample of about 15K emails from June 2010 which were human-labeled into the seven languages considered. For the eBay data, our goal is to improve on this 96% accuracy rate.

3. TWITTER STATUS UPDATES

As the obvious source of short web documents in multiple languages, we also consider Twitter status updates. We used the streaming API to capture a 1% sample of public status updates made freely available by Twitter, commonly known as the "Sprinkler" data feed [6].

For this study we excluded several languages that could easily be separately handled from the Twitter data feed because these languages use a different font encoding: Japanese (ISO 639-1 code "ja"), Chinese (both "zh-cn" and "zh-tw"), Korean ("ko") and Russian ("ru"). Similarly to how we grouped all different flavors of English from eBay's data into simply one English language, we mapped the intricately related macro-languages Malay ("msa") and Indonesian ("id") simply into Indonesian ("id"). The Twitter user profile languages (in the following often called "site languages") that are part of this study were determined on the training set and comprise the following eleven languages: English, Spanish, Portuguese, French, Dutch, Turkish, Indonesian, German, Italian, Filipino, and Hindi. In the end there actually was not a single Twitter status update in Hindi in the test sets, however all code used all eleven languages.

An important point is that while there are no missing values in the language field in the Twitter user profiles, the language specified in the user profile is only about 80% accurate if used to identify the language used for status updates. However, the public user profile distributed by Twitter as part of the streaming API also contains a location field. This field is populated for about three quarters of all status updates, but of these about absolute 15% of the values contain not even a single letter (e.g., contain a "smiley face"), leaving about 60% of locations with potentially identifiable values. Not all of these values map to an actual physical location on a map, but some contain other whimsical entries (e.g., "in my kitchen"). In the end, we prepared a simple parsing model on the location field by using both the geographic location and the language used to specify the location to assign a language to the location (e.g. both "Chicago" and "in my kitchen" would be assigned "English"). We managed to map about 50% of status updates to a location language (from an upper limit of 60%). Finally, we assigned this location language (if present) as the site-language, thus overriding the user profile language. As it turns out, the majority of Twitter users in Indonesia and in the Philippines post in English, and hence for these two locations we did not override the user language from the profile with the location language. With this improved site language from using both the language field and the location field from the user profile, the combined prediction accuracy if using it to predict the language used for the status updates is about 87%, comparing favorably to the 80% accuracy if using the user profile language alone.

In the rest of this study we use both of these Twitter site languages (user profile language alone, or user profile language and location language combined), and we list the results in separate tables.

4. THE ALGORITHM

4.1 Data Preparation

The algorithm expects only absolutely minimal data cleaning. Generally, digits and letters including umlauts and accented letters make up the words, anything else is a word break. For the eBay message data, after breaking the document into words, all words containing a digit were removed, which removed many user-selected user ids and names. For the Twitter data, on the other hand, numbers are often used as abbreviations (e.g., “gr8” for “great”), and hence these words were allowed. As a further concession to the abbreviated nature of Twitter status updates, we also allowed apostrophes (e.g., in this study “I’m” is considered one word in the Twitter data, but two words in the eBay data). Similar to the eBay data we removed Twitter user names from the status updates (e.g., “@Example”) and we saw a slight improvement in the language identification. Finally all the letters are transformed into a uniform case (lower case, for example). On the other hand, no stemming or aliasing is performed.

4.2 Learning of Frequency Tables

After pre-processing, we collect site or user language counts for each of the first two words and each of the last two words of each document. For the eBay M2M data we found during our experimentations that it helps slightly to combine the frequencies from the last two words into a single table. Hence for the eBay M2M data we created only three frequency tables, one each for the first words, the second words, and one for the combined collection of second-to-last and last words. The likely reason for this improvement is that in the combined frequency table stray entries from signature user names will obtain relatively less dominance as they would in a last-word-only table. The Twitter status update data contains no signatures at the end of tweets, and we observed no such improvement by combining the last two words into a single frequency table. We kept those tables separate.

The frequency tables can be both pre- and post-processed. The most obvious post-processing is to remove entries with insufficient total frequency. For both the eBay M2M data and the Twitter status updates data, we collected the frequency tables on a training set of several million documents, and we set the minimum frequency to 50 for the individual (first, second, second-to-last, last) word tables, and to 100 for the combined last words table in the case of eBay M2M data. Additional pre-processing may include removal of undesirable words (e.g., words which are known to carry no signal) from the document before extraction of the leading and terminating words for the frequency tables. For example, for the eBay M2M data we excluded the words ebay and paypal, while for the Twitter data we excluded the abbreviation RT (short for Re-Tweet).

4.3 Scoring

Assigning a language to a given document is highly efficient. It amounts to extracting the first two and the last two words of the document, and for each of these four words w_k (k = 1,…,4) and for each of the predetermined set of languages L looking up the corresponding frequency $f_{kL}(w_k)$ in the count tables k (k = 1,…,4), and computing a linear weighted sum. Recall that for the eBay M2M data we created one combined table for the last two words (k = 3,4), which is to say in this case $f_{3L}(w) = f_{4L}(w)$ for all languages L and all words w.

Let $f_k(w)$ denote the sum of the frequencies for a word w in table k across all languages. Given weights W_k (k = 1,…,4), the language score is computed for each language L as

$$s_L = W_1\, f_{1L}(w_1) / f_1(w_1) + W_2\, f_{2L}(w_2) / f_2(w_2) +$$

$$W_3\, f_{3L}(w_3) / f_3(w_3) + W_4\, f_{4L}(w_4) / f_4(w_4).$$

Lacking any prior, we chose all weights W_k = 1. The final language assigned is the language with the highest score. In case of a score tie, a predetermined preference order based on the expected distribution of languages could be used.

4.4 Combination with the Site Language

We chose a simple approach and combined the above language identification scores s_L in a linear fashion with the site language. Let $\delta(L)$ be the indicator function for the site language (value = 1 if L is the site language, 0 otherwise), and let W_S be a pre-chosen site weight, then the combined site and algorithm score is

$$s_{LS} = W_S\, \delta(L) + s_L.$$

For both the eBay M2M and the Twitter status data, we chose W_S = 1.75, with the idea that the site or user profile language is by itself already very accurate, and this way at least two of the features in the weighted sum s_L have to trigger at values close to 1 to override the site or user profile language.

5. EVALUATION ON EBAY DATA

5.1 Data

As a first evaluation of our approach, we fitted the models and tested the algorithm on proprietary eBay M2M data. Specifically we used the samples listed in the table below.

Table 1. eBay Data Sets

Time Period	Volume	Filter	Labeled By
June 2010	Several million	No	Site language
June 2010	1K per language	Biased	Site language
June 2010	15K	Cleaned	Human
Sept 30, 2010	20K	Cleaned	Human
June 2011	21K	Cleaned	Human

In the table above, "Cleaned" means that records that were using more than one language, or that were ambivalent were removed (e.g., a message consisting of digits only could be any language). "Biased" means that the given distribution of languages was ignored, and rather a specific number of documents was extracted for all languages in order to generate a reference set; these were also filtered so that each sender was allowed only one message in order to capture as much different user behavior as possible. The data set listed in row 1 is a superset of those in rows 2 and 3. No attempt has been made to keep the data sets from rows 2 and 3 disjoint.

5.2 Comparison Algorithms on eBay Data

We compared our algorithm to two industry-standard algorithms, which we also combined with the site language determination for further competitiveness.

5.2.1 Standalone Zipping

We prepared a set of reference documents for each of the seven languages, these are the 1,000 documents each from June 2010 in row 2 of Table 1. We used Gzip as the compression tool, and pruned the 1K reference document files so that for each of the seven languages the resulting gzipped reference set had roughly the same byte count (about 64KB each). We also used a much smaller reference set (about 2KB each) as a second classifier. Classification is as follows: Given a document, for each of the language reference sets add the given document and compress the resulting document set using gzip. Among the seven resulting document sets that one reference set that grew relatively the least must have had similar statistical byte patterns, and its language is selected [4]. One immediate drawback of this algorithm is that even for a one-word message the algorithm needs to compress seven 64KB (or 2KB, respectively) data sets, which is a relatively huge computational effort. Thus the expected run-time performance is expected to be very poor if compared to other methods.

5.2.2 Combination Zipping with Site Language

For the Zipping algorithm, a voting scheme determines whether to use the gzip-determined language or the site language. The voting scheme is as follows. For any language L, let r_L be the ratio of the size of the gzipped reference set with the document attached to the size of the gzipped reference set without the document attached. Let r_{L_2} be the second smallest ratio and r_{L_1} be the smallest ratio (for the given document), and let W_g be a pre-determined weight. The voting scheme is testing $W_g < r_{L_1} / r_{L_2}$, if true it uses the site language, otherwise the gzip language. The ratio of the two smallest ratios is thus taken as a measurement of how sure the gzip determination is of its result. For the 64K (2K) sized reference sets we experimentally fit $W_g = 0.999925$ (0.997, respectively) using the human-labeled June 2010 data (row 3 of Table 1). Note that this combination is not bootstrapped but uses a human-labeled dataset to obtain the optimal combination weight. It is included here for comparison purposes only, and not to illustrate the methods described in this paper.

5.2.3 Standalone Character n-gram Naïve Bayesian

We used the publicly available implementation from Cybozu Labs [2]. This Java software comes out-of-the box pre-trained for 49 languages, the training being based on Wikipedia documents. We used the tool as it is, but also used it with the 49 languages restricted to just the seven languages under consideration, and finally, we also retrained the underlying classifier using a sample of the eBay M2M data, using the same reference sets as for the Zipping classifier from June 2010. The classifier software drops short documents during training, in order to avoid this, we repeated any short message inside the eBay M2M reference training set until it exceeded that drop cut-off (a simple study showed this repetition did improve the performance because it increased the training data size). Note that this custom language profile generation follows the same bootstrapping philosophy as we used for our algorithm, no human-labeled data was needed.

5.2.4 Combination Bayesian with Site Language

The software described in the previous section outputs a probability score for the assigned language. If this score exceeds the prior precision known of the site language (0.96 for the eBay M2M data), then the language assigned by the Bayesian model was used, otherwise the site language was used.

Table 2. Models for eBay Data

Name	Description
Site	Language based on recipient site
Gzip64	Zipping 64K reference docs
Gzip64Site	Zipping 64K with site
Gzip2	Zipping 2K reference docs
Gzip2Site	Zipping 2K with site
Bayes49W	Naïve Bayes 49 languages, Wikipedia profile
Bayes7W	Naïve Bayes 7 languages, Wikipedia profile
Bayes7M	Naïve Bayes 7 languages, M2M profile
Bayes49WSite	Naïve Bayes 49 languages with site, Wikipedia profile
Bayes7WSite	Naïve Bayes 7 languages with site, Wikipedia profile
Bayes7MSite	Naïve Bayes 7 languages with site, M2M profile
Algo	Our algorithm
AlgoSite	Our algorithm with site

5.3 Results on eBay Data

Based on the discussions and explanations of the previous sections, Table 2 on the previous page lists the models we compared. Below are various result tables. Note that the June 2010 data is the training data, while the other data sets are unseen test data (out of sample, out of time).

Table 3. Runtime on a 32 Bit 2.66 GHz System

Name	1 record	15K records	100K records
Gzip64Site	0m 0.5s	30m 0s	(not performed)
Gzip2Site	0m 0.4s	0m 34s	3m 29s
Bayes7MSite	0m 0.3s	0m 3.5s	0m 19s
AlgoSite	0m 1.2s	0m 1.9s	0m 4.3s

Runtimes of the "without site" models are essentially equivalent.

Table 4. Precision on Human-labeled eBay Data

Name	June 2010	Sept 2010	June 2011
Site	0.960	0.958	0.956
Gzip64	0.971	0.974	0.972
Gzip64Site	0.987	0.986	0.988
Gzip2	0.962	0.960	0.958
Gzip2Site	0.988	0.986	0.987
Bayes49W	0.968	0.971	0.970
Bayes7W	0.979	0.981	0.981
Bayes7M	0.992	0.992	0.991
Bayes49WSite	0.984	0.986	0.986
Bayes7WSite	0.987	0.988	0.987
Bayes7MSite	0.995	0.995	0.994
Algo	0.995	0.994	0.996
AlgoSite	0.997	0.996	0.997

Table 5. Precision by Language on June 2011 eBay Data

Grouped True Language	True Count	Correct by Bayes7MSite	Correct by AlgoSite
English German	19,230	19,118 (0.994)	19,196 (0.998)
Dutch French Italian Polish Spanish	1,588	1,580 (0.995)	1,561 (0.983)
ALL	20,818	20,698 (0.994)	20,757 (0.997)

6. EVALUATION ON TWITTER DATA

6.1 Data

As a second cross-domain application, we determined the language profiles and tested the algorithm on public Twitter status update data. Specifically we used the following samples listed in the Table 6. In this table, “Cleaned” means that records that were using more than one language, or that were ambivalent were removed (e.g., a message consisting only of a smiley face could be any language). Additionally all languages not part of the set of eleven Twitter languages considered were removed.

The data from October 2011 was used to generate the location to language mapping explained at the end of Section 3. It was not otherwise used.

The November 2011 data was used in several ways. First, it was used to generate the word frequency tables. Second, a subset was used to generate custom profiles for the Character n-gram Naïve Bayesian classifier.

The December 2011 data was used as a near-term small test set, in particular as a sanity check, while the January 2012 data was used as a longer-term test set.

Table 6. Twitter Data Sets

Time Period	Volume	Filter	Labeled By
Oct 29-31, 2011	750K	No	Not labeled
Nov 13-19, 2011	14.6 million (100% of “Sprinkler” = 1% of all Twitter status data)	No	(a) User profile language (b) User profile language and location
Dec 11-17, 2011	1.2K (0.001% of “Sprinkler”)	Cleaned	Human
Jan 8-10, 2012	6.5K (0.01% of “Sprinkler”)	Cleaned	Human

6.2 Comparison Algorithms on Twitter Data

We followed our approach to evaluating our method on the eBay M2M data, and compared our algorithm to the industry-standard Standalone Character n-gram Naïve Bayesian classifier, which as for the eBay data we also combined with the site language (in the user profile sense, see Section 3) for further competitiveness. We did not evaluate the Zipping algorithm on the Twitter data as it had proved to be non-competitive on the eBay data.

6.2.1 Standalone Character n-gram Naïve Bayesian

As before, we used the publicly available implementation from Cybozu Labs [2], and as for the eBay data used it with all 49 languages, the training being based on Wikipedia documents, and also restricted to just the subset of the considered eleven Twitter site-languages. Similar to the eBay data we also retrained the underlying classifier using a sample of the Twitter data. For this we generated two sets of language profiles, once using the user profile language as the label, and once using the language determined from the location and language from the user profile as the label (see Section 3). The language profiles were generated on the November 2011 data, limited to at most 100K records per language (some less-used languages stayed significantly below this). As before for the eBay M2M data we appended to itself any short message inside the Twitter reference training set until it exceeded the software’s internal minimum message-length cut-off, and as for the eBay data a simple study showed this repetition did improve the performance. Once again this implements the same bootstrapping philosophy as we used for our algorithm, no human-labeled data was needed.

6.2.2 Combination Bayesian with Site Language

As previously explained, the software described in the section above outputs a probability score for the assigned language, and if this score exceeds the prior precision known of the pre-assigned site language, then the language assigned by the Bayesian model was used, otherwise the site language was used. “Pre-assigned site language” in the previous sentence means either (a) the language from the user profile with an accuracy estimate of 80%, or (b) the language determined from the combination of user location and profile language with an accuracy estimate of 87%.

Table 7. Models for Twitter Data

Name	Description
Site	Language based on user site (a) or (b) as explained above
Bayes49W	Naïve Bayes 49 languages, Wikipedia profile
Bayes11W	Naïve Bayes 11 languages, Wikipedia profile
Bayes11T	Naïve Bayes 11 languages, Twitter profile
Bayes49WSite	Naïve Bayes 49 languages with site, Wikipedia profile
Bayes11WSite	Naïve Bayes 11 languages with site, Wikipedia profile
Bayes11TSite	Naïve Bayes 11 languages with site, Twitter profile
Algo	Our algorithm
AlgoSite	Our algorithm with site

6.3 Results on Twitter Data

Based on the discussions and explanations of the previous sections, Table 7 above lists the models we compared, and below are the result tables. There are two sets of all results (Tables 8a and b), corresponding to the two site-language strategies: (a) the site language is the language from the user profile, or (b) the site language is the language determined from the combination of language and location from the user profile.

Table 8a. Precision on Human-labeled Twitter Data with Site-language = User Profile Language

	Dec 2011	Jan 2012
Algo	0.657	0.664
Bayes11T	0.821	0.810
Bayes49W	0.820	0.816
Site	**0.793**	**0.828**
AlgoSite	0.804	0.830
Bayes11TSite	0.858	0.848
Bayes49WSite	0.864	0.860
Bayes11W	0.873	0.863
Bayes11WSite	0.902	0.897

Table 8b. Precision on Human-labeled Twitter Data with Site-language = Language from Location and Language from User Profile

	Dec 2011	Jan 2012
Algo	0.679	0.689
Bayes49W	0.820	0.816
Bayes11T	0.853	0.834
Bayes11W	0.873	0.863
Site	**0.878**	**0.869**
Bayes11TSite	0.900	0.885
Bayes49WSite	0.893	0.889
AlgoSite	0.904	0.897
Bayes11WSite	0.917	0.914

Table 9. Precision by Language on Jan 2012 Twitter Data

True Language	True Count	Correct by Bayes11WSite	Correct by AlgoSite
English	3596	3390 (0.94)	3566 (0.99)
Portuguese	794	670 (0.84)	639 (0.80)
Spanish	674	595 (0.88)	601 (0.89)
Indonesian	411	355 (0.86)	247 (0.60)
Dutch	184	168 (0.91)	68 (0.37)
Turkish	55	52 (0.95)	49 (0.89)
French	53	48 (0.91)	36 (0.68)
Italian	22	19 (0.86)	11 (0.50)
German	20	18 (0.90)	8 (0.40)
Filipino	16	12 (0.75)	0 (0.00)
ALL	5825	5330 (0.914)	5225 (0.897)

7. DISCUSSION / CONCLUSIONS

As is apparent from comparing Tables 4, 8a and b, there are significant differences between the eBay M2M data, and the Twitter status data. On the positive side, on all data sets all models / approaches are very stable, and show no over-fitting. Scoring eBay M2M data one year out from the training period achieves the same precision, showing that the underlying patterns are well captured by all approaches. The Twitter data was evaluated on a shorter timeframe, but the models also appear stable.

On the eBay M2M data the Bayesian model shows, as expected, that a custom model (Bayes7M) performs better than a generic model (Bayes7W). On the other hand, this is not the case on the Twitter status data, where the generic profile (Bayes11W) outperforms the custom profile (Bayes11T). The email messages in the eBay M2M data are fairly clean text with highly accurate site language identification labels, certainly if compared to Twitter status updates, where abbreviations and creative slang are the norm and where the user profile language (and location) information is less accurate in identifying the language of status updates. In other words, the major differences between the eBay M2M and the Twitter status update data sets seem to be the cleanliness of the data and the vast difference in the labeling accuracy by the site language. It stands to reason that this noisy signal confuses both the Bayesian model's language profiles when built on the Twitter data, and also our dictionary-based model's frequency tables. Indeed, on the Twitter status update data labeled by the user profile language, which has the lowest accuracy of all site-languages considered in this study, both modeling approaches perform poorly when evaluated stand-alone. For the Bayesian model using an off-the-shelf profile avoids this pitfall.

From a pure run-time consideration (Table 3), the Zipping method is not competitive in a real-time high-volume application. While the Bayesian model has a shorter load time as compared to our model, which must load the more extensive count tables, once the model is loaded, scoring with the Bayesian model is more than an order of magnitude slower as compared to scoring with our model.

Our observation of the poor run-time performance of the Zipping method along with its also relatively poor language identification performance validates a previous observation from the literature dated shortly after the original publication of the Zipping algorithm [6].

Maybe surprisingly, our simple custom model performed in the same ballpark as all the other models, in an overall sense, at least if the underlying site language classifier was strong. Its natural role thus is as a boost to an already existing strong language detector. A more detailed breakdown (Tables 5 and 9) shows that it roughly matched or outperformed the best competing model on the popular languages, while having a (slightly) higher error rate on the less popular languages. This imbalance was more pronounced on the more noisy Twitter data. As to specific languages, the main root cause of the poor language identification performance of our algorithm on the Twitter data, even when combined with the site, in particular on the Filipino language (Table 9) is that all users tweeting in Filipino had English as their site language from their user profile. Because of the rarity of the Filipino language there was insufficient evidence in the frequency tables to override this dominant site language. The same explanation applies on the Twitter data to the lower identification rate of the Dutch language where more than half of the users have English as their site language, and also to German, where just under half of the users have English as their site language.

The performance on the Twitter data indicates that our word-frequency based approach has a strong dependency on the accuracy of the classifier based on the site-language. When this accuracy is lower, this lack of accuracy transports itself into the frequency tables, and the algorithm's performance suffers significantly (Table 8a). On the other hand, the boot-strapping procedure using an existing strong classifier based on the site-language works well, for all of the algorithms considered (Tables 4 and 8b). Indeed, even on noisy Twitter data the bootstrapping procedure boosts the precision to roughly the 90% range (Tables 8a and 8b), which is significantly better than the roughly 80% accuracy one obtains from the user profile language alone.

One key point is that both our algorithm and our bootstrapping are completely unsupervised in the classical sense that no human-labeled data is needed, instead the site provides the necessary labeling. The weights in our algorithm including the weight for the combination with the site language have not been optimized but were set based only on insight into the model structure, and could likely be improved. The combination of the Bayesian model with the site-language classifier requires a single a-priori estimate on the precision of the site-language classifier, but nothing else in form of labeled data, and it boosts the performance measurably.

In summary, we presented a simple, stable, and highly runtime-efficient algorithm for language identification, which does not need any human-labeled training data, provided there already exists a strong language identifier based on a site language or user language. While it is at heart a statistical method based on frequency tables on a machine-generated dictionary, the decision of building the dictionary only on the opening and closing words of a message was driven by human insight into the problem structure. The method shines on web-scale data, validating once more that bigger data is better data, and that, when combined with human understanding of the problem domain, bigger data is even better data. Our approach in particular works well for short documents, and its precision is on-par with state-of-the-art language identification.

8. ACKNOWLEDGMENTS

Our thanks go to Cybozu Labs, Inc. for publicly posting their implementation of a character n-gram Bayesian language classifier, thus allowing us to easily compare our method to a state-of-the-art classifier.

9. REFERENCES

[1] Dunning, Ted. 1994. *Statistical Identification of Language*. Technical Report MCCS 94-273. Computing Research Laboratory, New Mexico State University.

[2] Shuyo, Nakatani. 2010. *Language Identification Library*. Technical Presentation and Java Implementation. Cybozu Labs, Inc. http://code.google.com/p/language-detection/.

[3] Benedetto, Dario, Caglioti Emanuele and Loreto, Vittorio. 2002. *Language trees and zipping.* Physical Review Letters, 88:4.

[4] Řehůřek, Radim and Kolkus, Milan. 2009. *Language Identification on the Web: Extending the Dictionary Method*. Computational linguistics and intelligent text processing. Lecture Notes in Computer Science, 2009, Volume 5449/2009, 357-368, DOI: 10.1007/978-3-642-00382-0_29.

[5] Twitter, *Streaming API.* Online documentation at https://dev.twitter.com/docs/streaming-api.

[6] Goodman, Joshua, 2002. *Extended comment on "Language Trees and Zipping".* Microsoft Research, Feb 21 2002.