Layered approach for language identification of multilingual documents

Team 11, AlphaDogs

Name	USC ID	E-mail
Minchul Park	8210695817	minchulp@usc.edu
Meng-Yu Chung	9208398418	mengyuch@usc.edu
Nithin Chandrashekhar	1922846582	nithinch@usc.edu
Samual Krish Ravichandran	6334599483	samualkr@usc.edu

1. Introduction

Language identification (LID) plays a key part in many NLP tasks that require linguistic assumptions, including machine translation. This task can be difficult; even commercial-level translators like Google translate often confuse among those languages closely related by script (Wallon and French, Punjabi and Hindi/Urdu) or lexicon (Hindi language family).

Our approach is to find those similar language groups in a systematic way by LID and gradually specialize identifiers to each language group by considering specific linguistic knowledge. This allows us to break down a hard problem of LID with more than 100 languages into small modularized problems and incrementally improve the overall performance with ease.

Meanwhile, multilingual documents make the problem complicated. Previous works on multilingual identification are done mostly on a document level [1] [2] [3] [4] and there have been only a few attempts at identifying multiple languages in a fine-grained way [5] [6]. As our approach requires exact spans of each language, we will treat it as a sequence-labeling problem.

2. Method

- Materials

Our task will be based on the ALTA-2010 Shared Task [7] dataset. This dataset provides multilingual texts for 74 languages by concatenating comparable monolingual documents. In the case of need for more data/annotations, we will use the same method.

- Procedure

We will build a basic identifier from several standard methods for a sequence-labeling problem, such as linear-chain CRFs [8], structured SVM [9]. In both cases, we will use PyStruct [10]. Based on this (potentially character-based) identifier, we will find similar languages in a systematic fashion:

1. Identify languages from language-pair datasets.
2. Build a weighted graph using error rates between two languages as weight values.
3. Partition the graph to determine similar language sets.

Once similar language sets are achieved, we can repeatedly try LID with advanced features and categorize similar languages until narrowing down to one answer. We cannot assume that all languages in a document are in the same group; it is one reason of this being a sequence-labeling problem.

We expect that many generic features (like characters, words, N-grams, etc.) can be shared among identifiers while language specific features can also be introduced. Furthermore, an ideal configuration for a given set of features could be automatically derived by enumerating possible state space.

- Evaluation

For evaluation, we will take averages of word-level F1 scores for each language and a baseline method will be simple dictionary lookup based on word counting. The scope of evaluation can vary; it might be worth to compile and compare identification performance for each language group. It would be also interesting to check correlation between widely accepted language family and our categorization result.

3. References cited

[1] Linguini: Language Identification for Multilingual Documents, JMIS, 1999

[2] Reconsidering Language Identification for Written Language Resources, LREC, 2006

[3] Automatic Language Identification: An Alternative Unsupervised Approach Using a New Hybrid Algorithm, IJCSA, 2010

[4] Automatic Detection and Language Identification of Multilingual Documents, Tran. ACL, 2014

[5] A Fine-Grained Model for Language Identification, Proc. SIGIR, 2007

[6] Labeling the Languages of Words in Mixed-Language Documents using Weakly Supervised Methods, NAACL HLT, 2015

[7] Multilingual Language Identification: ALTW 2010 Shared Task Dataset, ALTW, 2010

[8] Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, 2001

[9] Large Margin Methods for Structured and Interdependent Output Variables, JMLR, 2005

[10] PyStruct - Structured Learning in Python

4. Division of labor between the teammates

	Theory	Coding	Data	Writing
Minchul	O	O	O	
Meng-Yu		O	O	
Nithin		O	O	O
Samual	O	O		O

This document contains 510 words except the reference part.