Language Identification of Tweets Using LZW Compression

Brian O. Bush
Center for Spoken Language Understanding
Oregon Health & Science University
Portland, OR

Abstract

Language identification plays a major role in natural language processing applications and is commonly used as a pre-processing phase. Typical techniques employ n-gram models and require text parsing and cleanup such as punctuation removal, white-space normalization, etc. This paper investigates the use of compression to construct a system that requires little preprocessing. Initial results are provided for small text messages (tweets) from Twitter. No comparisons are made with other techniques and therefore this work should be seen as an exploratory feasibility study in using compression for language identification of short texts.

Keywords: Language Identification, LZW Compression, Twitter

1 Introduction

The primary task of language identification is deciding a language among k candidate natural languages a given text is written. Many different methods have been used for language identification, including n-gram (Cavnar and Trenkle, 1994), high-frequency words (Grefenstette, 1995), and other statistical approaches (Dunning, 1994).

In this paper, we present an implementation of LZW-based compression to the problem of language identification. Specifically, we present results using this approach on short text documents. Benedetto et al. (Benedetto et al., 2002) presented the problem of language identification as simply an extension of Shanon's idea of entropy in that the compressibility of any given document depends on the source language.

2 Description of LZW

Ziv and Lempel are the creators of a family of loss-less compression algorithms (Ziv and Lempel, 2002). The general scheme that is consistent among their algorithms is by replacing a repeated sequence of characters by a reference to a previous occurrence. LZW (Welch, 1984) was created by Lempel, Ziv and Welch as an improvement to an earlier algorithm, LZ78 also by Lempel and Ziv. The popularity attached with LZW can be primarily attributed to its simplicity and is adaptiveness to underlying content being compressed. As the target text is being compressed a dictionary mechanism tracks strings that have been seen so far. The dictionary creation algorithm is as follows:

1. Initialize the table to contain all single-byte strings.
2. Read the first byte, c, from the input buffer. Set the prefix, w, to that byte.
3. Read the next input byte into c. If at end of buffer, exit.
4. If wc is in the dictionary then set w to wc and continue reading (step 3).
5. Store wc in the dictionary, set w to c and continue reading (step 3).

3 Implementation

There are two primary stages in language identification. The first stage is *modeling*, which consumes a set of documents in a specific language to derive a language model – essentially, a dictionary. The modeling stage is where specific features of the language are derived. The implementation only needs to perform LZW compression, since we are primarily concerned with the resulting dictionary.

The second stage is the run-time, or *classification*, which loads all available language models and evaluates an unlabeled document. In evaluation, each model compresses the target document and the model that results in the highest compression ratio is chosen as the the document language.

The language models were built from the translated versions of *European Union Human Rights Declaration*, which are freely available in over fifty languages. Using language analysis of a crawl The 43 specific languages[1] were loosely chosen based on popularity of Internet content. No pre-processing was performed on the training data and the dictionary for each model was limited to a maximum of 8k entries.

The test set was compiled by native speakers of Italian, English, Japanese, Portuguese and Spanish. First, search terms were compiled for each language. Second, using the Twitter search REST API, tweets were downloaded using those search terms. Finally, each language set of tweets were manually reviewed. The resulting test set was composed of 20 tweets per language. To be considered in a specific language, a tweet had to be composed of at least 80% words in that language. This final set of reviewed tweets were used to gauge the accuracy of our language identifier.

4 Results

The following results are broken down by language and the accuracy for that specific language.

Language	Tweets
Italian	85%
English	80%
Japanese	100%
Portuguese	60%
Spanish	75%

In analyzing the incorrect tweets, we found that the message had many foreign loanwords, slang, abbreviations (e. g. 'RT'), hashtags (e. g. #nwnlp2014) and URLs. Removing non-linguistic artifacts would most likely increase accuracy especially in small text applications. It is important to note that languages, such as Japanese, are limited to 140 characters, but the actual byte count is up to 420 bytes. Larger texts, such as web pages, have yielded more accurate results in informal testing.

[1]Languages in our system: Afrikaans, Albanian, Arabic, Armenian, Basque, Bulgarian, Catalan, Chinese (Simplified/Traditional), Czech, Danish, Dutch, English, Estonian, Finnish, French, Georgian, German, Greek, Hebrew, Hungarian, Indonesian, Italian, Japanese, Korean, Latvian, Lithuanian, Macedonian, Malaysian, Norwegian (Bokmal/Nynorsk), Polish, Portuguese, Romanian, Russian, Slovakian, Spanish, Swedish, Tagalog, Thai, Turkish, Ukrainian, Vietnamese

5 Conclusions

In this paper the task of using LZW compression to perform language identification has been presented. The benefits of using compression techniques are primarily in the ease of implementation and the lack of pre-processing during run-time language identification. Possible enhancements to presented approach are: (1) Expand test set size and compare with other techniques, and (2) Fold case in both training and run-time data, which would most likely increase accuracy in cases where uppercase characters are used often.

References

Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. 2002. Language trees and zipping. *Phys. Rev. Lett.*, 88(4):048702, Jan.

W.B. Cavnar and J.M. Trenkle. 1994. N-gram-based text categorization. *Ann Arbor MI*, 48113:4001.

Ted Dunning. 1994. *Statistical identification of language.* Citeseer.

G. Grefenstette. 1995. Comparing two language identification schemes. In *Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data (JADT'95)*, volume 1, pages 263–268.

Terry A. Welch. 1984. A technique for high-performance data compression. *IEEE Computer*, 17(6):8–19.

J. Ziv and A. Lempel. 2002. A universal algorithm for sequential data compression. *Information Theory, IEEE Transactions on*, 23(3):337–343.