Language Identification for Text Chats

Vesa Siivola, Bryan Pellom and Meagan Sills

Rosetta Stone Labs, Boulder, Colorado, USA

vsiivola@rosettastone.com, bpellom@rosettastone.com, msills@rosettastone.com

Abstract

This work aims to classify the language of typed messages in a text chat system used by language learners. A method for training a language classifier from unlabeled data is presented. A dictionary-based method is used to produce initial classification of the messages. Character based n-gram models of order 3 and 5 are built. A method for selectively choosing the n-grams to be modeled is used to train 15-gram models. This method produces the best-performing classifier. It has models for 57 languages and obtains over 95% accuracy on the classification of messages that are unambiguously in one language.

Index Terms: language identification, n-grams, variable-order models

10.21437/Interspeech.2011-733

1. Introduction

The problem of automatic language identification for written text is a well-researched topic [1]. The subject of this study, corpus of messages from a text chat for language learning, poses interesting challenges for language identification. The messages are short, they may be ungrammatical and contain spelling errors, they may contain words from different languages and the script of the language may be romanized in several ways. In this paper, we will show how to build a language identification system based on character n-grams from an unlabeled training corpus. One possible application for a text chat message classification system would be in language learning: the teacher could monitor in which proportion the students are using different languages in the task that has been assigned to them and how long they stay on the task. We will not discuss approaches to text-based language identification that are not n-gram based (e.g., advanced dictionary-based methods [2], string kernels [3], decision trees [4]).

1.1. Prior work

Beesley observed that the probability distribution of character 2-grams is different for all languages and can be used to identify the language [5]. The paper does not describe the language classifier in great detail. Cavnar and Trenkle suggest that for each language, a list of n-grams seen in the training set for all orders up to a given order be constructed (the full list of order 5 would contain 1-grams, 2-grams, ..., 5-grams) [6]. The list is then ranked by frequency of appearance. The same procedure is done for all languages. The text of an unknown language is processed in the same way and the ranking of the n-grams is compared to the trained lists. The best-matching list is the recognized language. Cordoba et al. use this kind of method for a phonotactic model of language identification system for audio segments [7]. Grefenstette calculates the probabilities of all trigrams that have appeared more than 100 times in the training set and uses this as a basis for estimating the most probable language of an unknown document [8]. He also shows that small words such as conjunctions can be used for language identification. Similarly, McNamee and Mayfield use character n-grams as search terms in information retrieval [9]. Teahan uses Prediction by Partial Match to create character-based Markov models for several languages [10]. The language model that gives the highest probability (lowest cross-entropy) to the unknown text indicates the recognized language. Vatanen et al. compare the baseline method by Cavnar and Trenkle to a character based n-gram modeling that uses modern smoothing techniques [11]. They show that the n-gram based techniques outperform the baseline. This paper uses an approach that is similar to the n-gram based techniques presented by Vatanen et al.

2. Method for training the classifier

The training of the language identification system begins with producing a labeled set of training samples from the unlabeled data with a dictionary-based classifier. This set is used to train the initial n-gram models. The n-gram models are then used to produce a new labeled training set for the next iteration of n-gram training. The iteration is finished when the performance of the classifier no longer increases for the development data set.

2.1. Initialization with dictionaries

The object is to create a labeled text corpus, from which the first iteration of character-based n-gram models can be trained. Each message of the training set $\mathcal{M} = \{w_1, \ldots, w_N\}$ was tested against all of the available dictionaries $\{d_1, \ldots, d_O\}$ and the number of words matching a dictionary was recorded. Since there were not dictionaries for all languages and the most well-known missing ones (e.g., Chinese, Japanese, Korean) were not based on Latin alphabet we also calculated the ratio of non-ASCII characters c_{na} to all characters c. This was thought to reflect the probability, that the message was in one of the languages for which we did not have a dictionary. The result was scaled to work with the results from the dictionaries: if all characters were non-ASCII, that would correspond to having 3 words match the dictionary of a language. This seemed to be a reasonable heuristic. The resulting count would be the score $s(\mathcal{M}, d_l)$ for the language l:

$$s(\mathcal{M}, d_l) = \begin{cases} |\{i : w_i \in d_l\}|, & \text{if } \exists d_l \\ 3c_{\text{na}}/c, & \text{otherwise} \end{cases} \quad (1)$$

When creating the initial labeled data set, we used a simple heuristic confidence function to select only the data that the dictionary-based classifier was likely to get right. The rest of the data was discarded. For each message in Russian, Ukrainian or Bulgarian, a romanized version of the same message was added to the training set. We would have liked to do the same for languages like Arabic, Japanese and Chinese, but did not have tools to romanize them properly.

Copyright © 2011 ISCA

2.2. Full n-gram models

Among the methods that are often used for language modeling in speech recognition systems, an interpolated modified Kneser-Ney smoothed n-gram model seems to give the best results [12]. There exist other methods that can match that performance [13] or even outperform it [14], but they require significantly more computational resources. In this work, the full character n-gram models are trained with interpolated modified Kneser-Ney smoothing. The n-gram model of the language that gives the highest probability with respect to the message determines the language label assigned to the message.

2.3. Variable order n-gram models

A full n-gram model stores estimates for the probabilities of all n-grams that are found in the training text up to the given maximum order. One problem with this approach is that the memory consumption of both the training algorithm and the actual model increases almost exponentially with the order of the model. The size of the resulting model can be decreased by pruning away the n-grams that do not have much effect on the performance of the model. The memory consumption of the training algorithm can be decreased by choosing to explicitly model only a subset of possible n-grams before pruning. The growing and pruning methods can be combined in such a manner that they produce variable-order models which have similar smoothing characteristics to the Kneser-Ney smoothing for full models [15]. This is the method that is used in the experiments.

The basic idea of the growing algorithm is that we start from an unigram model and consider adding new distributions to the n-gram model. If the distribution increases the modeling accuracy on training data sufficiently compared to the increase in the model size, it is accepted. The distributions to be added are sampled based on what already is in the model: take an n-gram from the model and try adding a distribution for all words following that n-gram. When the model is grown, we modify the existing distributions so that they fullfil the marginal property that inspired the Kneser-Ney smoothing. For pruning, we look at each n-gram and see if removing the n-gram reduces the model size enough in comparison to the loss of modeling accuracy. This is similar to entropy pruning [16], except the estimates used in pruning have been modified to be more accurate for Kneser-Ney smoothed models. Again, after each pruning operation we need to adjust the existing distributions to fullfil the marginal property of Kneser-Ney smoothing.

3. Experiments

3.1. Data

The training data consisted of 270M chat messages containing 1.1G words (5.4G characters) collected from a language learning site. The average length of a message is 20 characters. Each participant in the chat had been asked to list the languages he knows. This information was not completely reliable and based on the data we decided to add English as a known language for every user. A separate set of 10 000 messages with 41 000 words (230 000 characters) was labeled by hand and put aside, one half for the development set and the other half for the test set. The development set was used for tuning the parameters of the learning process. The parameters that gave the best classification accuracy were used for the final tests that were run on the test set. The distribution of different languages in the hand labeled test set is shown in Figure 1. Since the 5 000 hand labeled

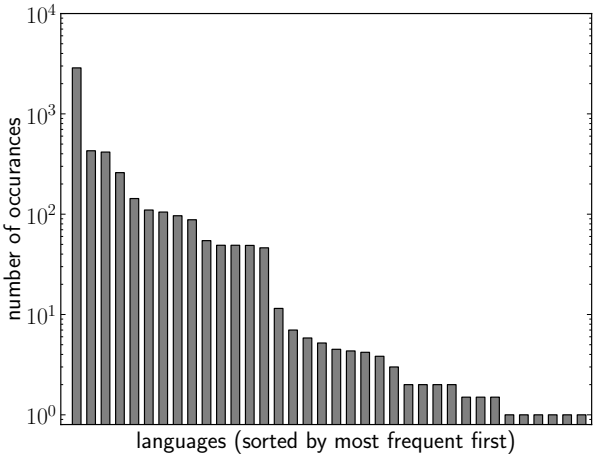


Figure 1: *Distribution of different languages in the hand labeled test set.*

samples of the test set were randomly picked from the data, we believe that this also represents the trend in the full data set.

Languages that use different character sets (e.g., cyrillic, greek, kanji, hiragana) were often written in romanized form. The language may change from one message to another or even within one message. The character encoding for all of the data was UTF-8. The discussions usually involved only a few languages. For this work, each message was considered as a separate entry and no effort was made to model the flow of the discussion. Also, the classifier tries to match just one language to each message. For some types of messages it was impossible to determine the language based on the message alone (e.g., messages containing only smileys, URLs, e-mail addresses, proper names, 'umm' or 'hahahaa'). Other messages were ambiguous; some languages could be ruled out but several would remain valid (e.g., 'si', 'sto', 'pronto', 'tak'). Some messages contained abbreviations not commonly used in print (e.g., 'lol', 'rotflmao'). Since the users may not be fluent in the language in which they are writing, the text could contain a high degree of grammatical and spelling errors.

3.2. Training

For training, we limited the number of languages that were checked for each message. We calculate the entropies and confidences over the languages that at least one of the participants knew or were learning (union of the sets of languages known to the participants). The confidence value was calculated based on a simple likelihood ratio. If the classifier output would not be a language known to all participants (intersection of the sets of languages known to the participants), the message would be discarded from the training set of the next round. The message would also be discarded if the confidence of the classifier was not high enough.

The initial dictionary-based classifier was built on top of Pyenchant (`http://www.rfk.id.au/software/pyenchant`) which used GNU aspell (`http://aspell.net`) to provide the back-end dictionaries. We used dictionaries for 107 languages. There were a few common languages that were not in this set, like Chinese, Korean and Japanese. If a language was detected to be character-based, limiting the search to the languages that the participants of the discussion

knew helped to find the correct one. A set of regular expressions was used to find unclassifiable messages (e.g., URLs, number sequences, smileys) and the results were used to train a "junk" model. The classifier was run on the training set to produce the initial labeling for the training of the n-grams.

All character-based n-gram models were trained with the VariKN toolkit (`https://sites.google.com/site/vesassiivola/varikn`). The full models were trained with interpolated modified Kneser-Ney smoothing. A combination of Kneser-Ney growing and revised Kneser pruning was used to create the variable-order models. We assumed there would be no significant information for language identification above order 15 models so this was set as the maximum order to save a bit of computing time. The n-gram models were used to produce a new labeled version of the training data that was used to train the next iteration of n-gram models. This was repeated until the performance of the model on the development set no longer improved. If there was a language that had less than 1000 bytes of training data available during any iteration, that language was removed altogether from the rest of the process. After iterations, 57 models were completed, one of which was a model for messages that were equally fit for all languages (e.g, smileys, number sequences, URLs). The training parameters for pruning, growing and confidence threshold were tuned by hand on the development data and the most accurate classifiers were run on the test data.

3.3. Testing

The classifier was free to choose any of the 57 modeled languages for all of the messages of the test set. The test set contained sentences in 40 different languages (for the distribution of the hand labeled set, see Figure 1). We chose not to create a test set that would contain the same amount of sentences of all languages for two reasons. First, finding a reasonably large fixed number of sentences for all languages by hand would take a prohibitive amount of work. Second, the current set should be distributed similarly to real world data.

In the test, 9 classifiers were tried. The *Dummy* classifier labeled all messages with the most common language of the data, English. The *Dictionary*-based classifier that was used to initially label the data was also tested. For this classifier, all ties involving English were resolved in favor of English and other ties were resolved arbitrarily. Now that the dictionary-based classifier was free to choose any of the available languages, the classifier had no means of distinguishing languages that it had no dictionary for. The tested n-gram classifiers were *Full 3-g*ram, *Full 5-g*ram, *Pruned 3-g*ram, *Pruned 5-g*ram and *Variable*-order classifiers. We also trained a full 3-gram and 5-gram classifier from the development data for 25 different languages to see what kind of performance a small hand-labeled training set would give.

In the test data, 4 different kinds of messages were found. For *Unambiguous* messages, the message was clearly in one single language (86.4% of test data). *Junk* data (7.9% of test data) would fit any language equally well or badly (e.g., numbers, URLs, smileys etc). *Ambiguous* messages could be valid in many languages (4.4% of test data). *Multilingual* messages contained words in two or more different languages (1.3% of training data). The results for unambiguous data are clear; the classification result is either correct or wrong. For ambiguous and multilingual data, the classification was counted as correct if it matched any of the possible languages. The results are given in Table 1.

Table 1: Classification results. For "Unambig.", all multilingual, ambiguous or junk messages were removed from the test. Entries with "devel" were trained from the 5 000 sentence development set whereas the ones with "bootstrap" were trained from the full training set using dictionary-based initialization.

Classifier	num n-grams	Correct % All	Correct % Unambig.
Dummy	NA	63.2	66.8
Dictionary	NA	78.5	79.0
Full 3-g (bootstrap)	6.8M	87.0	87.8
Full 3-g (devel)	31k	87.9	89.2
Pruned 3-g (bootstrap)	740k	88.6	89.5
Full 5-g (devel)	110k	90.1	91.0
Full 5-g (bootstrap)	57.7M	92.8	94.3
Pruned 5-g (bootstrap)	1.6M	93.6	95.1
Variable-g (bootstrap)	1.7M	93.9	95.4

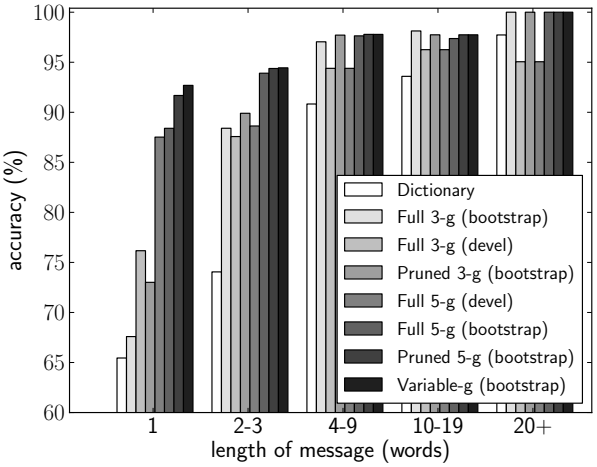


Figure 2: *The accuracy of the classifiers with respect to message length. Only unambiguous test set sentences were taken into account.*

Figure 2 shows how the length of the message affects the classification accuracy. Figure 3 shows the confusion matrix for two of the classifiers. For variable order models, Figure 4 shows how n-grams are distributed between different orders and which n-gram orders are used during the classification.

4. Discussion

The most popular language of the messages was English, as shown by the performance of the dummy classifier. The n-gram based approaches give clearly better results than our dictionary-based approach. Training n-grams from the small hand-labeled set seems to give a good performance. There were a few surprises in the results: pruned models had better performance than the full models. Our hypothesis is that the pruned models are more robust to variation in the data. Further, the trigram model trained from hand-labeled data outperforms the full trigram model trained with the presented bootstrap procedure. It appears that having more accurate data was more important than having a lot of less accurate data. The situation is reversed for the 5-gram model, which can better exploit the large amount of data. It would be interesting to see if adding priors for each

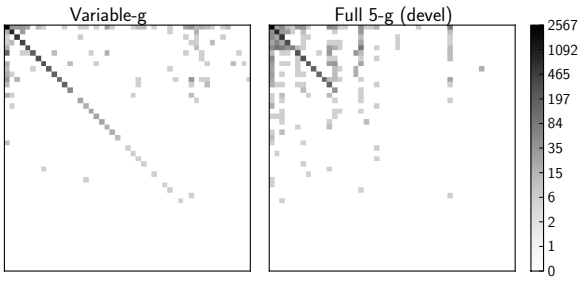


Figure 3: *Confusion matrices. On y-axis, the real language from the most common (top) to the least common (bottom) is shown. On x-axis, the language that was recognized is show. The 5-gram model trained from the development set is failing for the less common languages because the training set was small.*

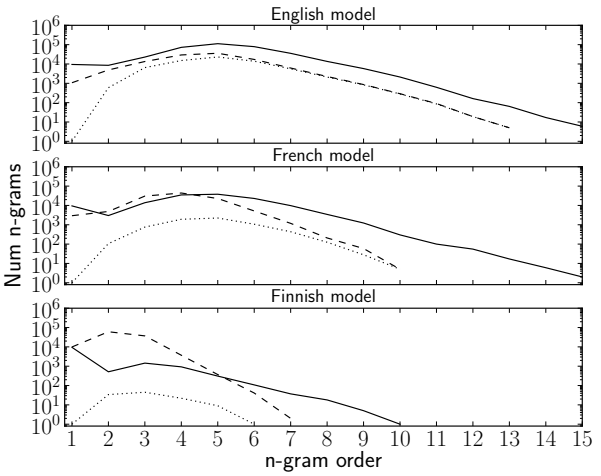


Figure 4: *The number of n-grams by order and the n-gram hit rate on the test set for selected models for the variable order classifier. The solid line shows the how the n-grams are distributed between different orders in the model. The dashed line shows which n-gram orders were used when classifying the 5 000 messages of the test data. The dotted line shows which n-gram orders were used when classifying the data that was in the same language as the model.*

language would improve the performance of the models. The variable-order model trained with the suggested bootstrap procedure seems to give the best results.

The cost function for growing and pruning the variable-order n-gram models reflects only the gain in modeling accuracy for that language. The cost function could be made to reflect the gain in discrimination between models. By choosing an interesting subgroup of languages and using discriminative cost function it may be possible to construct models that offer some insight into what the most important differences are in the character level for the chosen languages. The models produced in this manner should be small, but the model estimation would be computationally more demanding.

It would be possible to augment the training data with texts for which we know the language. In preliminary experiments, adding text for languages that were common in the original corpus didn't seem to increase the performance. Finally, it would be interesting to experiment with the variable-order n-gram method as a model of phonotactics for language identification in the audio domain.

5. Conclusions

We have shown that it is possible to build a high accuracy language identification system for text chat messages from unlabeled data. Initial labeling was created based on the knowledge of the languages that the participants of the chat knew and dictionaries were used to choose between the possible languages. The final classifier was based on character n-grams. We found that controlling the number of parameters of the n-gram model through combination growing and pruning methods gives us a compact model with excellent accuracy. It seems that including more information about possible romanizations of languages written in non-Latin scripts would further improve the accuracy of the classifier.

6. References

[1] B. Hughes, T. Baldwin, S. Bird, J. Nicholson, and A. McKinlay, "Reconsidering language identification for written language resources," in *Proc. of LREC'06*, 2006.

[2] R. Lins and P. Conçalves, Jr., "Automatic language identification of written texts," in *Proc. of SAC'04*, 2004.

[3] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Christianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, 2002.

[4] J. Navrátil, "Recent Advances in Phonotactic Langauage Recognition Using Binary-Decision Trees", in *Proc. of Interspeech'06*, 2006.

[5] K. Beesley, "Language identifier: A computer program for automatic natural-language identification of on-line text," in *Proc. of 29th Annual Conference of ATA*, 1988.

[6] W. Cavnar and J. Trenkle, "N-gram-based text categorization," in *Proc. of SDAIR'94*, 1994.

[7] R. Cordoba, L. D'Haro, F. Fernandez-Martinez, J. Macias-Guarasa, and J. Ferreiros, "Language Identification based on n-gram Frequency Ranking", in *Proc. of Interspeech'07*, 2007

[8] G. Grefenstette, "Comparing two language identification schemes," in *Proceedings of the 3rd International conference on Statistical Analysis of Textual Data*, 1995.

[9] P. McNamee and J. Mayfield, "Character n-gram tokenization for European language text retrieval," *Information Retrieval*, vol. 7, 2004.

[10] W. Teahan, "Text classification and segmentation using minimum cross-entropy," in *Proc. of RIAO2000 CBMIA*, 2000.

[11] T. Vatanen, J. Väyrynen, and S. Virpioja, "Language identification of short text segments with n-gram models," in *Proc. of LREC'10*, 2010.

[12] S. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech and Language*, vol. 13, no. 4, 1999.

[13] S. Chen and R. Rosenfeld, "A survey of smoothing techniques for maximum entropy models," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, 2000.

[14] S. Huang and S. Renals, "A parallel training algorithm for hierarchical Pitman-Yor process language models," in *Proc. of Interspeech'09*, 2009.

[15] V. Siivola, T. Hirsimäki, and S. Virpioja, "On growing and pruning Kneser-Ney smoothed n-gram models," *IEEE Trans. Speech Audio Language Process.*, vol. 15, no. 5, 2007.

[16] A. Stolcke, "Entropy-based pruning for backoff language models," in *Proc. of DARPA Broadcast News Transcription and Understanding Workshop*, 1998.