Boot-Strapping Language Identifiers for Short Colloquial Postings

Moises Goldszmidt, Marc Najork, and Stelios Paparizos

Microsoft Research, Mountain View, CA 94043, USA
{moises,najork,steliosp}@microsoft.com

Abstract. There is tremendous interest in mining the abundant user generated content on the web. Many analysis techniques are language dependent and rely on accurate language identification as a building block. Even though there is already research on language identification, it focused on very 'clean' editorially managed corpora, on a limited number of languages, and on relatively large-sized documents. These are not the characteristics of the content to be found in say, Twitter or Facebook postings, which are short and riddled with vernacular.

In this paper, we propose an automated, unsupervised, scalable solution based on publicly available data. To this end we thoroughly evaluate the use of Wikipedia to build language identifiers for a large number of languages (52) and a large corpus and conduct a large scale study of the best-known algorithms for automated language identification, quantifying how accuracy varies in correlation to document size, language (model) profile size and number of languages tested. Then, we show the value in using Wikipedia to train a language identifier directly applicable to Twitter. Finally, we augment the language models and customize them to Twitter by combining our Wikipedia models with location information from tweets. This method provides massive amount of automatically labeled data that act as a bootstrapping mechanism which we empirically show boosts the accuracy of the models.

With this work we provide a guide and a publicly available tool [1] to the mining community for language identification on web and social data.

Keywords: Language Identification, Wikipedia, Twitter.

1 Introduction

The last decade has seen the exponential rise of user-generated content such as contributions to web forums, Facebook posts and Twitter messages. There is a tremendous interest in mining this content to extract trends, to perform sentiment analysis [14], for automatic machine translation [10], and for different types of social analytics [2]. There is consequently an entire industry providing infrastructure, tools, and platform support to address these problems. Many of the techniques are either language dependent (i.e., affective words for sentiment analysis) or can benefit dramatically from knowing the language to apply certain

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 95–111, 2013.

type of rules and knowledge. Thus, reliable automatic language identification is a basic requirement for providing higher level analytic capabilities.

Even though there is a large body of research on language identification, labeled as "solved" by some [13], the conventional wisdom is no longer valid. Computers and smartphones have become internationalized, making it trivial to publish postings in one's native language. Constraints in the number of characters have resulted in abbreviations (such as OMG, LOL), and in sentiment and emphasis being expressed using repetition of letters. In addition a new vernacular, particular to each language, using misspellings and incorporating other languages into the mix has emerged. These characteristics are different from clean and editorially managed corpora used to train language identifiers in the past. These trends combined with the rise of microblogging has caused renewed interest on research in language identification [3,4].

An big obstacle to adapting established models and techniques for language identification is the generation of a sizable corpus of labeled data for all languages in the world that keeps up with the trends described above. In this paper we propose and evaluate such a methodology. The first step is to use Wikipedia as training material for producing an initial set of language identifiers. Wikipedia provides a good source of user generated content, covering a wide variety of topics in 280 languages. We restrict ourselves to the 52 languages with at least 50,000 documents each. Besides being user-generated, these documents have interesting characteristics such as incorporating words from non-primary languages (e.g., Latin and Greek word definitions, pronunciation guidance, etc.), making classifiers more robust to multilingual content. Using language identifiers trained this way, we first characterize the tradeoff of the various parameters for a popular set of language classifiers. We then directly apply the classifiers trained on Wikipedia to two sets of labeled Tweets with different characteristics. Our results indicate that this procedure already yield acceptable identifiers. In order to further improve performance in the context of Twitter, we combine our language prediction with the country information found in tweets to assemble a more appropriate training corpus – each tweet that is labeled as being in language "L" comes from a region where the native language is indeed "L". We retrain our language identifiers using this corpus and repeat our experiment on tweets, resulting in a significant accuracy increase.

All our experiments are performed on well established language identification models and algorithms. The contributions of this paper are:

1. A careful empirical quantification of the different tradeoffs in the selection of the free parameters of existing language identifiers to be used as a guide on selecting the most efficient solution for training and testing depending on the task and the scale.
2. A methodology for automated and unsupervised labeled data generation for current social and colloquial postings by taking advantage of side information such as location. This can be generalized for any additional feature besides location and provides a boot-strapping mechanism for other social data sets.

This paper is organized as follows: Section 2 provides a taxonomic overview of statistical language classifiers and tokenization methodology; Section 3 defines a principled way to evaluate language identification, Section 4 describes the setup and detailed results of our experiments; Section 5 surveys related work, and Section 6 offers concluding thoughts.

2 Statistical Language Classifiers

Automatic language identification is a classification task consisting of finding a mapping from a *document* to the language it is written in. In this paper we consider statistical classifiers, namely those that model both languages and documents using token frequencies.

In order to construct language models, we use a corpus of documents labeled with their respective languages, extract tokens from each document, compile their frequencies, and use the token frequencies of all documents in a given language to build a *language profile*, a function from tokens to frequencies. The document to be classified is represented by a *document profile* using the same techniques. The classification process consists of computing a similarity score between a document profile and each language profile, and reporting the language whose profile is most similar to the document.

Such a statistical approach is language-agnostic and presents the advantages that (i) models can be constructed without any knowledge of the morphology or grammar of a language; and (ii) there is no need for stemming, part-of-speech tagging, and the like. Still, there are several design choices to be made regarding text normalization, tokenization, profile size, and the choice of profile similarity metric, all of which we explore in this paper.

Text normalization refers to any cleanup performed on the training corpus or the document instance prior to tokenization, e.g. the removal of spurious whitespace characters and punctuation. It may also include case folding: whether to convert upper-case characters into their lower-case equivalents or not. For some languages, there are firm grammatical rules governing capitalization; for example, in German, all nouns are capitalized. Other languages, including English, only capitalize the first word of a sentence and certain parts of speech.

In virtually all previous work, tokens are either character n-grams or word n-grams[1]. Language profiles may contain n-grams for multiple values of n. We write $\{n_1, n_2, n_3\}$ -grams to denote token sets that include n_1-grams, n_2-grams and n_3-grams. There are various trade-offs to consider when choosing between character n-grams and word n-grams and when deciding on the value(s) of n. The alphabet of Western languages is quite limited (52 lower- and upper-case letters in English), so the set of possible n-grams remain manageable for small values of n. By contrast, the set of valid words in a language is very large.

[1] In the context of this paper, a character is a Unicode symbol. We break words on whitespace characters; ignoring the different word-breaking rules of Asian languages.

We consider three different families of profile similarity measures:

1. Rank-based approaches (RANK). In these approaches the tokens in the language and document models are ranked (according to their observed frequencies in the training corpus and document respectively). In this paper we investigated two rank correlation measures in order to compute a similarity score between these two rankings: Spearman's footrule (SF) [16] and Spearman's rho (SR) [15].
2. Vector-based approaches (VECTOR). These approaches assume that each language model defines a vector in the (very high-dimensional) space of tokens, and the similarity between the vector of the language model and the vector of the document model is given by the cosine between the vectors. Vector-based approaches are used widely in Information Retrieval systems to quantify the similarity between a document and a query.
3. Likelihood-based approaches (LIKELY). These approaches use the frequencies in the models to estimate the probability of seeing a token. Given a document to be classified we can estimate the likelihood that the document was generated from the language model. Likelihood-based approaches differ from the rank and vector approaches in that the document under consideration does not need to be converted into a document profile.

The remainder of this section provides a more formal definition of the profile similarity measures we consider in this paper. We use D to denote a document and L to denote a language profile. All of the measures described below have been adjusted such that higher values indicate greater similarity between D and L. Let T denote the universe of tokens. We write $F_\Psi(t)$ to denote the frequency of token t in profile Ψ, and we write $P_\Psi(t)$ to denote the normalized frequency of token t in profile Ψ, $P_\Psi(t) = \frac{F_\Psi(t)}{\sum_{t' \in T} F_\Psi(t')}$. We will write $t \in \Psi$ to denote the tokens in Ψ, that is, those $t \in T$ where $F_\Psi(t) > 0$.

Rank-Based Approaches (RANK): All rank-based similarity measures rank-order the tokens in D and L and then compute a similarity measure on the rank-ordering. It is worth pointing out that rank-based approaches use token frequency information only to rank-order tokens (and discards the difference in magnitude). Let $R_\Psi(t_i) = i$ denote the rank of the token in the sorted order. If there are z ranks, we define $R_\Psi(t) = z + 1$ for all $t \notin T$. Our variant of **Spearman's footrule** is defined as $SF(D, L) = -\sum_{t \in D} |R_L(t) - R_D(t)|$, where $|x|$ denotes the absolute value of x. Our variant of **Spearman's rho** is defined as $SR(D, L) = -\sum_{t \in D} (R_L(t) - R_D(t))^2$. In both these measures a value of 0 indicates perfect similarity between D and L.

Vector-Based Approaches (VECTOR): The prototypical similarity measure between two vectors in a high-dimensional space is their **cosine similarity**, that is, their dot product normalized by their Euclidean lengths: $CS(D, L) = \frac{\sum_{t \in D} F_D(t) F_L(t)}{\sqrt{\sum_{t \in D} F_D(t)^2} \sqrt{\sum_{t \in L} F_L(t)^2}}$.

Likelihood-Based Approaches (LIKELY): Likelihood-based approaches use the statistical information captured in each language profile to estimate the **probability** that document d was generated from a given language model. Assume that d consists of symbols (characters or words) $(s_1 \dots s_z) \in S$, S the universe of symbols, and that the language models capture n-gram frequency information. We estimate the probability that d is generated by language model L by directly looking up $P_L(s_1 \dots s_n)$, and then sliding an n-symbol window over d, advancing one symbol at a time, and computing $P_L(s_{1+i} \dots s_{n+i} | s_{1+i} \dots s_{n+i-1})$, the probability that symbol s_{n+i} is preceded by $s_{1+i} \dots s_{n+i-1}$. Thus $P_L(s_1 \dots s_z) \approx P_L(s_1 \dots s_n) \prod_{i=1}^{z-n} P_L(s_{1+i} \dots s_{n+i} \, | s_{1+i} \dots s_{n+i-1})$, where $P_L(s_{1+i} \dots s_{n+i} \mid s_{1+i} \dots s_{n+i-1}) = \frac{P_L(s_{1+i} \dots s_{n+i})}{\sum_{s \in S} P_L(s_{1+i} \dots s_{n+i-1} s)}$. It is possible for L not to contain an entry for some n-gram in d. As we are performing a maximum likelihood estimation of the parameters this will be a problem as we will divide by zero. One standard approach to this problem is to smooth the maximum likelihood estimate by using information from a larger (and related) population that has a non-zero values. There are many choices including the amount of smoothing (usually a linear combination of values) as well as the selection of a suitable population. After experimentation, we settled for an approach that assumes a small value such as the min $P'_L(t')$ amongst all tokens t' in all languages L'.

3 Performance Measures

The accuracy A of an identifier is estimated as follows [6]: Let $True(\ell)$ be the number of documents in language ℓ in $\mathcal{L}$, and $TP(\ell)$ be the number of true positives – that is, documents in ℓ that were correctly classified, then $A = \frac{\sum_{\ell \in \mathcal{L}} TP(\ell)}{\sum_{\ell \in \mathcal{L}} True(\ell)}$. Given the fact that different languages have very different numbers of documents for testing (and training), we would like our estimate to reflect that our confidence in a classifier performance depends on the number of samples. To this end, we use *weighted accuracy*, where each language is weighted using the standard deviation of the estimate of the accuracy as a measure of the uncertainty on that estimate. Thus we think of $TP(\ell)$ as coming from a Binomial distribution with parameters $(A(\ell), True(\ell))$, where $A(\ell)$ is the maximum likelihood estimate of the accuracy: $A(\ell) = \frac{TP(\ell)}{True(\ell)}$. Now, we take the weight the inverse of the standard deviation of this estimate: $W(\ell) = \sqrt{\frac{True(\ell)}{A(\ell)(1-A(\ell))}}$. So the weighted accuracy of an identifier is: $WA = \frac{\sum_{\ell \in \mathcal{L}} A(\ell) W(\ell)}{\sum_{\ell \in \mathcal{L}} W(\ell)}$.

To compare the statistical significance between the reported accuracies WA_1 and WA_2 of identifiers 1 and 2, we can use a Wald test [19] directed at rejecting the null hypothesis (H_0) that $WA_1 - WA_2 = 0$ as advocated in [19] when comparing predictors. Again assuming that the accuracies come from a Binomial distribution, and letting n_1 and n_2 denote the number of samples used in classifiers 1 and 2 respectively, we have that

$$\frac{WA_1 - WA_2}{\sqrt{\frac{WA_1 * (1 - WA_1)}{n_1} + \frac{WA_2 * (1 - WA_2)}{n_2}}} \tag{1}$$

essentially computes the number of standard deviations providing a degree of confidence necessary to reject H_0. Regarding this expression as approximating a standard Normal distribution,[2] we can reject H_0 and declare the accuracies as different if this expression is bigger than 2 with approximately 95% confidence.

For example, assume $n_1 = n_2 = 20$ million samples, and one classifier has a weighted accuracy of 0.901 and another has a weighted accuracy of 0.90. The difference is 0.001 – is this significant? An application of Eq. 1 yields a results of 10.5, which is bigger than 2 and provides us with confidence that the difference is significant.[3] This should not be surprising with this number of samples. Indeed, taking the weighted accuracy as coming from a Binomial with around 20 million samples, only differences in the fourth digit should start concerning us.

4 Experiments

We used Wikipedia to build our language profiles and evaluate the various language identifier design choices. We downloaded the XML files for all languages. We selected the languages that have over 50000 documents as the popular and most representative ones. We concatenated the text of all documents of a language into a single document and build the language profile. Table 1 shows a summary of information for the languages we considered.

In our experiments, we probed a variety of design choices for language identifiers. For character n-grams, we considered $\{3\}$-grams (a popular choice in the literature) as well as $\{1, 2, 3, 4, 5\}$-grams (as suggested by Cavnar & Trenkle [6]). Going above the 5 character grams did not produce any benefits and increased the language profiles dramatically. For word n-grams, we considered only 1-grams – in other words, a language profile is simply the lexicon of that language. Multi-gram words could be used, but this would be more suitable for phrase prediction than language prediction, and the space required for even bi-grams words made it computationally infeasible for us. For language and document profile sizes, we explored retaining all tokens, and for performance reasons we also explored using the 10k or 500 most frequent ones. Table 2 summarizes our choices. In each case, we explored both case folding and leaving the capitalization unchanged. In other words, in each of the experiments described below we compare the performance of 42 different classifiers.

4.1 Language Identification Design Alternatives

In-Model Accuracy: Our first experiment compares the performance of the 21 design choices described in Table 2. We used the uncleaned Wikipedia abstracts of the 52 languages shown in Table 1 for both training and testing; we did not perform any case folding. We conducted in-model testing using every abstract in the collection. The results are summarized in Table 3. We observe the following:

[2] This will certainly be true for all tests sets involving Wikipedia data, given that even the smallest test set contains millions of samples.

[3] When the samples are the same we are violating an assumption of independence that will impact the degrees of freedom; yet with this many samples this has no effect.

Table 1. Basic statistics for the 52 languages we trained and tested on

Language	# docs	# words	# chars	Language	# docs	# words	# chars
English	3,841,701	69,995,728	420,822,801	Arabic	154,105	3,638,910	21,077,370
German	1,334,677	21,476,871	153,119,580	Serbian	151,409	1,850,106	12,006,824
French	1,175,638	19,664,258	116,384,106	Lithuanian	142,468	1,314,498	10,254,168
Italian	874,827	14,892,786	92,885,963	Slovak	130,348	1,703,213	11,452,875
Dutch	871,310	16,666,710	108,570,923	Malay	130,170	1,878,252	12,846,493
Polish	852,219	11,107,886	79,515,315	Hebrew	124,884	2,408,802	13,677,123
Spanish	851,369	18,239,492	109,873,261	Bulgarian	124,665	2,192,227	13,911,993
Russian	800,527	8,972,267	66,586,091	Kazakh	122,442	1,879,736	14,817,377
Japanese	791,350	5,390,566	40,296,474	Slovene	121,968	1,496,767	10,103,516
Portuguese	706,771	13,518,871	81,500,258	Volapük	118,923	1,757,761	9,813,402
Swedish	417,092	7,671,785	49,951,384	Croatian	109,103	1,499,300	10,120,238
Chinese	385,528	2,526,386	19,823,371	Basque	106,846	1,463,312	11,361,999
Catalan	359,848	7,860,184	44,345,082	Hindi	92,371	4,337,272	10,367,220
Ukrainian	330,559	4,148,304	29,714,043	Estonian	90,333	1,061,293	8,128,345
Norwegian	320,318	5,557,555	34,993,643	Azerbaijani	84,265	714,444	5,041,032
Finnish	284,303	3,093,762	25,700,474	Galician	78,419	2,089,220	12,265,696
Vietnamese	247,286	5,141,075	25,607,852	Nynorsk	75,399	1,385,501	8,466,704
Czech	214,219	3,040,486	20,456,549	Thai	70,863	2,085,809	9,704,446
Hungarian	206,518	2,613,681	19,410,794	Greek	67,634	1,676,886	11,126,017
Korean	186,746	2,751,819	11,251,750	Latin	62,985	1,149,511	8,383,645
Indonesian	182,026	3,174,474	22,359,174	Occitan	55,520	830,662	4,685,239
Romanian	170,328	2,372,032	15,006,843	Tagalog	54,796	809,647	4,856,032
Persian	170,137	3,397,007	16,845,518	Georgian	53,736	731,235	5,847,744
Turkish	164,263	2,114,148	15,257,393	Haitian	53,575	509,151	2,650,222
Danish	158,497	2,711,688	17,229,078	Slavomacedonian	53,185	857,418	5,538,833
Esperanto	158,152	2,761,440	17,518,543	Serbo-Croatian	52,922	826,400	5,460,844

Table 2. Design alternatives explored in this paper

	$\{1,\cdots,5\}$-char			$\{3\}$-char			$\{1\}$-word				$\{3\}$-char	$\{1\}$-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	×	×	×	×	×	×	×	×	×	VECTOR	×	×
RANK SR	×	×	×	×	×	×	×	×	×	LIKELY	×	

Table 3. Weighted accuracy % of in-model testing

	$\{1,\cdots,5\}$-char			$\{3\}$-char			$\{1\}$-word				$\{3\}$-char	$\{1\}$-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	93.35	89.17	78.56	89.45	88.56	80.92	90.97	88.66	83.39	VECTOR	80.19	79.88
RANK SR	93.35	89.64	79.08	87.41	87.57	81.14	83.03	81.17	77.96	LIKELY	88.68	

- $\{1,\cdots,5\}$-char RANK(SF, all) and RANK(SR, all) have the highest weighted accuracy.
- $\{1,\cdots,5\}$-char RANK is preferable.
- Using the full language profile (all) is better than restricting to smaller sizes.
- The vector-based approaches are not competitive.

Table 4. Weighted accuracy change for ten-fold cross validation vs in-model testing from Table 3. Negative values indicate over-fitting.

	{1, ··· , 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	-0.51	0.01	0.00	-2.87	-0.04	0.00	-2.90	-0.09	0.03	VECTOR	-0.02	-0.51
RANK SR	-0.57	0.00	0.00	-2.87	-0.08	-0.02	-3.65	-0.12	0.01	LIKELY	-0.90	

Table 5. Weighted accuracy change for case folding. Negative values indicated that case folding hurts accuracy (compared to Table 4).

	{1, ··· , 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	-0.82	0.02	0.21	0.00	-0.56	-0.41	-0.96	-0.14	0.31	VECTOR	-0.42	-0.61
RANK SR	-0.96	-0.02	0.28	0.13	-1.03	-0.35	-1.06	-0.19	0.19	LIKELY	-0.61	

Cross-Validation: In order to quantify the impact of over-fitting we repeated the same experiment using ten-fold cross validation. Table 4 shows the difference between the weighted accuracies of ten-fold and in-model experiments. A negative value indicates the ten-fold accuracy is lower, i.e. over-fitting occurred. We observe the following:

- $\{1, \cdots, 5\}$-char, RANK(SF, all) and RANK(SR, all) still have the highest weighted accuracy.
- Over-fitting is a bigger issue when using the full language profile (all). This makes sense: the truncated language profiles omit low frequency tokens.
- Amongst rank-based approaches, $\{1, \cdots, 5\}$-char are less affected.
- Rank-based classifiers are more affected than Vector or Likelihood-based for the same tokenization scheme and profile limit.

Case Folding: We lowercased both training and test data and repeated the same ten-fold cross validation experiment with all other choices unmodified. Table 5 shows the difference between the weighted accuracies of using case folding vs leaving the capitalization as is. A negative value indicates that case folding lowers accuracy, i.e. lower casing is a bad idea. We observe the following:

- $\{1, \cdots, 5\}$-char, RANK(SF, all) and RANK(SR, all) still have the highest weighted accuracy.
- By and large, case folding not only does not help much, but in many cases produces statistically significant worse results. We attribute this to the fact that in some languages, such as German, capitalization is governed by strict grammatical rules.

Language Specific Results: Next, we tested the accuracy of classifiers with respect to the 52 languages in our corpus. The results are shown in Figure 1. The solid black line shows the weighted accuracy of the classifier from Table 2 that performed best for each given language; the dotted red line shows the weighted

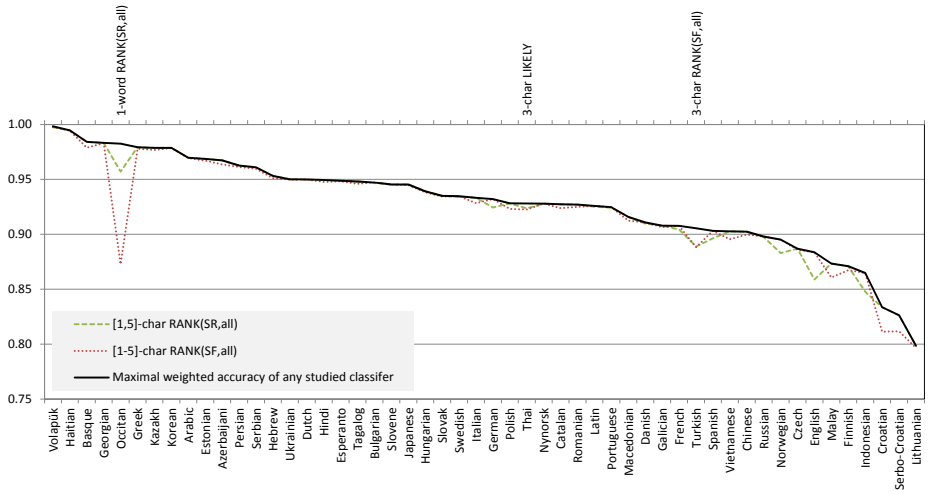

Fig. 1. Classification accuracy broken down by language. The solid black line shows the accuracy of the classifier from Table 2 that performed best for each given language; the dotted red line shows the accuracy of $\{1, \cdots, 5\}$-char RANK(SF, all), and the dashed green line shows the accuracy of $\{1, \cdots, 5\}$-char RANK(SR, all). For the three languages (Occitan, Thai, and Turkish) where these two classifiers did not perform best, the best-performing classifier is listed along the upper horizontal axis.

accuracy of $\{1, \cdots, 5\}$-char RANK(SF, all), and the dashed green line shows the weighted accuracy of $\{1, \cdots, 5\}$-char RANK(SR, all). There were only three languages where neither of these two classifiers performed best: Occitan, Thai, and Turkish. In the case of Thai, the accuracy gap was minor, in the case of Turkish and more so Occitan it was substantial. For Occitan, 1-word RANK(SR,all) performed best; for Turkish, it was 3-char RANK(SF,all), and in the case of Thai, the 3-char LIKELY classifier slightly outperformed the overall leaders. We speculate that Occitan is sufficiently close to Catalan that their character n-gram profiles are very similar, while their lexica are different enough to allow differentiation, giving an edge to word-based approached. We are not sure why 3-character tokens work better than one- to five-character tokens for Turkish. Languages with languages-specific characters sets (such as Hebrew and Greek) provide a strong signal to any character-based classifier, and thus can be recognized with high accuracy. By contrast, our classifiers performed relatively poorly on Chinese. This is to be expected because it has a large alphabet of information-rich symbols, meaning the space of character n-grams is both large and sparse.

Number of Languages: We hypothesized that language detection becomes harder as the set of languages increases. In order to test that hypothesis we repeated the experiment described in Table 3 using only the 8 languages listed in [6] (namely, English, German, Dutch, Polish, Portuguese, Spanish, Italian and French). Table 6 shows the difference between the weighted accuracies of 52

Table 6. Weighted accuracy change for restricting the set of languages to 8. Positive values indicate fewer languages produce better accuracy (compared to Table 3).

	{1, ···, 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	-0.39	2.10	5.99	0.04	1.49	6.14	1.18	2.84	4.22	VECTOR	6.31	9.02
RANK SR	-0.71	1.90	5.58	-1.25	1.37	5.72	3.35	5.07	7.41	LIKELY	2.65	

Table 7. Two best classifiers applied to full Wikipedia articles and Wikipedia abstracts of at least 150 characters for both 8 and 52 languages

[1, 5]-char, all	8L full	8L abs	52L abs		8L full	8L abs	52L abs
RANK(SF) weighted	99.52	99.96	99.95	un-weighted	98.91	99.90	99.74
RANK(SR) weighted	99.46	99.97	99.98	un-weighted	97.47	99.82	99.71

and 8 languages. Generally speaking differences are positive, indicating that it is indeed easier to classify languages when presented with fewer choices. However, the two classifiers that perform the best ({1, ···, 5}-char RANK(SF, all) and RANK(SR, all)) perform slightly worse for fewer languages. Moreover, none of the accuracy values approaches the accuracy scores reported by [6].

Limiting Minimum Document Size: In order to understand the difference between our performance and that of [6] we explored three alternative hypotheses: (i) the quality of Wikipedia abstracts might be low; full Wikipedia documents might be better, (ii) some abstracts are very short, and these tend to be of low quality, (iii) the weighted accuracy measure we use might produce different results. We tested this by taking the two best performing classifiers from our previous experiments (({1, ···, 5}-char RANK(SF, all) and RANK(SR, all))) and applied them to (i) full Wikipedia articles, (ii) only abstracts with at least 150 characters, and (iii) computed both weighted and un-weighted accuracy measures. Besides testing on only the 8 languages described in [6], we also computed weighted and un-weighted accuracies for the length-restricted abstracts of the 52 languages from Table 1. Restricting to abstracts of minimum length eliminated about 71.5% of the Wikipedia abstracts. The results are shown in Table 7. We observed that both the weighted and un-weighted accuracies are in the same range to those from [6]. This is a dramatic improvement over the previous results we reported in Tables 3 and 6. The false negative rate was reduced by 99%. We attribute this to the longer average size of each document tested. Using the full Wikipedia articles does not provide an improvement over the length-restricted abstracts. Furthermore, there is no statistically significant difference between the 8 and 52 languages. Finally, the un-weighted accuracy is lower but inline with [6]. So considering only documents with at least 150 characters when building the language profile shows significant improvement.

Length of Test Documents: To examine whether classifier accuracy is indeed affected by the length of test documents, we constructed a synthetic collection

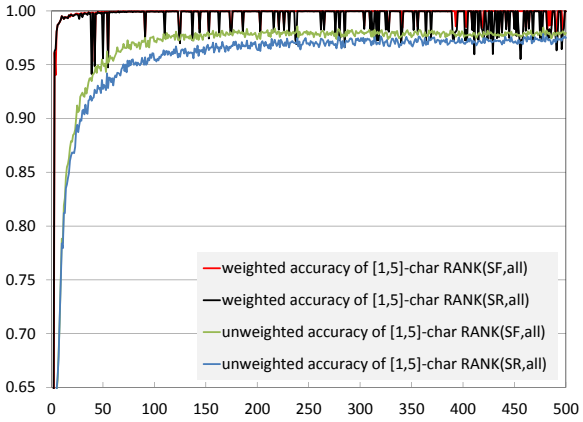

Fig. 2. Accuracy as a function of test document size in characters

of test cases where we control the size of the documents. For a language ℓ with n_ℓ Wikipedia abstracts, we generated $0.1n_\ell$ documents consisting of consecutive words drawn from the concatenated wikipedia abstracts of ℓ. The lengths of these synthetic documents is uniformly distributed in the range of 1 to 500 characters per language. We tested the performance of the two best classifiers – $\{1, \cdots, 5\}$-char RANK(SF, all) and RANK(SR, all) – on this collection, using the 52 languages from Table 1. Figure 2 shows the weighted and un-weighted accuracies for the classifiers. We observe that the weighted accuracy of both classifiers reaches 99% at around 9 characters average document length. The un-weighted accuracy grows more gradually, approaching 98% at the 500 character limit. The curves show some noise which is due to the limited sample size – for example, less popular languages have just 10 test cases per bucket.

4.2 Short Colloquial Postings and Tweets

Synthetic Tweets: We assembled a synthetic collection with a word count distribution that follows that of Twitter. We used all 52 languages and again $0.1n_\ell$ documents per language. The Twitter word count distribution was based on one month worth of real tweets, with Twitter commands, usernames, emoticons and URLs removed. 12.5% of cleaned tweets contained a single word and 64.5% contained at most ten words. Table 8 shows the weighted accuracy for all our classifiers applied to this synthetic data set. We observe that the two best classifiers – $\{1, \cdots, 5\}$-char RANK(SF, all) and RANK(SR, all) – still perform best. However, when comparing with the results of Table 3, RANK(SR, all) improves slightly whereas RANK(SF, all) deteriorates slightly.

Real Tweets from Tromp and Pechenizkiy [17]: The experiment uses the 9066 labeled tweets made available by [17]. They extracted tweets from accounts known to only contain messages in a specific language, namely German, English,

Table 8. Synthetic documents using Twitter's word-length distribution

	{1,⋯,5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	94.05	89.78	81.40	87.55	86.27	78.39	82.56	79.38	72.96	VECTOR	76.87	66.06
RANK SR	93.03	89.05	80.47	84.03	83.68	77.45	79.07	76.60	71.23	LIKELY	88.80	

Table 9. Trained on Wikipedia and applied on tweets from [17]

	{1,⋯,5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	98.00	99.10	95.34	95.83	98.19	96.84	97.51	98.35	94.89	VECTOR	91.51	85.15
RANK SR	89.22	99.12	95.08	89.21	95.51	96.67	96.41	98.03	94.78	LIKELY	98.75	

French, Spanish, Italian, and Dutch. They also used six different accounts per language, to provide diversity (tweets from different accounts) and consistency (several tweets from the same account). The tweets in the corpus are all lowercase, thus for this experiment we only applied our classifiers with models that had case-folding. We limited the language profiles used by our classifier to the six languages found in the data set. The results are displayed in Table 9. As in our previous experiments the {1,⋯,5}-char RANK classifiers perform best. We observed the tweets in the test set are longer and relatively cleaner than average tweets, in fact the length is even higher than the median length of Wikipedia abstracts. Therefore, the numbers produced are inline with our findings on minimum 150 character length. Furthermore, we highlight that we are able to beat the results from [17]. Even though we used Wikipedia as the training data and our rank-based classifier, in contrast to their specialized Machine Learning approach trained and tested on this very same data set.

Real Tweets – Uniform Random Sample: We sampled 2000 random tweets from the Twitter stream using uniform distribution. We labeled them ourselves to the correct language resulting in 869 tweets that we could process confidently. We run the same set of language identifiers on the labeled tweets and report the results in Table 10. The two best-performing classifiers are {1,⋯,5}-char RANK(SF,10k) and {1,⋯,5}-char RANK(SR,10k). Compared to Table 9, the performance is worse overall. We attribute this to the tweets in this sample being shorter and much less pristine than those provided by [17]. The best way to illustrate this is with some examples: (a) "*u nah commmming ?*", (b) "*alreadyyyy!!!!!!!*", (c) "*omg im bussin up*". Note the use of letter repetition for emotion, single letter such as '*u*' for 'you', and what is becoming standard abbreviations such as '*omg*' for '*oh my god*'. These should be contrasted with the tweets from [17], which look like: "*egypts supreme military council to sack cabinet suspend both houses of parliament and govern with head of supreme court from reuters*". The performance difference is expected once we consider such differences in the sets of tweets. We feel the random sample of our tweets is representative of what to expect from average tweets. The results indicate that our language identifiers trained on Wikipedia do a good job in processing tweets.

Table 10. Trained on Wikipedia and applied on a random sample of real tweets

	{1, ··· , 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	81.73	87.90	72.57	81.20	82.47	70.55	79.40	80.28	67.81	VECTOR	53.61	37.11
RANK SR	74.58	87.89	74.24	67.44	72.51	69.61	74.13	77.77	66.00	LIKELY	79.60	

Tweets from an Uncommon Language with Latin Alphabet: To test the strength of our methodology, we chose to explore the performance of a classifier trained on Wikipedia and applied on Twitter, but this time on a language that had the following characteristics: (i) is written mostly in Latin characters (so the character set will not help the classification), (ii) was not on the top list in terms of data available for training (i.e., relatively obscure), and (iii) we have access to a native speaker capable of labeling the results. One language that met all these conditions was Romanian (with 170,000 abstracts – see Table 1). For this experiment we sampled 250 tweets that our classifier labeled as Romanian and that had more than 50 characters (to avoid too much ambiguity) and asked our native speaker to label them. We found that 85.2% (213 of 250) were indeed Romanian. While this result is based on a moderately sized sample, in combination with the previous experiments it provides evidence that classifiers trained on Wikipedia is generally helpful in automatically classifying tweets.

Boot-strapping Twitter Labels: As our experiments have demonstrated, Wikipedia is a great resource for building language profiles, however its corpus is missing idioms that are particular to social postings such as tweets (e.g., the examples given above). One way to address this problem is to train models using large amounts of labeled tweets which is an expensive proposition (when human provide the labels). We automatically (and inexpensively) created this labeled set using the following methodology: by selecting tweets containing a country code in their metadata, labeling each tweet with our best language prediction as "L", and verifying it comes from a country where the native language is indeed "L". To avoid over-representation of popular languages, we put a cap of 10 million tweets at most per language. We were able to generate 88.3 million labeled documents – pairs of tweet and corresponding language. The set of 52 languages from Table 1 was reduced to 26, as we can only capture the actively used languages for which we could extract location information from the tweets. Using these documents we constructed new language profiles and repeated the set of experiments on real tweets.

Table 11 summarizes the results for the tweets from Tromp & Pechenizkiy [17]. In comparison to Table 9, accuracy is higher across the set of models tested, and even the (previously) weaker performing algorithms perform great. This is truly an indication of the value of our method for generating boot-strapped labels – the large amount of automatically training data generated by our method boosts the accuracy of our relatively simple classifiers, perform better than the specialized approach in [17] and the relatively more complex approach in [11] which were tested on the same dataset.

Table 11. Boot-strapped Twitter labels, applied on tweets as in Table 9

	{1, ··· , 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	99.11	99.52	96.40	99.02	99.06	97.97	99.31	99.44	97.03	VECTOR	95.41	93.98
RANK SR	95.05	99.53	96.42	96.53	97.44	97.93	98.83	99.27	96.99	LIKELY	99.64	

Table 12. Boot-strapped Twitter labels, applied on tweets as in Table 10

	{1, ··· , 5}-char			{3}-char			{1}-word				{3}-char	{1}-word
	all	10k	500	all	10k	500	all	10k	500		all	all
RANK SF	96.23	95.37	89.07	93.85	95.05	90.22	92.80	94.25	90.06	VECTOR	84.55	79.16
RANK SR	93.95	95.44	88.78	87.99	90.27	90.83	90.25	92.02	88.73	LIKELY	95.09	

Table 12 summarizes the results for the random sample of tweets we obtained directly from the Twitter stream. As we discussed earlier, we believe this data set to be a closer representation of the average tweet. The results show a very high degree of accuracy. We note that our language identifiers are triggered on all tweets – we will always return our top language guess. Consequently, the overall best results have precision above 96% with 100% recall.

5 Related Work

Our study is inspired by the work on using n-grams for text classification and statistical language identification, like [5,7,8], and in particular Cavnar and Trenkle [6]. This work proposed to use character n-grams for values of n ranging from 1 to 5, to limit the number of n-gram/frequency pairs retained for each language to 400, to compare languages and documents using a rank-based approach, and specifically to use Spearman's footrule [16] to estimate rank correlation. Cavnar and Trenkle recognized 8 languages and trained on a total of 3713 samples (documents) from a Usenet group. They reported accuracy of 92.9% for language profiles of 100 tokens and documents of less than 300 characters, and accuracy of 99.8% for language profiles of 400 tokens documents over 400 characters in length. We achieved similar accuracy (99.82; see Table 7) using the same 8 languages but training and testing on Wikipedia. Furthermore, we did a more exhaustive study: we experimented on 52 languages, trained on over 18 million Wikipedia documents, and tested on both Wikipedia and Twitter documents with different characteristics. We found that language profiles should be one to two orders of magnitude larger than suggested by [6].

Our work compares a larger number of models and classifiers including those based on using words as tokens, and those that take the frequencies of the tokens to estimate probabilities, and use likelihood for discrimination. In that sense our work is closer in spirit to Grothe et al. [9]. Once again, our study differs from theirs in terms of scale: we recognize almost three times as many languages (52 vs. 18), use a much larger corpus for training and testing, and also expand

on the number of classifiers, the use of capitalization, and the length of the documents to be classified.[4] We are also able to explain their observed phenomena of restricting the number of tokens in the profile based on word models. Using all the words (as opposed to the top ones) will result in over-fitting.

More recently Majlis [12] proposed a study using n-grams with W2C (WebToCorpus) training corpora and various methods like SVM, naive Bayes, regression tree and their own YALI algorithm. The results show high precision for large documents but much lower precision for document sizes of 30 to 140 characters. In some cases the results are in the order of 70% precision for the smaller documents. The YALI method proposed in the paper produces the best results when using 4-grams and shows an in-model testing accuracy of around 93% for small documents. Conceptually, there are similarities between YALI and our approach in that we both use a variation of n-gram retrieval to determine the language of documents. Another important difference is in the testing methodology. We trained and tested on Wikipedia and also tested on real tweets. The authors of YALI trained and tested on a carefully curated W2C corpus. By comparison, we achieved significantly better accuracies on "dirtier" data.

Dunning in [8] considered using probabilistic methods to identify the language of a document including Markov chains and Bayesian inference. Our likelihood classifier is similar in spirit and has similar performance characteristics.

Tromp and Pechenizkly [17] explore a supervised machine learning approach to language identification in the context of Twitter. They made their corpus available, enabling us to benchmark our implementations on their test corpus. In comparing the tweets contained in their collection to a uniform random sample of tweets we collected and labeled, we found the tweets in their collection to be much "cleaner" than the average tweet: They are fairly long, tend to be grammatically well formed, and contain fewer Twitter-specific acronyms than we observe "in the wild". Tromp and Pechenizkly used the same corpus for training and for testing. Our classifiers are unsupervised, start with Wikipedia and can be extended with automatically generated Twitter labels, and produce better results on their test set.

Another approach that is similar to ours is the work described in [11]. Their approach is based on a combination of a naive Bayes classifier using n-grams as features, with separate feature selection for each use case. By contrast our approach relies on side information to automatically obtain labeled data, allowing us to use over 80 million Tweets for training – three orders of magnitude larger than the corpus used by [11]. Moreover, the corpus can be generated dynamically and therefore our approach adapts to changes in style and usage patterns. Finally, [11] report 94% accuracy on the dataset from [17] while our method yields accuracies above 99% on the same dataset.

The work in [18] explores the task of identification of very short text segments on the order of 5 to 21 characters, training on the Universal Declaration of Human Rights as a corpus. They explore this space at full scale in terms of the

[4] We note that in one of their evaluation they use 135 documents from Wikipedia.

number of languages, using 281 (we restricted to 52 because of limitations on the labeled data of the corpus). They also explore n-grams-based models, but because of their restrictions to very short text they have to explore smoothing techniques (which do not seem to be necessary for identification of longer objects). We did not find that expanding the set of languages has a deleterious effect, but we did find that accuracy is sensitive to document length.

Finally, there is recent work in identifying the language in coping with short idiomatic text, such as tweets. The authors in [4] propose to use both endogenous and exogenous information to improve classification accuracy. In addition to the text of a tweet, they use the posting history of the author and other users mentioned in a tweet and web page content referred in the message. The underlying idea is to increase the size of each document by leveraging the structure of Twitter messages. The results reported are in line with our findings in that an increase in document length will yield higher accuracy. The authors in [3] focus on tweets from low-resource languages. Their approach is to collect such tweets, label them via the use of Mechanical Turk and use a supervised learning algorithm to train a classifier. They compare against three readily available language identifier systems, including an implementation of Cavnar and Trenkle [6]. Like [4], they also incorporate some meta information, such as tweet authorship. In contrast, our classifiers are unsupervised, using Wikipedia and tweet location to boot-strap mass labeling, and perform in a higher accuracy range.

6 Conclusions

In this paper we focused on two aspects of language identification. (i) study the various algorithms and free parameters and offer a guide on what works well and what not, and (ii) determine a methodology to provide high quality language identification for short colloquial postings, e.g. Twitter.

For language identification in general we learned that: (a) rank based classifiers are both effective and efficient, (b) if memory-limited, we can obtain good results with the top 10k tokens in the language model, (c) case folding does not matter, (d) the number of languages makes little difference, if enough training data exists, and (e) the length of test documents make a big difference, with larger being easier to classify.

Specifically to Twitter postings we learned that (a) Wikipedia works great for a solid baseline language model, and (b) generating labeled data using a combination of Wikipedia classification and a Twitter specific signal, like location, allows us to boot-strap superior language models.

According to our knowledge, this work describes the best overall methodology for an automated, unsupervised, and scalable technique in language identification on short colloquial postings. Our implementation is available [1] as a building block for further research in the area of social analytics.

References

1. Automatic language identification tool, `http://research.microsoft.com/lid/`
2. Aggarwal, C.C. (ed.): Social Network Data Analytics. Springer (2011)
3. Bergsma, S., McNamee, P., Bagdouri, M., Fink, C., Wilson, T.: Language identification for creating language-specific twitter collections. In: Proc. Second Workshop on Language in Social Media, pp. 65–74 (2012)
4. Carter, S., Weerkamp, W., Tsagkias, E.: Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. Language Resources and Evaluation Journal (2013)
5. Cavnar, W.: Using an n-gram-based document representation with a vector processing retrieval model, pp. 269–269. NIST SPECIAL PUBLICATION SP (1995)
6. Cavnar, W., Trenkle, J.: N-gram-based text categorization. In: SIDAIR (1994)
7. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science 267(5199), 843–848 (1995)
8. Dunning, T.: Statistical identification of language. Technical Report MCCS-94-273, New Mexico State University (1994)
9. Grothe, L., Luca, W.D., Nurnberger, A.: A comparative study on language identification methods. In: Proc. of LREC (2008)
10. Lopez, A.: Statistical machine translation. ACM Comput. Surv. 40(3) (2008)
11. Lui, M., Baldwin, T.: landid.py: An off-the-shelf language identification tool. In: Proc. of ACL (2012)
12. Majliš, M.: Yet another language identifier. In: EACL 2012, p. 46 (2012)
13. McNamee, P.: Language identification: A solved problem suitable for undergraduate instruction. Journal of Computing Sciences in Colleges 20(3) (2005)
14. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2007)
15. Spearman, C.: The proof and measurement of association between two things. The American Journal of Psychology 15(1), 72–101 (1904)
16. Spearman, C.: Footrule for measuring correlation. The British Journal of Psychiatry 2(1), 89–108 (1906)
17. Tromp, E., Pechenizkly, M.: Graph-based n-gram language identification on short texts. In: Proc. of BENELEARN (2011)
18. Vatanen, T., Vayrynen, J., Virpioja, S.: Language identification of short text segments with n-gram models. In: Proc. of LREC (2010)
19. Wasserman, L.: All of statistics. Springer (2004)