Graph-Based N-gram Language Identification on Short Texts

Erik Tromp E.TROMP@STUDENT.TUE.NL
Mykola Pechenizkiy M.PECHENIZKIY@TUE.NL
Department of Computer Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

Keywords: language identification, classification, n-gram

Abstract

Language identification (LI) is an important task in natural language processing. Several machine learning approaches have been proposed for addressing this problem, but most of them assume relatively long and well written texts. We propose a graph-based N-gram approach for LI called LIGA which targets relatively short and ill-written texts. The results of our experimental study show that LIGA outperforms the state-of-the-art N-gram approach on Twitter messages LI.

1. Introduction

Motivation. The problem of language identification (LI) is interesting on its own. However, in many practical applications LI can be seen as one of the steps of some larger process. Accurate LI can facilitate use of background information about the language and use of more specialized approaches in many natural language processing tasks dealing with a collection or a stream of texts, each of which can be written in a different language. The multilingual sentiment analysis on social media would be a typical motivating example. In (Tromp, 2011), an extensive experimental study shows that the multilingual sentiment classification can be performed more accurately when the process is split into four steps; LI, part-of-speech tagging, subjectivity detection and polarity detection. This becomes possible because at each step after LI, models that utilize language specific knowledge can be applied. Obviously, if the language of some text is identified incorrectly then this error will effect the forthcoming steps of the multilingual sentiment analysis and thus compromise the use of the relatively complicated four step procedure. Therefore, it is highly desirable to minimize the error of LI. Consider another example of machine translation where the source text's language is not known. The translation can not even commence without first identifying this source language.

Numerous supervised machine learning methods have been proposed for LI (Luca et al., 2008; Kranig, 2006). Several experimental studies reported high accuracy results for different collections of relatively long texts with proper grammar. The most widely accepted approach is to use N-grams (Cavnar & Trenkle, 1994). It was shown to be almost 100% accurate for long texts. For example, in (Cavnar & Trenkle, 1994) the experimental results showed 99.8% accuracy on the collection of documents written in fourteen different languages. The results suggested that high accuracies can be achieve for texts having a text length of at least 400 characters. However when LI is done for documents shorter than 300 characters, accuracies start to decrease much faster (though still pertaining the level of 93%) with respect to relatively longer texts having at least 400 characters.

Over recent years, with the popularity of social media, including Twitter and social networks, and consequently social media data analysis like opinion mining, the need for accurate LI (but now on short and grammatically-ill text messages) has become well-motivated again.

Problem formulation and our approach. We consider LI as a supervised learning task, particularly plain single label multi-class classification. Given some historical or training data in which for each text t we know a label l, the language in which this text is written, our goal is to learn a model such that given some previously unseen text we can say as accurately as possible in which language this text is written. We do not consider cases when for a text written partly in one

Appearing in *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands.*

language and partly in some other language someone would like to get both labels as an output. We also do not consider language groups or any other dependencies between the language labels.

We introduce a Graph-based Approach for LI (LIGA) that allows to learn elements of grammar besides using N-gram frequencies.

Main results. The experimental study we performed with the collections of Twitter messages written in six languages shows that LIGA is statistically significantly more accurate than the existing N-gram based approach regardless of the training set size used to learn the models (95-98% for LIGA vs. 87-93% for N-grams) and that LIGA is less prone to overfitting the sources and domain-specific jargon.

Outline. The rest of the paper is organized as follows. In Section 2, we give an overview of the related work on LI discussing the applicability of different approaches to short, grammatically-ill texts. In Section 3 we introduce LIGA, our approach for LI. Section 4 describes the experimental study in which we compare LIGA with the traditional N-gram method. Finally, Section 5 summarizes our findings and suggests direction for future work.

2. Background and Related Work

In this section we discuss related work on LI with the focus on N-gram based approach as the current state-of-the-art for LI, which we use in LIGA.

2.1. N-Gram-Based Approach

The N-gram-based approach for LI (Cavnar & Trenkle, 1994) chops texts up in equally-sized character strings, N-grams, of length n. It is assumed that every language uses certain N-grams more frequently than other languages, thus providing a clue on the language the text is in. This idea works due to Zipf's law stating that the size of the r-th largest occurrence of the event is inversely proportional to its rank r (Ha et al., 2003). Experimental studies in (Cavnar & Trenkle, 1994) suggest that using trigrams (at the character level) generally yields the best results.

In (Cavnar & Trenkle, 1994) the out-of-placement measure is used to compare unlabeled text against the model. This measure sorts the N-grams in both the model as well as the unlabeled text separately based on their occurrence counts and compares the model's occurrence list with the text's list. Later, in (Ahmed et al., 2004) it was shown that the use of a cumulative frequency based measurement yields similar accuracy results yet is more time efficient. The out-of-placement measure works well when sufficient training data is available whereas the cumulative frequency measurement works equally well with little data at hand. Therefore we will use the cumulative frequency measurement in our experiments.

2.2. Other Approaches

The idea behind the N-gram-based approach is borrowed from (Dunning, 1994) where using Markov Models for LI was considered. This approach however lacks the intuition the N-gram approach has and requires more time for training a model and for classifying a new text.

A similar straightforward approach is to use word frequencies. One variant is to use short words (Prager, 1999) as they occur regularly in a language and usually differ per language. Another variant is to use the most frequently occurring words (Martino & Paulsen, 1996; Cowie et al., 1999) for the same rationale.

The compression-based approach for LI was proposed in (Harper & Teahan, 2001). Labeled data is compressed using so-called prediction by partial matching (PPM) approach to construct language models. An unlabeled text is also compressed and the number of bits required to encode this new document is compared to the number bits used in the language models. The likeliness of a text with a language model is computed using entropy as a similarity measurement.

3. LIGA - Graph-Based N-gram Language Identification

In our approach we want to utilize not only word presence and occurrences but also their ordering. To capture the ordering of words, we create a graph model on labeled data. The labels of its vertices represent the presence of words in a given language. The weights of the vertices represent the frequencies of words in a given language. The crucial part is in the presence and weights of the edges, which try to capture the grammar (in this particular case only the word ordering) of a language.

As a starting point for our method, we use the N-gram-based approach described in Section 2.1. We will thus not truly capture word information but N-gram information. Next, we give the preliminaries (Section 3.1), the methodology to learn LIGA and to use it for the model (Section 3.2) and to classify unlabeled texts (Section 3.3), and time and space complexity analysis (Section 3.4).

3.1. Preliminaries

We extend a basic graph $G = (V, E)$ with a labeling function $\mathcal{L} : V \to L$. This labeling function assigns to each vertex $v \in V$ a label $l \in L$ uniquely identifying the vertex. Let *Lang* denote all languages present in our training set, then the function $\mathcal{W}_v : V \times Lang \to \mathbb{N}$ assigns for each vertex $v \in V$ and every language $l \in Lang$ a weight. For edges we have a similar function $\mathcal{W}_e : E \times Lang \to \mathbb{N}$. Since we will incrementally construct our graph, we may encounter that, for a label l of a vertex u to be added, $\exists_{v \in V} : v \neq u \wedge \mathcal{L}(v) = l$. We then say that $\mathcal{L}(v)$ is *defined*. We say $\mathcal{L}(v)$ is *undefined* in all other cases. We use the same notion of *defined* for $\mathcal{W}_v$ and $\mathcal{W}_e$.

Using the mentioned extensions, we represent a graph as the following quintuple.

$$G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$$

A labeled text t of which the language l is known is denoted as the binary tuple (t, l). An unlabeled text s of will be denoted as the binary tuple (s, λ). We denote all N-grams of a text t as the ordered list $N_{n_t} = [g_1, g_2, ..., g_k]$ where n denotes the length of the N-grams. The order in N_{n_t} respects the order in which the N-grams occur in (t, l).

3.2. Learning a Model

Our goal is given a training set $\mathcal{T}$ consisting of labeled texts for every language in *Lang* to learn a model consisting of a single graph G.

For each text $(t, l) \in \mathcal{T}$ we construct the list N_{n_t}. For every $m \in N_{n_t}$ we create a vertex v with $\mathcal{L}(v) = m$, but only if $\mathcal{L}(v)$ is *undefined*. For every $e \in \{(u, v) \in V \times V : (\mathcal{L}(u) = m_i \wedge \mathcal{L}(v) = m_{i+1}) \Rightarrow (m_i, m_{i+1} \in N_{n_t})\}$ we create an edge e, but only if $e \notin E$. The weights are updated as follows:

$$\mathcal{W}_v(v, l) = \begin{cases} \mathcal{W}_v(v, l) + 1 & \text{if } \mathcal{L}(v) \text{ is } \textit{defined} \\ 1 & \text{otherwise} \end{cases}$$

$$\mathcal{W}_e(e, l) = \begin{cases} \mathcal{W}_e(e, l) + 1 & \text{if } \mathcal{W}_e(e, l) \text{ is } \textit{defined} \\ 1 & \text{otherwise} \end{cases}$$

When we add a node or edge (i.e. when $\mathcal{L}(v)$ or $\mathcal{W}_e(e, l)$ is *undefined* respectively), we initialize the weights of all languages for that vertex or node to 0 before applying the weight updates. When applying the aforementioned definitions for all $(t, l) \in \mathcal{T}$ we get our graph $G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$. We illustrate this by an example. Consider the following two texts of which the first is in Dutch (NL) and the second is in English (EN).

$$(t_1, NL) = \textit{ is dit een test}$$
$$(t_2, EN) = \textit{ is this a test}$$

We then first create the ordered lists of N-grams. When using trigrams ($n = 3$) we get the following, where a space is denoted by a dot ·.

$N_{3_{t_1}} =$ [is·, s·d, ·di, dit, it·, t·e, ·ee, een, en·, n·t, ·te, tes, est]

$N_{3_{t_2}} =$ [is·, s·t, ·th, thi, his, is·, s·a, ·a·, a·t, ·te, tes, est]

We next start constructing the graph. For each $n \in N_{3_{t_1}} \cap N_{3_{t_2}}$ we add a vertex v to our graph, having $\mathcal{L}(v) = n$. For example, for the first element in $N_{3_{t_1}}$ we will create the vertex v having $\mathcal{L}(v) = is\cdot$. For the first element in $N_{3_{t_2}}$ we will not add a new vertex as $\mathcal{L}(v)$ is defined. In our example, for vertex v (having $\mathcal{L}(v) = is\cdot$) we will have $\mathcal{W}_v(v, NL) = 1$ and $\mathcal{W}_v(v, EN) = 1$ as $is\cdot$ occurs once in both the Dutch as well as the English text.

We next add edges. We will have edges from for example v to u ($e = (v, u)$) where $\mathcal{L}(v) = is\cdot$ and $\mathcal{L}(u) = s \cdot d$, capturing the order between the first and second elements of $N_{3_{t_1}}$. Since this connection occurs only once, and only for the Dutch text, we have that $\mathcal{W}_e(e, NL) = 1$ and $\mathcal{W}_e(e, EN) = 0$.

Figure 1 shows the graph resulting from this example. The labels of the vertices are shown at the topmost position inside the vertices. The weights per language are listed with the vertices and edges.

3.3. Classifying a Text

Once we have constructed G, we can use it to classify unlabeled texts. To do so, we first need to transform an unlabeled text into something similar to G such that we can compare the two.

While for constructing G we use weights to indicate multiple occurrences of a given N-gram, for the unlabeled text we create multiple vertices – one for every occurrence. We thus in fact get a simple graph, a path $\pi = (V, E, \mathcal{L}, v_{start})$. Here $|V| = |N_{n_t}|$ and if $N_{n_t} = [n_1, n_2, .., n_k]$ then $E = \{(u, v) | \mathcal{L}(u) = n_i \wedge \mathcal{L}(v) = n_{i+1}\}$. The node $v_{start} \in V$ is the starting node of our path. To illustrate this, we consider the following (Dutch) text *is dit ook een test* of which we would not know the label in advance. The path π for this example is shown in Figure 2.

A crucial step is to compare π and G. We compute the so-called *path-matching scores* for each language $l \in Lang$. Conceptually, this is done by 'laying' the path over the graph and measuring its similarity for each language. Since all languages we have knowledge

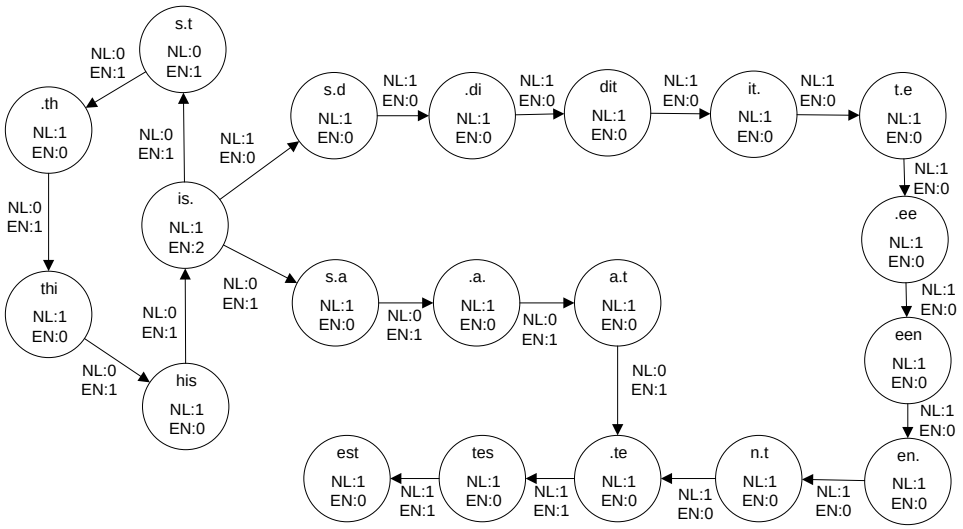

Figure 1. The graph resulting from the example

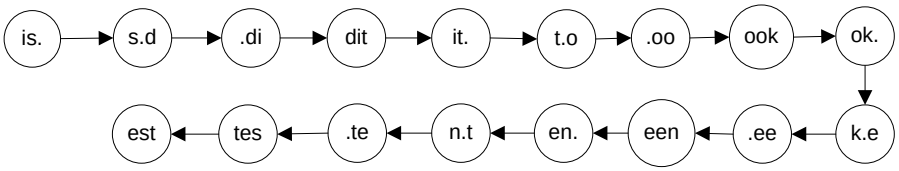

Figure 2. The path resulting from the unlabeled example

of are present in a single model, we can compute the scores for all languages in one go.

To keep track of the matching scores of our languages, we maintain a scoring function $\mathcal{PM} : Lang \to \mathbb{R}$ assigning a rational number to every language $l \in Lang$. Initially, we say that $\mathcal{PM}(l) = 0$ for all $l \in Lang$. Let $G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$ be the graph representing our model and $\pi = (V', E', \mathcal{L}', v_{start})$ be our labeled path. In order to keep our method robust with respect to differing quantities of labeled data for different languages, we will normalize the scores. Let $\Sigma_v = \Sigma_{v \in V}(\Sigma_{l \in Lang}(\mathcal{W}_v(v, l)))$ be the total sum of all weights contained in all nodes in G. Also, let $\Sigma_e = \Sigma_{e \in E}(\Sigma_{l \in Lang}(\mathcal{W}_e(e, l)))$ be the sum of all weights contained in all edges in G. We then start traversing our path π in G starting with node v_{start}. Let v'_i denote the node we are currently visiting (initially, $v'_i = v_{start}$). We try to find a $v \in V$ such that $\mathcal{L}(v) = \mathcal{L}'(v'_i)$. Note that there is at most one such node but there may be none. We update the matching scores according to (1). In this step we account for the occurrence of a given N-gram from π in G.

The next step is to account for the order. We find the only edge $e' \in E'$ that has our previous node v'_i as source, if any (since the last node has no outgoing edges). We thus find the edge $e' = (v'_i, v'_{i+1})$. We then update the matching scores according to (2).

We have now accounted for the order between two N-grams present in our path whose label is given in nodes v'_i and v'_{i+1}. We next continue traversing our path by performing (1) again for v'_{i+1} after which we can apply (2). This process continues until we find a node $v'_{end} \in V'$ such that $\neg(\exists_{v' \in V'} : (v'_{end}, v') \in E')$; this is the ending node not having any outgoing edges.

When we have matched the entire path onto the graph, we need our function $\mathcal{PM}$ to determine the language. For each language, $\mathcal{PM}$ maps it onto a value in $\mathbb{R}$. More specifically, since the weights of the nodes accounted to $\mathcal{PM}$ are normalized and so are the weights of the edges, we end up with a score in $[0, 2]$ (when our model is isomorphic to our path for a given language, we get a score of 1 for the nodes and 1 for the edges). We now say that the text of which we were identifying the language is in language $l = argmax_{l \in Lang} : \mathcal{PM}(l)$.

$$\forall_{l \in Lang} : \mathcal{PM}(l) = \begin{cases} \mathcal{PM}(l) + \frac{\mathcal{W}_v(v,l)}{\Sigma_v} & \text{if } \exists_{v \in V} : \mathcal{L}(v) = \mathcal{L}'(v'_i) \\ \mathcal{PM}(l) & \text{otherwise} \end{cases} \quad (1)$$

$$\forall_{l \in Lang} : \mathcal{PM}(l) = \begin{cases} \mathcal{PM}(l) + \frac{\mathcal{W}_e(e,l)}{\Sigma_e} & \text{if } \exists_{e \in E} : (\exists v, w \in V : \mathcal{L}(v) = \mathcal{L}'(v'_i) \wedge \\ & \quad \mathcal{L}(w) = \mathcal{L}'(v'_{i+1}) \wedge e = (v, w)) \\ \mathcal{PM}(l) & \text{otherwise} \end{cases} \quad (2)$$

3.4. Time and Space Complexity of LIGA

Let $\mathcal{T} = \{(t_1, l_1), (t_2, l_2), .., (t_m, l_m)\}$ represent all texts t_i with their respective labels l_i of our training set. For every t_i we create its N-grams which can be done in $\mathcal{O}(|t_i|)$ time, i.e. linear in the size of the text. As we create the N-grams, we add them to our graph in parallel by keeping track of the previous N-gram for the edge addition. The time required to build G is hence $\mathcal{O}(|t_i|)$ for a single text t_i. Let now $t_{max} = argmax_{t_i \in T} : (\forall_{t_j \in \mathcal{T}} : |t_i| \geq |t_j|)$ be the longest all texts in $\mathcal{T}$. The total time required to train a model is bound by $\mathcal{O}(m \cdot |t_{max}|)$, where m is the number of texts in $\mathcal{T}$, which is asymptotically equivalent to the regular N-gram-based approach. Note however that the constant for LIGA is greater than for the N-gram-based approach.

When we classify (t_{new}, λ) with G, we again first create the N-grams in $\mathcal{O}(|t_{new}|)$ time, forming a path. We then traverse the path in our graph. Since for every N-gram in our path we have to check at most one node in G (namely, the node having the same label) and at most one edge, this is also linear in the size of the unlabeled text. Classifying a text hence requires $\mathcal{O}(|t|)$ time.

For the space required to store our model, the worst case would be when we have a completely non-overlapping training set. Every text in such a training set would yield a single string or path that is disconnected from all other paths. Since such a single path represents a single text t, we have one node for each N-gram of the text, thus requiring $\mathcal{O}(|t|)$ space. We will have exactly $|t| - 1$ edges since the starting N-gram has no incoming edges and the ending N-gram has no outgoing edges. The space required to store the edges is hence also linear in the text's size, requiring $\mathcal{O}(|t|)$. Using similar notation as with the time complexity, we need $\mathcal{O}(|l| \cdot m \cdot |t_{max}|)$ space to store the model. The $|l|$ component is the number of languages present in our model and originates from the fact that we need to store $|l|$ weights for each node and edge. Similar reasoning shows that we require $\mathcal{O}(|t|)$ space to store the path representing unlabeled data.

4. Experimental evaluation

We compare the performance of the proposed LIGA approach with the N-gram-based approach for LI. Particularly, we are interested in the following three aspects; (a) comparing the accuracy of two approaches on the LI of *short texts* (Twitter messages), (b) looking into the effect of reducing the amount of training data on the accuracies, and (c) comparing the generalization and (over)specialization properties of two approaches with respect to the source, domain or jargon the present in the training data.

In their work (Cavnar & Trenkle, 1994) use newsgroup articles having culture as topic. The messages in their dataset hence all share the same domain. The generalization/specialization experiments address this problem.

In the following sections we first describe the dataset used in this study and the experiment setup, and then present the main results.

4.1. Dataset and Experiment Setup

The dataset was constructed from the social medium *Twitter*. The Twitter API is used to extract messages from accounts known to only contain messages of a specific language. We do this for six languages and six accounts per language. The six languages are German, English, Spanish, French, Italian and Dutch. These are languages we have sufficient knowledge of to identify. Moreover, including Spanish, French and Italian presents a challenge as these languages contain a lot of similar word extensions and trigram patterns. For every language we have at least one account of a person instead of an institution (such as BBC News). We assume that each account has its own domain and hence its own jargon that typically is not, or less often, used in the other domains. When we indicate a domain as *random* we mean that its messages are a mixture of other domains. The rationale behind using six accounts per language is that our experiments require us to have different domains but incorporating tens or hundreds of accounts is very laborious.

As a pre-processing step, we inspect our data as in (Cavnar & Trenkle, 1994), removing messages that

Table 1. Splitting the data into training and test sets.

Lang.	Domain	Exp. 1	Exp. 2
DE	Sports	s_1 ... s_n → $train_1$, $test_1$	$test_{2_{gen}}$
	National news	n_1 ... n_n → $train_1$, $test_1$	$train_2$, $test_{2_{spec}}$
	Random	r_1 ... r_n → $train_1$, $test_1$	$test_{2_{gen}}$
NL	Telecom.		
...	...	...	...

Table 2. Accuracies averaged over 50 runs.

Experiment	LIGA	N-gram	t-Test
5% sample	94.9 ± 0.8	87.5 ± 1.5	√
10% sample	96.4 ± 0.5	90.6 ± 1.0	√
25% sample	97.3 ± 0.5	92.5 ± 0.9	√
50% sample	97.5 ± 0.5	93.1 ± 0.8	√
5% vs. 50%	94.9 ± 0.8	93.1 ± 0.8	√
Generalization	92.4 ± 1.0	83.5 ± 1.8	√
Specialization	98.3 ± 0.8	95.5 ± 1.6	√
One holdout	95.6 ± 1.6	89.2 ± 4.3	√
Two holdouts	95.2 ± 0.9	88.1 ± 2.7	√

contain multiple languages or bilingual terminology. From each message we also deliberately remove links, usernames preceded by an @ sign, term references preceded by a # sign or smilies such as *:)* and punctuation. The rationale behind this is that we want to learn a model on the language itself whereas these entities are language-independent. Moreover, the use of Twitter for our experiments is just to show an application on short texts, learning Twitter-specific patterns containing username or channel references is not desired. The final dataset contains 9066 labeled messages of at most 140 bytes.[1]

In both approaches we use trigrams (as suggested in (Cavnar & Trenkle, 1994)) and the frequency based measurement of (Ahmed et al., 2004).

In all cases we compare the mean accuracy of the N-gram approach against the mean accuracy of LIGA. The mean accuracies are obtained by taking the mean of 50 different ten-fold cross-validation experiment runs. We check for statistical significance using pairwise T-tests.

In order to compare the accuracies of both methods, we learn a model on one part of the data, which we call $train_1$, and test it on another part of the data, called $test_1$ formed as illustrated in Table 1, column Exp. 1. We use all accounts of all languages. The size of $train_1$ varies to investigate the influence of the corpus' size. We use 5%, 10%, 25% and 50% of the entire dataset stratified per language and sampled uniformly.

We also investigate how much each approach learns about the actual language itself rather than about a particular domain. To analyze this, we learn a model on data from one domain and test it on other domains. For each language, we choose a single account on which we learn our model and then test on the remaining accounts (Table 1, column Exp. 2). The training set $train_2$ consists of $\frac{2}{3}$ of all data of one single account for each language. There are now two test sets. The first, $test_{2_{spec}}$ is the remaining $\frac{1}{3}$ of the same account and will allow us to judge how specific each model is for a single domain. The second, $test_{2_{gen}}$ consists of all other accounts. This test set will show us how well the learnt model generalizes to other domains.

Finally, we analyze the influence of jargon. To study this, we learn a model on all accounts of all languages except for one or two hold-out accounts which are reserved for testing. Assuming that each account represents a single domain, we can get to know how important it is to capture domain-specific jargon since any domain-specific jargon present in one of the holdout accounts likely will not be included in our model. The formation of the training and test datasets can be seen as the inverse of Table 1, column Exp. 2.

4.2. Experiment Results

The main results are presented in Table 2 which shows, averaged over 50 experiment runs, the accuracies and standard deviations for LIGA and N-gram approaches. The last column shows the result of pairwised T-test (which was positive for each comparison in our case).

Effect of the training set size. As expected we can see that as the size of the training data grows, the accuracy increases and variance decreases with both approaches. However, LIGA has much better performance than N-gram approach especially when a small amount of labeled data is available.

The row '5% vs. 50%' of Table 2 compares LIGA

[1]The dataset is made available online at `http://www.win.tue.nl/~mpechen/projects/smm/`

learned on 5% of the training data against the N-gram-based approach using 50% of the training data. LIGA still has statistically significant higher accuracy.

It can be clearly seen from Figure 3 that LIGA *consistently* outperforms N-gram approach on each of the 50 experiment runs. To avoid overloading of the graph we present only accuracies corresponding to the use of 5% and 50% of available labeled data.

Using more than 50% of labeled data for training a LI model in our case did not result in any further significant improvement of the classification accuracy for either of the two approaches and therefore we omit these results for the sake of conciseness.

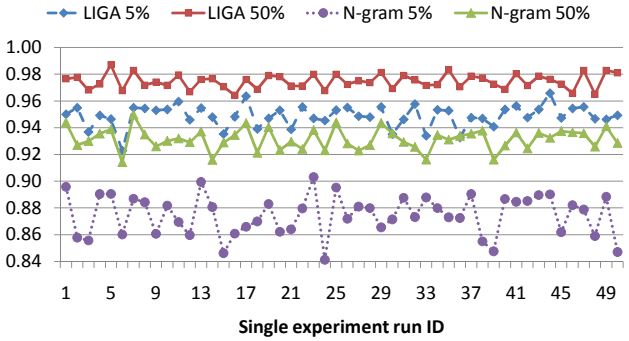

Figure 3. Accuracy results over 50 different runs for LIGA and N-gram approaches when trained either on 5% or on 50% of available labeled data.

Generalization and domain-specific jargon results. The 'Generalization' and 'Specialization' rows in Table 2 and Figure 4 show the results from the generalization experiments. As one would expect, a model learned on a specific domain yields higher accuracies in that domain than in other domains. This behavior is shown by the difference in accuracies between the generalization results and the specialization results. The specialization accuracy for both approaches is often close to 100%, yet LIGA is statistically significantly better than N-gram approach.

For the generalization experiments, we clearly see that LIGA again consistently (Figure 4) outperforms the N-gram based approach. The accuracy of LIGA never drops below 90% whereas that of the N-gram based approach is never higher than 90%.

LIGA achieves higher accuracies both within the domain the model was trained on as well as outside that domain. This indicates that LIGA is likely to be less sensitive to domain-bound usage of language and hence is likely to learn more about the language

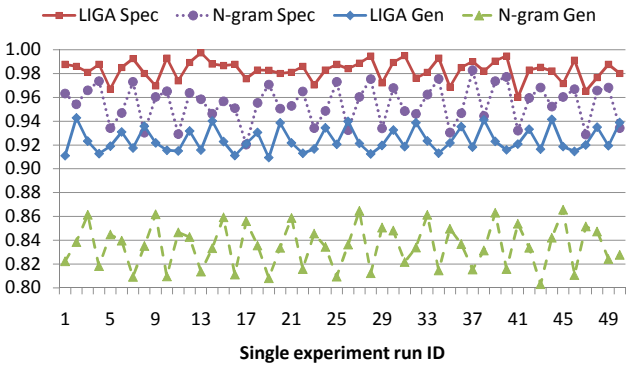

Figure 4. Accuracy results over 50 different runs for LIGA and N-gram approaches when trained on one domain and tested on all other domains (see Table 1, column Exp. 2 for clarification).

itself rather than about the domain messages belong to.

The results plotted in Figure 5 stem from holding out one or two accounts. Averaged results are in the last two rows of Table 2. We can observe a moderate drop in accuracy for both approaches with respect to not holding out any account. LIGA's accuracy drops by 2-3% on a single holdout and the N-gram-based approach's accuracy drops by 4-5%. Testing on two holdout accounts decreases accuracy a bit further but not much. This indicates that the use of domain-specific jargon introduces an error of around 3% for LIGA and 5% for the N-gram-based approach. LIGA consistently outperforms the N-gram-based approach.

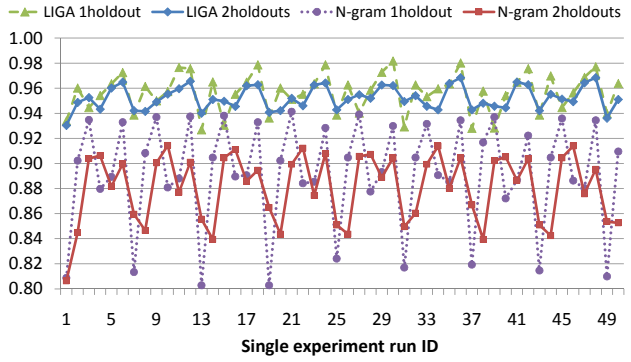

Figure 5. Accuracy results over 50 different runs for LIGA and N-gram approaches with one and two hold out accounts.

We observe that for some holdout accounts the accuracy is always lower than for others. There are thus some domains that use more jargon than others. How-

ever, across multiple experiment runs, the variance in accuracy is always much larger for the N-gram approach.

5. Conclusions

In this paper we have studied the problem of language identification on relatively short texts typical for social media like Twitter. Earlier works on language identification showed promising and highly accurate results for well-constructed, sufficiently long enough texts. However, the results were shown to deteriorate considerably when texts become a few hundred characters long.

We proposed LIGA – a graph-based approach that aims to capture the elements of language grammar. The experimental results confirmed our expectation that for short texts LIGA outperforms the current state-of-the-art N-gram approach for language identification, showing consistently higher accuracy results even with an order less labeled data used to learn the model from. Besides that, our experiments suggested that LIGA is less likely to be sensitive to use of jargon or to domain boundaries.

Future Work. LIGA currently captures only one aspect of grammar, the order of words or trigrams at the character level. It would be interesting to see what other grammatical aspects can aid to the problem of language identification. One such aspect may be to incorporate words that can (or often do) start sentences. Our method can be easily extended to use this information by computing N-grams that begin some texts and taking them into account in path-matching.

We evaluated the performance of our approach only on short texts extracted from Twitter with respect to the goals of this study. However, it would be interesting to compare these results with results obtained from using longer texts or texts extracted from other sources. In addition to regarding other sources, using more data and especially incorporating more languages, gives stronger results and a broader comparison.

Language identification is typically part or one step of a bigger process. The error made in this step is hence propagated along the pipeline of remaining tasks. It is Interesting to study how severely this error affects succeeding steps. In (Tromp, 2011) we show how to quantify this effect for the case of the multilingual sentiment analysis on social media.

An open issue not addressed in related work on language identification is dealing with an absence of certain N-grams or words that influences the certainty of classification. When an unlabeled text contains a number of N-grams not present in a learned model, the learned model will not be confident about the label. This suggests assigning confidence scores to the assigned labels that can be utilized in the further natural language processing routine or in the mechanism suggesting updates to or relearning of the language identification models.

References

Ahmed, B., Cha, S., & Tappert, C. (2004). Language identification from text using n-gram based cumulative frequency addition. *Proc. CSIS'04.*

Cavnar, W., & Trenkle, J. (1994). N-gram-based text categorization. *Proc. 3rd Symp. on Document Analysis and Information Retrieval (SDAIR-94).*

Cowie, J., Ludovic, Y., & Zacharski, R. (1999). Language recognition for mono- and multilingual documents. *Proc. of the Vextal Conference.*

Dunning, T. (1994). Statistical identification of language. *TR-MCCS-94-273, New Mexico State Univ..*

Ha, L., Sicilia-garcia, E., Ming, J., & Smith, F. (2003). Extension of zipfs law to word and character n-grams for english and chinese. *Journal of Computational Linguistics and Chinese Language Processing.*

Harper, D., & Teahan, W. (2001). Using compression-based language models for text categorization. *Proc. Workshop on Language Modeling and Information Retrieval.*

Kranig, S. (2006). Evaluation of language identification methods. *Proc. ACM SAC 2006.*

Luca, E. D., Grothe, L., & Nuernberger, A. (2008). A comparative study on language identification methods. *Proc. 6th Conf. on International Language Resources and Evaluation.*

Martino, M., & Paulsen, R. (1996). Natural language determination using partial words. *US Pat. 6216102 B1.*

Prager, J. (1999). Linguini: Language identification for multilingual documents. *Proc. 32nd Hawaii Int. Conf. on System Sciences.*

Tromp, E. (2011). Multilingual sentiment analysis on social media. Master's thesis, Dept. Computer Science, Eindhoven University of Technology.