12-2017

Authorship identification of translation algorithms.

Keishin Nishiyama
University of Louisville

AUTHORSHIP IDENTIFICATION OF TRANSLATION ALGORITHMS

By

Keishin Nishiyama
M.A., Language and Culture, Shimane University, Japan, 2011

A Thesis
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2017

AUTHORSHIP IDENTIFICATION OF TRANSLATION ALGORITHMS

By

Keishin Nishiyama
M.A., Language and Culture, Shimane University, Japan, 2011

A Thesis Approved On

12/1/2017
Date

By the following Thesis Committee:

Roman Yampolskiy, Ph.D., Thesis Director

Ibrahim Imam, Ph.D.

Ayman El-Baz, Ph.D.

ACKNOWLEDGEMENTS

I thank Dr. Yampolskiy for his countless support for my master thesis and my family for encouraging and helping me during the whole master program. Likewise, I thank Dr. Chang and Dr. Imam and Dr. El-Baz for accepting to serve on my thesis committee.

Finally, I would like to say thank you to my colleagues in the Computer Engineering and Computer Science Department for their support and friendship.

ABSTRACT

AUTHORSHIP IDENTIFICATION OF TRANSLATION ALGORITHMS

Keishin Nishiyama

December 1, 2017

Authorship analysis is a process of identifying a true writer of a given document, and has been studied for decades. However, only a handful of studies of authorship analysis of translators are available despite the fact that online translations are widely available and also popularly employed in automatic translations of posts in social networking services. The identification of translation algorithms has potential to contribute to the investigation of cybercrimes, involving translation of scam messages by algorithmic translations to reach speakers of foreign languages.

This study tested bag of words (BOW) approach in authorship attribution and the existing approaches to translator attribution. We also proposed a simple but accurate feature that extracts the combinations of lexical and syntactic information from texts. Our experiments show that the proposed feature is text size invariant.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Authorship analysis of translation algorithms, or translator analysis, deals with identifying which translation engine has been used in a given translation task. In machine translation, given an input text in a known language, the goal is to generate the text with the same meaning in a different language. For ages, this task was performed by professional human translators, who spent countless hours learning their jobs, and typically, perfecting translation only between a handful of languages. Recently, machine generated translation has been widely adopted due to the many advancements in the field of machine learning and natural language processing. This trend is driven by the popularity of many translation engines such as Google Translate, Bing Translator, FreeTranslation.com, and many more. Even some social networking services such as Facebook and Twitter provide machine translation for users viewing posts in their preferred languages.

For long text corpuses machine translation tends to underperform. However, for short text, most machine translation algorithms perform almost to an accuracy similar to that of human expert [1]. In criminal investigations, it may be important to identify which translation agent or algorithm has been used in producing a given translation. The identification could be very challenging to do manually; thus, there is a need to develop an automated way to identify the agent behind a given translation.

In recent literature, some researchers have presented a few solutions to this problem. Most previous work seems to follow a common paradigm: a feature extraction step followed by a classification step. This common classification architecture has proven to work effectively for the problem at hand. However, the reported per-

formance has been measured on small and relatively easy datasets. Moreover, the evaluation was done a decade ago and since then, translation agents have significantly improved, making their identification more challenging.

In this thesis, we deal with those shortcomings by proposing a new classification pipeline for authorship analysis of translation algorithms. Specifically, we propose new features and show their effectiveness, and also show that our proposed method performs well on new and challenging datasets. The rest of this thesis is structured as follows. In section 2, we discuss related work and section 3 presents our proposed method. The implementation details and results are presented in section 4 and section 5 concludes this thesis.

CHAPTER 2

LITERATURE REVIEW

This chapter will cover background for authorship analysis, translator analysis and classification algorithms that will be used in our experiments. Authorship analysis is covered first since translator analysis is a subset of authorship analysis.

2.1 Authorship Analysis

The recent massive increase of available electronic texts allows various kinds of electrical texts to be analyzed, e.g., books, email messages, online forum messages, blogs, source code, etc.. In many cases, the main focus of these studies is to identify the true author of a give text among a set of possible authors [2].

Authorship attribution studies have seen significant progress as a result of advancements in machine learning and natural language processing. Previous studies widely adopted the machine learning models such as Naive Bayes, Support Vector Machine (SVM), and k-nearest neighbors. [2–4]. As inputs to machine learning algorithms, a wide variety of features extracted from texts have been proposed for different styles of texts. However, the most popular way of extracting features from a text is bag-of-words (BOW) approach. BOW approach simply views a document as a bag full of words by ignoring its structures such as its paragraphs, sentence order, or word order. It counts the occurrence of each word and provides the frequencies of words in a text as an input features to machine learning algorithms. BOW approach can utilize not only words but any tokens such as Part-of-Speech (POS) tags, characters, etc. as long as it can count the frequencies. BOW can handle anything countable as tokens, but a specific token type is generally selected to extract both lexical and syntactical information or either of them from texts. Lexical features employ the

vocabulary used in texts, while syntactic features extract syntax information.

Lexical text analysis treats each word as a token and establishes frequency counts for all words via an efficient algorithm [3]. That is, word level BOW and character-level BOW approaches further decompose a word into a sequence of characters and count the frequency of occurrence of individual characters. In BOW approaches, moreover, counting a series of tokens as a single token is widely adopted and called n-grams. Word n-grams can capture phrases and character n-grams can capture sub-words or morphemes. A word n-gram can be also viewed as a capture of partial syntactic information since a series of words contains word orders.

In other ways, topic model approaches, tokens, mainly words, are represented by a mixture of topic distributions [5]. Likewise, word embedding approaches map a token to a fixed-length vector of real numbers [6, 7]. Those two approaches represent words as fixed-size vectors and provide dense inputs comparing to the features that contain lots of zero values in BOW. Mapping to dense vector also helps to handle large vocabulary that contains a large number of rare words, which could be a problem in BOW approaches.

A use of lexical features with BOW is a simple and efficient approach but ignores word-order information and discards many informative syntactic features. The underlying idea of using syntactic features is that authors have specific syntactic patterns that they tend to follow regardless of whether they do so consciously or unconsciously. For example, a novel writer is likely to use more vocabulary-rich and longer sentences than a child would. A simple use of syntactic features is to use Part-of-Speech (POS) tags estimated by a POS tagger with BOW, instead of using words or characters as tokens directly [8, 9]. POS taggers assign a tag of syntactic markers to each token, e.g., words, based on the given contextual information around the token. Since POS tagger estimate tags to words with high accuracy [2] and constructing syntax trees for each sentence in texts are expensive operations. Moreover, grammatical errors need to be handled precisely and this could be a problem for our translation data. Our data is generated by translation algorithms not by a human and have more errors. Instead, we will use a common approach, n-grams of POS

tags, to extract syntactic information and to represent it as features [10]. Syntactic features improve the classification results especially when the length of the given text is short. A large number of short texts make word BOW features sparse and reduce performance of the classification algorithms. However, the use of syntactic markers produces considerably smaller vocabulary size compared to that of lexical features, and improves short-text classification.

Table 2.1 summarizes the features by information group [2, 4]. Lexical feature extract information about vocabulary such as word n-grams, and character n-grams. Syntactic information is mainly about word order and specific syntactic rules. N-grams capture partial word orders and functional words and punctuation as well as syntactic information. Semantic features extracts meaning of sentences, or texts. A check of synonyms is one of the easiest ways to handle semantics. In many cases, meaning is represented by some special forms such as Frame semantics [11]. Finally, other information can be used as a feature as long as we can extract it such as text length.

TABLE 2.1

A classification of features

Groups	Feature
Lexical	Word n-gram Character n-gram Misspelled words Special characters Topic model Word embedding
Syntactic	Functional word Punctuation Syntax tree POS n-gram Word n-gram Syntactic error
Semantic	Synonyms Semantic dependency Frame semantic Sn-gram
Other	Application specific approaches

2.2 Translator Analysis

Few studies of authorship analysis have been done with the focus of translator recognition. Hedegaard and Simonsen (2011) investigated authorship attribution of human translated texts. Their approach demonstrates adding semantics features that are extracted by frame semantics to lexical and syntactic features would improve authorship attribution. However, their corpus is built from human translations of Russian novels to English in Project Gutenberg and limited in terms of available translations because of a lack of human translators. This study shows that authorship analysis approach can also identify translators, but the translators are human, not algorithms.

Suresh et al. (2011) investigated Translator Attribution of Google and Bing Translate on the French text. Their approach first generates topic distributions of stop words from Latent Dirichlet Allocation (LDA) [12, 13] and features are created

by those topic distributions of stop words. Each of documents in their experiment is a chapter in books and the classification is done with SVM. Their study indicates a small number of topics and stop words can generate informative features but this could be because each input document is large.

Caliskan and Greenstadt (2012) tested the translator attribution with the pairs of the translations between French and English, and those between Dutch and English. The size of each document was about 500 words. The used translators were Google Translate, Language Weaver and Systran. The combination of nine different features (translation set) produced 92.75% accuracy on French dataset, and 94.44% on Dutch dataset by Support Vector Machine (SVM) and Naïve Bayes (NB) classifiers. The translation set is a collection of 9 different features namely: average character size per word, the character counts, the frequencies of function words, the frequency of letters, punctuation, special characters, top letter bigrams, top letter trigrams, words, and word lengths. Caliskan and Greenstadt (2012) produced state-of-the-art accuracy of translation attribution. Their studies show the features in general authorship attribution also work for true translator detection. Table 2.2 compares three previous studies for translator classification in terms of translators, the size of a text, and features. As to length of a text, their studies uses fairly long texts such as chapters of books and use custom features. This leads us to investigate translator attribution with respect to text length with simple n-gram approach.

TABLE 2.2

A comparison of three previous studies for translator attribution.

Study	Translator	Length of a text	Feature
Hedegaard and Simonsen (2011)	Human translators	200 to 33000 words	Character n-gram Frame semantics
Suresh et, al. (2011)	Google Translate, Bing	A chapter in a book	Topic model by LDA
Caliskan and Greenstadt (2012)	Google Translate, Language Weaver, Systran	500 words	Translation set: 9 different features

2.3 Classification algorithms

This section reviews classification algorithms for text classification. Though many of classification algorithms have been applied on text classification, we discuss 2 algorithms that we apply to our datasets: Naive Bayes and Random forest. Although SVM is selected for classification algorithm in previous studies [5, 14], we omit it for our experiments due to its computational expensiveness.

2.3.1 Naive Bayes

Naive Bayes is a supervised-learning classification algorithm based on Bayes theorem and the assumption: all features are independent. The probability of a class variable $y \in Y$ given a document $d \in D$ is computed as

$$P(y|d) = \frac{P(y)P(d|y)}{P(d)}. \tag{2.1}$$

Since Naive Bayes receives a document d as a set of features: x_i to x_n, it will be

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)}. \tag{2.2}$$

By independence assumption of features, all x_i are independent to each other. The equation is equivalent to

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}. \tag{2.3}$$

Since $P(x_1, ...x_n)$ is constant and can be omitted, the above equation is proportional to

$$P(y|x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y). \tag{2.4}$$

For classification, the predicted class $\hat{y}$ for a given document d is determined by probabilities of $P(y|d)$. Therefore, $\hat{y}$ is determined by

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y)\prod_{i=1}^{n} P(x_i|y). \tag{2.5}$$

$P(y)$ and $P(x_i|y)$ are estimated by Maximum A Posteriori (MAP) estimation. Assuming each document is generated by independent trials drawn from a multinomial distribution of words, $P(x_i|y)$ and $P(y)$ are estimated based on empirical counts of them in given data. $P(y)$ is estimated by the proportion of given training documents that belong to class y and $P(x|y)$ is approximated as

$$P(x_t|y) = \frac{1 + \sum_{i=1}^{|D|} N_{ti} P(y|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} P(y|d_i)}. \tag{2.6}$$

Where vocabulary $V = \{x_i, \ldots, x_{|V|}\}$, N_{ti} is the number of times that word x_t occurs in the document d_i. The additional one in the numerator is to handle zero-frequency problem and $|V|$ in the denominator is to keep the sum of probabilities to one [15, pp. 100-109] [16].

2.3.2 Random Forest and Decision Tree

A random forest is an ensemble classifier based on the combination of different decision trees. Therefore, we will cover decision tree first and random forest later.

2.3.2.1 Decision tree

A decision tree is a tree which has two types of nodes; decision nodes and leaf nodes. A decision node specifies a rule to split a given data from its parent into two groups, which are passed to dependent child nodes. A leaf node indicates a class. During inference, new samples are classified by iteratively applying decision rules.

In learning phase, a decision tree is constructed by partitioning the training data recursively in order to make the resulting subsets as pure as possible. Each partitioning rule will be a decision node in a tree. Each node chooses the best attribute and threshold to partition the data at the current node according to the attributes of the data given to the node. The best attribute and threshold is selected based on the function which minimize the impurity after the partitioning, or maximize the purity. Although there are several function that measures impurity such as Gini index [17, p. 134], the most popular function used for decision tree learning is information gain,

which is used in C4.5 [18]. Information gain is calculated by using the following entropy function [15, pp. 100-109].

$$entropy(D) = -\sum_{j=1}^{|C|} P(c_j) \log_2 P(c_j) \tag{2.7}$$

where $P(c_j)$ is the probability of class c_j in data set D, which is the number of examples of class c_j in D divided by the total number of examples in D. In the entropy computation, $0 \times \log 0$ is defined as 0. In order to detect the attribute which can reduce the impurity most if it is used to partition D, every attribute is evaluated. By Letting the number of possible values of the attribute A_i be v and using A_i to partition the data D, we will divide D into v disjoint subsets D_1, D_2, ..., D_v. The entropy after the partition is

$$entropy_{A_i}(D) = \sum_{i=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j). \tag{2.8}$$

Here we assume all A_i is binary value for simplicity, then the information gain of attribute A_i is computed by

$$Information\ gain(D, A_i) = entropy(D) - entropy_{A_i}(D). \tag{2.9}$$

Algorithm 2.1 finds the best attribute for partitioning to given data by checking impurity scores of all possible splits. When the given attributes are continuous features, we need to find the threshold for partition in addition to finding the best attribute. The threshold can be determined by sorting values of an attribute and checking all possible thresholds.

Algorithm 2.2 constructs a decision tree by finding best partitions with Best-Split in Algorithm 2.1. It calls itself recursively until it reaches a leaf node, which only has samples of a class. In many case, a decision tree is pruned by setting maximum depth, minimal number of samples in a node, or more.

2.3.2.2 Random Forest

A random forest classifier is constructed with multiple decision trees and estimates by combining the estimations of the trees. A tree in random forests is built

Algorithm 2.1 BestSplit(D, A)

Inputs: D: Training data, A: a set of attributes.
Output: A_{best}: the best attribute to split on

```
l_min ← 1;
for each A_i ∈ A do
  split D into subsets D_1, ..., D_l according to the values V_j of A;
  if Impurity(D_1, ...D_l) < l_min then
    l_min ← Impurity(D_1, ...D_l);
    A_best ← A_i;
  end if
end for
return A_best
```

Algorithm 2.2 DecisionTreeConstruction(D, A)

Inputs: D: Training data, A: a set of attributes.
Output: T: A tree constructed

```
if D contains only examples of a class c_j ∈ C then
  return a leaf node labeled with class c_j
end if
S ← BestSplit(D, A)
split D into D_i according to S;
for each i do
  if D_i ≠ ∅ then
    T_i ← DecisionTreeConstruction(D, A)
  else
    T_i is a leaf node labeled with a class c_j of D
  end if
end for
```

with two techniques; bootstrap aggregating and random subspace method. These two methods let different trees have different subsets of samples and see different subsets of attributes.

Bootstrap aggregating, or bagging for short, is a simple but effective ensemble method that creates diverse models on different random samples of the original training data. Samples are drawn uniformly from training data with replacement and those samples are called bootstrap samples. Because of sampling with replacement,

a set of bootstrap samples generally contains duplicated samples and also lacks some of the original data points even when the size of bootstrap samples is set to the same size as the original training sample size. Therefore, different trees are constructed with different bootstrap sample sets but drawn from the same data.

When creating a tree with boost strap samples, random forests apply another technique called subspace sampling to those samples. Subspace sampling randomly selects k features from original feature set. A tree can use only randomly-selected k features of bootstrap samples in its training. The essence of subspace sampling is to prevent trees from being strongly correlated with each other. If a few features are very strong predictors for label estimation, most of trees rely on the features and become correlated. Ensembling less-correlated models leads to better performance. The number of k is a hyper-parameter but typically $\sqrt{K}$ is set when the number of attributes is K.

Algorithm 2.3 summarize the processes of Random forest training. In inference time, the final prediction is determined by majority vote of all trees.

Algorithm 2.3 Random Forest Training

Inputs: D: Training data, T: Number of trees

for t in T **do**
  create a subset of samples s from D
  select a subset of features k
  train a decision tree with subset s by only using selected feature set k
end for

CHAPTER 3

EXPERIMENTAL DESIGN

This section will present our approach to translator recognition, data collection, and experimental details.

3.1 Text feature extractions

Though three of the previous studies presented highly accurate results, their translated texts are long and there is a space to investigate shorter translations. Moreover, their sophisticated approach could be too complicated and a simpler approach could produce similar results. Therefore, we initially selected word or POS n-grams as features to extract lexical and syntactic features. Word frequency approach is one of the most simple but popular features in text classification tasks. Prior to making n-grams, all words are stemmed and POS tags are extracted by The Natural Language Toolkit (NLTK) [19], which uses 41 POS tags as in Table 3.1. Even though NLTK tags words with 41 different tags, the main 7 categories are: nouns, adjectives, adverbs, interrogative words, functional words, and symbols. By adding extra information such as use of singular and plural form, the number of tags expands to 41.

TABLE 3.1

POS tags by NLTK

Tag	description	Tag	description
$	dollar	NNP	noun, proper, singular
”	closing quotation mark	NNPS	noun, proper, plural
(	opening parenthesis	NNS	noun, common, plural
)	closing parenthesis	PRP$	pronoun, possessive
,	comma	RB	adverb
-	dash	RBR	adverb, comparative
.	sentence terminator	RBS	adverb, superlative
:	colon or ellipsis	RP	particle
CC	conjunction, coordinating	SYM	symbol
CD	numeral, cardinal	TO	”to” as preposition or infinitive,marker
DT	determiner	UH	interjection
EX	existential there	VB	verb, base form
FW	foreign word	VBD	verb, base form
IN	preposition or conjunction, subordinating	VBG	verb, present participle or gerund
JJ	adjective or numeral, ordinal	VBN	verb, past participle
JJR	adjective, comparative	VBP	verb, present tense, not 3rd person singular
JJS	adjective, superlative	VBZ	verb, present tense, 3rd person singular
LS	list item marker	WDT	WH-determiner
MD	modal auxiliary	WP	WH-pronoun
NN	noun, common, singular or mass	WP$	WH-pronoun, possessive
		WRB	Wh-adverb

After making n-grams, both word and POS n-grams are weighted by term frequency-inverse document frequency (Tf-Idf.) The words that appears in too many or too few documents are removed and the ratios are determined by validation sets. Tf-Idf weighting to term t in document d is given by

$$Tf\text{-}Idf_{t,d} = Tf_{t,d} \cdot (\log(\frac{N}{df_t}) + 1) \tag{3.1}$$

where N denotes the total number of texts in a collection, $Tf_{t,d}$ denotes frequency of term t in document d, and df_t denotes the number of documents that contain term t (document frequency). Our initial experiments with word or POS n-

gram indicated that unigram of word and bigram of POS tags are useful. Hence, the combination of word and POS features could produce better results, utilizing both lexical and syntactic information. Simple concatenation of word and POS n-gram Tf-Idf values did not improve results. In our experiments, the sequences made by arranging words and POS tags alternately produced better result. We simply refer to it as word + POS feature from now on. Table 3.2 shows the samples of word, POS, and word + POS features. POS tags extracted from the word in word feature and word + POS feature combines word and POS features by placing words and POS tags in sequence.

TABLE 3.2

Word, POS, and word + POS samples.

Feature	Sample sequence
Word	Welcome to the Internet electronic library, Aozora Bunko
POS	VB TO DT NNP JJ NN , NNP NNP
Word + POS	Welcome VB to TO the DT Internet NNP electronic JJ library NN , , Aozora NNP Bunko NNP

Although a unigram in word + POS feature is just a unigram of word or POS, a bigram and a trigrams are different. Bigrams could have two sequence patterns: a set of n_{th} word and n_{th} POS, or a set of n_{th} POS and $n + 1_{th}$ word. A set of nth POS and $n + 1_{th}$ word can be a new feature that contains lexical and syntax information, specifying the POS tag appearing before the certain word. Trigrams are also produced from two kinds of sets: a set of n_{th} word, n_{th} POS and $n + 1_{th}$ word, or nth POS, $n + 1_{th}$ word and $n + 1_{th}$ POS. They can be considered as the more specified n-grams of word and POS: POS specified word n-grams and word specified POS n-grams, containing both lexical and syntactical information. Table 3.3 lists the possible placements of words and POS tags that can appear in unigrams, bigrams, and trigrams of word + POS features.

TABLE 3.3

N-grams in word + POS feature. A token is placed in a pair of round brackets.

N-gram	Tokens
Unigram	$(word)$, (POS)
Bigram	$(word_n, POS_n)$, $(POS_n, word_{n+1})$
Trigram	$(word_n, POS_n, word_{n+1})$, $(POS_n, word_{n+1}, POS_{n+1})$

3.2 Dataset creation

There is no publicly available dataset for translator analysis, and therefore we start with creating datasets for our experiments. The original texts of our dataset are the novels that were written in Japanese and collected from the websites called Aozora Bunko[1]. 127 novels of 5 authors are collected and translated into English using 4 online machine translation services, Google Translate[2], Bing Translate[3], FreeTranslation.com[4], and Systranet[5]. These four engines were used to translate each of these Japanese novels respectively. The total of 508 translated texts were used as textual data. Table 3.4 shows 2 selected sample translations: translations of a sentence by different translators which could be similar to each other as in translations of input 1, but could be different as in translations of input 2.

[1]http://www.aozora.gr.jp
[2]https://translate.google.com
[3]https://www.bing.com/translator
[4]http://www.freetranslation.com
[5]http://www.systranet.com/translate

TABLE 3.4

Input texts and translation samples. The translations of input 1 are similar but ones of input 2 are different. FT and SYS stand for FreeTranslation and Systranet respectively.

Input 1	インターネットの電子図書館、青空文庫へようこそ (Welcome to the Internet electronic library, Aozora Bunko)
Google	Welcome to the electronic library of the Internet, Aozora Bunko
Bing	Welcome to the Internet Electronic Library and the Blue Sky Bunko
FT	Welcome to the electronic library of the Internet, Aozora Bunko
SYS	Welcome to the electronic library and the blue sky library of Internet
Input 2	午後。風がすつかり呼吸を停めた。 (The wind died down in the afternoon.)
Google	Afternoon. Wind parked vinegar soaked breathing.
Bing	PM. The wind is already soaked breath parked.
FT	afternoon. air breathing is to put an end to that.
SYS	In the afternoon. The wind does you stopped temporary breath.

Since original Japanese texts are novels and greatly vary in their text lengths and writing styles, all of texts are grouped by translator and split into sentences, and normalized so that each sample has the same number of sentences. Though the precise word count per document minimizes the variance coming from the difference of text lengths [20], splitting documents by sentences is more natural and similar to actual usages. The number of sentences per document in a dataset is set to 150, 30, 20, 10, 5, 3, and 1. We refer to them as dataset plus document size, for example dataset 150. These datasets are considered as different datasets and used for comparing the effect of document size on classification accuracy since the length of text greatly affect classification accuracy [20,21]. Sentences that still contain Japanese characters are removed to avoid situations where the untranslated Japanese characters are good features and classification results immensely depend on them. Table 3.5 shows the statistics of each dataset: number of samples by translator and mean and standard deviation of number of words and that of characters in a dataset. Samples size for translators vary because of the removals of sentences that contain Japanese characters and different styles of translation, e.g. one-to-one or one-to-many sentence correspondence between Japanese and English.

TABLE 3.5

Statistics of datasets: the numbers of samples and word and character counts. FT and SYS stand for FreeTranslation and Systranet respectively.

Sentence	Samples					Mean (std)	
size	Google	Bing	FT	SYS	Total	Words	Characters
150	292	308	386	216	1202	3076 (±414)	13942 (±1553)
30	1458	1536	1929	1076	5999	616 (±107)	2793 (±443)
20	2187	2303	2894	1613	8997	411 (±80)	1862 (±343)
10	4373	4606	5787	3226	17992	206 (±52)	930 (±228)
5	8746	9211	11573	6451	35981	103 (±34)	465 (±156)
3	14576	15352	19287	10752	59967	62 (±26)	278 (±120)
1	43728	46054	57861	32255	179898	21 (±15)	92 (±68)

3.3 Experimental settings

Our experiments are done by changing the following settings in order to see the effects of document size, feature representations, classification algorithms, n-gram range, and stop word removal. All other parameters are explained in next section.

- Dataset: Dataset 150, 30, 20, 10, 5, 3, or 1.
- Features: word, POS, or word + POS.
- Classifiers: Random Forest or Naive Bayes.
- N-gram range: n-gram range is set between 1 through 3. We will check each n-gram itself and combinations of different n-grams.
- Stop words: either removing stop words or not.

3.4 Software implementation detail

We use scikit-learn as the main platform for our experiments and NLTK for word tokenization and POS tagging. 64% of a dataset is for training, 16% is for validation, 20% is for testing. All the tokens that appear in more than 97% of documents or less than 2 times are removed before calculating Tf-Idf scores of each token except

for the classifications of dataset 150 with Naive Bayes. In these classifications, the tokens appear in more than 50% of documents or less than 10 times are removed instead. Multinomial Naive Bayes and Random Forest are selected as classifiers for the experiments due to training speed and classification accuracies. We determined the parameters of the classifiers by using validation sets and the same parameters are used for all experiments for consistency. Random Forests classifiers operate 300 decision trees pruned to maximal depth 90.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we will discuss the experimental results and analysis. Each result reported have been averaged over 3 experimental runs..

4.1 Results of Classifications

Tables 4.1, 4.2, and 4.3 show all the classification accuracies for different experimental settings. Generally, classification accuracy goes down if the size of document decreases. Random forests produce better classification accuracies than Naieve Bayes for long-sized documents. However, Naive Bayes performs better for short-sized documents.

Table 4.1 shows bigram or trigram of words themselves do not perform well comparing to unigram. However, when it combined with unigrams, bigrams or trigrams of words improves classification.

As to POS ngrams, Table 4.2 shows that unigrams of POS do not perform well on the contrary. However, bigram and trigrams produce better classification. The number of POS tags, 41, could be too small by themselves. This could indicate that syntactic information are highly retained in word order.

Finally, it was observed that the performance of Naive Bayes built with word + POS feature deteriorates for long-sized document corpus and shown in the results for dataset 150 in Table 4.3. This could come from the same reason that Naive Bayes underperforms with long documents and word features.

TABLE 4.1

word feature

Model	N-gram	Stop words	Dataset						
			150	30	20	10	5	3	1
Random Forest	(1,1)	included	1.000	0.997	0.986	0.951	0.870	0.788	0.593
	(2,2)		0.996	0.993	0.974	0.937	0.831	0.739	0.539
	(3,3)		1.000	0.982	0.944	0.868	0.741	0.664	0.431
	(1,2)		1.000	0.997	0.979	0.956	0.886	0.802	0.609
	(2,3)		1.000	0.989	0.969	0.938	0.829	0.744	0.537
	(1,3)		1.000	0.992	0.978	0.955	0.886	0.803	0.610
	(1,1)	removed	0.983	0.998	0.984	0.928	0.826	0.722	0.498
	(2,2)		1.000	0.961	0.912	0.850	0.623	0.494	0.352
	(3,3)		0.971	0.808	0.662	0.479	0.379	0.340	0.328
	(1,2)		1.000	0.997	0.986	0.940	0.836	0.730	0.499
	(2,3)		1.000	0.961	0.913	0.849	0.623	0.494	0.352
	(1,3)		1.000	0.996	0.982	0.942	0.834	0.730	0.501
Naïve Bayes	(1,1)	included	0.931	1.000	0.998	0.981	0.932	0.844	0.599
	(2,2)		0.922	0.937	0.958	0.966	0.919	0.819	0.553
	(3,3)		0.950	0.952	0.954	0.911	0.775	0.622	0.399
	(1,2)		0.950	0.987	0.988	0.991	0.966	0.903	0.662
	(2,3)		0.954	0.950	0.963	0.972	0.924	0.831	0.555
	(1,3)		0.977	0.993	0.991	0.995	0.970	0.909	0.664
	(1,1)	removed	0.913	1.000	0.998	0.976	0.912	0.804	0.542
	(2,2)		0.972	0.990	0.978	0.876	0.625	0.484	0.351
	(3,3)		0.686	0.788	0.667	0.481	0.379	0.340	0.328
	(1,2)		0.931	1.000	0.997	0.980	0.923	0.811	0.545
	(2,3)		0.968	0.992	0.978	0.877	0.626	0.485	0.351
	(1,3)		0.940	1.000	0.997	0.981	0.924	0.811	0.545

TABLE 4.2

POS feature

Model	N-gram	stopwords	Dataset						
			150	30	20	10	5	3	1
Random Forest	(1,1)	included	0.384	0.716	0.643	0.592	0.694	0.614	0.515
	(2,2)		1.000	0.983	0.961	0.901	0.809	0.706	0.547
	(3,3)		0.996	0.986	0.961	0.891	0.779	0.671	0.521
	(1,2)		1.000	0.984	0.958	0.898	0.805	0.704	0.551
	(2,3)		1.000	0.987	0.961	0.906	0.804	0.697	0.545
	(1,3)		1.000	0.988	0.964	0.913	0.814	0.705	0.556
	(1,1)	removed	0.405	0.727	0.651	0.587	0.710	0.626	0.522
	(2,2)		1.000	0.988	0.957	0.918	0.813	0.720	0.558
	(3,3)		0.996	0.986	0.951	0.908	0.785	0.687	0.536
	(1,2)		1.000	0.99	0.958	0.908	0.812	0.723	0.565
	(2,3)		0.996	0.99	0.968	0.923	0.812	0.714	0.56
	(1,3)		1.000	0.989	0.959	0.923	0.82	0.725	0.569
Naïve Bayes	(1,1)	included	0.281	0.655	0.593	0.517	0.604	0.53	0.433
	(2,2)		1.000	0.99	0.982	0.923	0.831	0.749	0.557
	(3,3)		0.826	0.988	0.989	0.943	0.867	0.781	0.574
	(1,2)		1.000	0.985	0.982	0.919	0.821	0.737	0.549
	(2,3)		0.835	0.992	0.99	0.947	0.868	0.782	0.583
	(1,3)		0.835	0.99	0.991	0.944	0.864	0.779	0.579
	(1,1)	removed	0.264	0.662	0.598	0.518	0.588	0.509	0.428
	(2,2)		1.000	0.987	0.974	0.909	0.81	0.727	0.537
	(3,3)		0.926	0.985	0.979	0.937	0.847	0.756	0.555
	(1,2)		1.000	0.987	0.973	0.902	0.8	0.717	0.532
	(2,3)		0.95	0.987	0.979	0.934	0.846	0.757	0.563
	(1,3)		0.942	0.99	0.98	0.934	0.842	0.754	0.562

TABLE 4.3

word + POS feature

Model	N-gram	Stop words	Dataset						
			150	30	20	10	5	3	1
Random Forest	(1,1)	included	1.000	0.998	0.986	0.948	0.885	0.811	0.632
	(2,2)		1.000	1.000	0.992	0.969	0.893	0.805	0.632
	(3,3)		1.000	0.997	0.983	0.949	0.860	0.769	0.588
	(1,2)		1.000	0.999	0.992	0.970	0.902	0.817	0.647
	(2,3)		1.000	0.999	0.989	0.965	0.883	0.796	0.627
	(1,3)		1.000	0.999	0.992	0.968	0.897	0.816	0.649
	(1,1)	removed	0.979	0.998	0.986	0.936	0.875	0.774	0.600
	(2,2)		0.996	0.999	0.991	0.961	0.874	0.781	0.588
	(3,3)		1.000	0.992	0.975	0.917	0.819	0.728	0.517
	(1,2)		1.000	0.998	0.994	0.969	0.895	0.798	0.619
	(2,3)		1.000	0.999	0.988	0.955	0.866	0.778	0.588
	(1,3)		1.000	0.998	0.988	0.964	0.894	0.805	0.619
Naïve Bayes	(1,1)	included	0.954	1.000	0.998	0.982	0.939	0.859	0.625
	(2,2)		0.986	0.980	0.989	0.988	0.970	0.919	0.681
	(3,3)		0.972	0.943	0.973	0.982	0.960	0.898	0.632
	(1,2)		0.963	0.978	0.987	0.987	0.969	0.921	0.694
	(2,3)		0.972	0.927	0.966	0.982	0.972	0.925	0.687
	(1,3)		0.959	0.935	0.968	0.981	0.975	0.930	0.704
	(1,1)	removed	0.945	1.000	0.998	0.977	0.929	0.827	0.582
	(2,2)		0.972	0.995	0.990	0.986	0.956	0.885	0.626
	(3,3)		0.959	0.987	0.982	0.974	0.934	0.830	0.530
	(1,2)		0.940	0.993	0.989	0.984	0.960	0.889	0.645
	(2,3)		0.963	0.982	0.980	0.983	0.960	0.891	0.633
	(1,3)		0.950	0.982	0.981	0.983	0.964	0.900	0.656

4.1.1 Feature

We summarize the results with respect to features and dataset in Figure 4.1. Except for dataset 150, classification accuracies deteriorate with the decrease in document size. Outliers in word feature are as a result of poor performance of bigrams and trigrams, and ones in POS result from poor performance of unigrams. Word + POS features have less outliers and have less variance for n-grams compared to word features, with long boxes. Word + POS, which combines word and POS features, produces the most stable results.

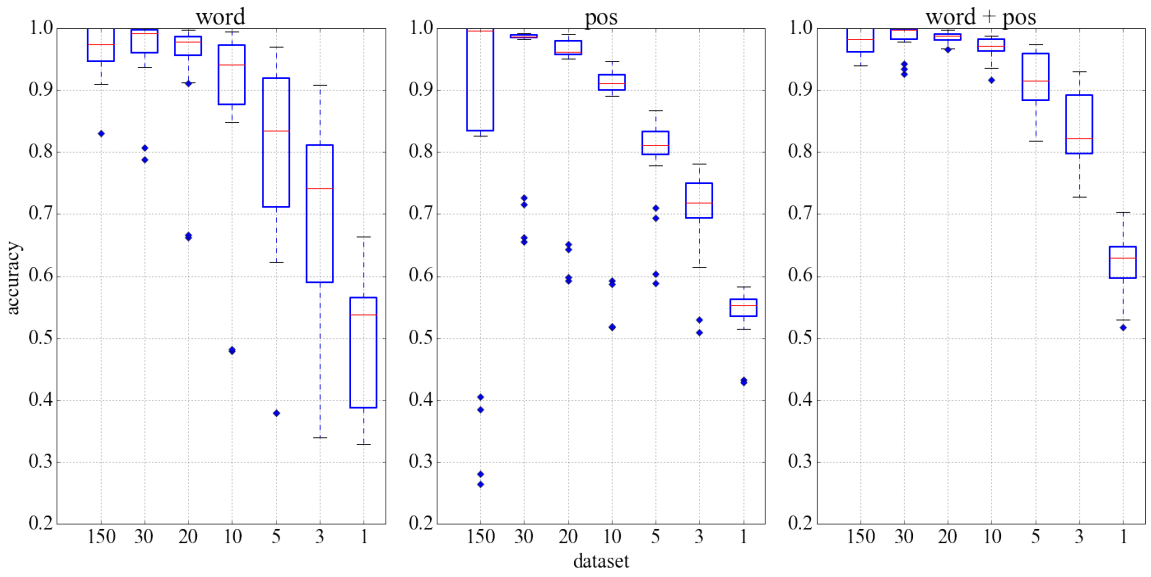

Figure 4.1: Accuracies grouped by feature. The top edge line of a box denotes 25 percentile, a read line denotes a median, and the bottom edge line denotes 75 percentile. From left to right: word, POS, and word + POS features.

4.1.2 Stop words

Figure 4.2 shows classification accuracies for dataset 1 with n-gram of range 1 to 3 grouped by classifier and feature. Blue lines indicate the classification results of classifications without stop word removal and green lines is for results with stop words removal. Regarding the dataset size, we observe the effect of stop words more when document size is small. When the length of documents is small, removing stop words has negative effect of removing high frequency tokens from a sample. Stop word removal effects on classifications using word features but does not do on ones using POS features. Since word + POS feature combines word and POS features, the effect of stop word removal to classification accuracy is between those of words and POS features. The effect of stop words can also be seen in the classification of large-sized documents, dataset 150, with Naive Bayes classifiers and stop-word removal improve classification results.

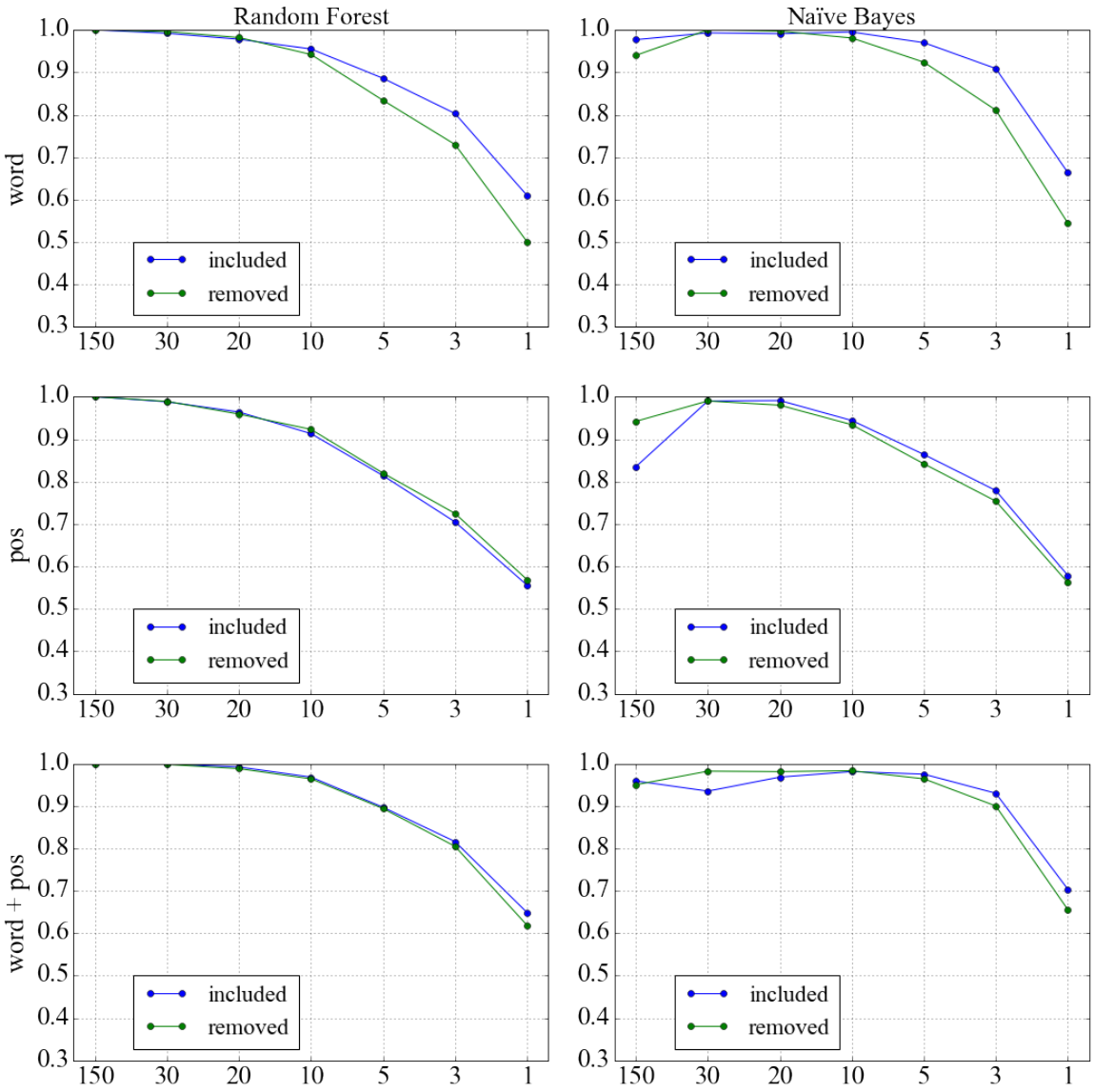

Figure 4.2: Accuracies of dataset 1 classification with n-gram range (1,3)

4.2 Error Analysis

Next, we examine mis-classification errors. Figure 4.3 is a collection of confusion matrices for experiments on dataset 1. N-gram range is set from 1 to 3. Each translation algorithms are denoted by numbers:

- 1: Google translate
- 2: Bing Translate

- 3: FreeTranslation
- 4: Systranet

Among the all results, the most distinctive samples are those of label 3 (FreeTranslation). This could caused by the fact that we have more samples of FreeTranslation than others. This can also be seen in the mis-classified samples, where the samples of all other 3 labels are mis-classified more to label 3 than others. The second most accurate algorithm is labeled 4, Systranet, going by their respective accuracies.

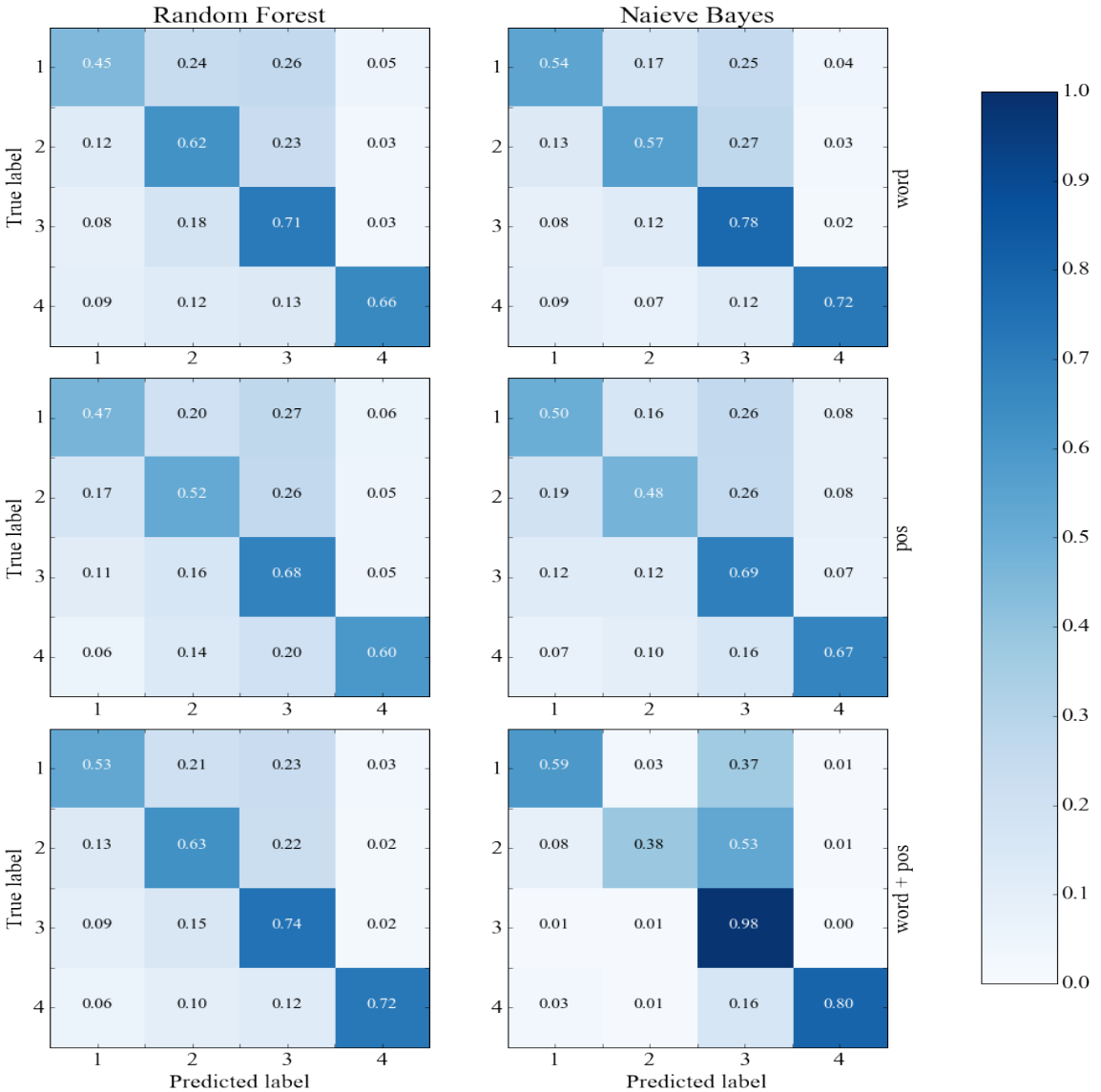

Figure 4.3: Confusion matrices for classification of dataset 1 with ngram range (1,3)

4.3 Decision nodes of trees

Finally, we will check the decision rules generated by decision trees. Table 4.4 contains decision rules up to the depth 2 nodes in the trees built with word, POS, and word + POS features. The trees are constructed with the training samples of dataset 30 without stop word removal. The n-gram range is set from 1 to 3. Though we use Random forests for actual classification, where each tree is constructed with a random subset of features, we here use decision trees for simplicity and clear decision rules.

The term 'which' appears in the trees constructed with word and word + POS features. The POS tag 'WDT', WH-determiner also have strong correspondence to 'which' by seeing the token 'NN which WDT' in word + POS feature. The decision nodes for word feature also see some general word collocations such as 'and the', 'it be', and 'some of'. Also, it catches 'ru' and 'te' in depth 2 nodes, which are the mis-translations of Kanji, Chinese characters. The tree build with POS feature mainly see the collocation around nouns, which are denoted by 'NN' or 'NNP'. 'NN WDT' probably has strong relation to the appearance of 'which' in word tree. In word + POS feature, 'NN which WDT' and 'NN which' correspond to 'which' in word and 'NN WDT' in POS features. Also, the token 'and CC the' corresponds to 'and the' in word tree. On top of that, it also catches the certain word choice, 'shall' and 'paragraph'.

TABLE 4.4

Decision rules in nodes up to depth 2: Each tree is built on the training set of dataset 30 with n-gram range (1, 3).

depth	word	POS	word+POS
0	which	NN WDT	NN which WDT
1	and the	CC DT	shall MD
1	which	NN VBG	IN for
2	it be	NNP PRP	and CC the
2	some of	NN TO BV	paragraph
2	ru	NNP PRP	NN which
2	te	NNP NNP	PRP$.

4.4 Comparison

In this section, we will compare the classification results among the translation set features from Aylin and Rachel [14], and topic model features from Suresh et al. [5]. Random Forest and Naive Bayes are used for the classifications with all 3 feature sets. The number of topics are set to 10, 15, 20, or 25, and topics are extracted by using LDA as mentioned in [5]. The results are the average of 4 results with different number of topics from 10 to 25 by 5 because there is no significant difference among the classifications with different topic sizes. Our feature is a collection of unigram, bigram, and trigrams of word + POS features based on its classification accuracy. Table 4.5 and Figure 4.4 compile those classification results. Figure 4.4 shows that all the 3 feature sets produce high accuracies on long-document-sized datasets 150, 30, 20, and 10, except for the poor performance of Naive Bayes and word + POS features. For short document size dataset 5, 3, and 1, however, our proposed method, word + POS feature produces better classification accuracies. This comparison shows that our proposed features are robust to short text classification comparing to existing studies, even though the methods that those previous studies proposed, Translation set and Topic model, work well on long document size dataset as in their reports.

TABLE 4.5

Classification accuracies of translation, topic model, and word + POS approaches.

	Translation set		Topic model		Word + POS	
Dataset	RF	NB	RF	NB	RF	NB
150	1.000	1.000	1.000	1.000	1.000	0.959
30	0.989	0.999	0.997	0.996	0.999	0.935
20	0.975	0.984	0.989	0.986	0.992	0.968
10	0.923	0.942	0.935	0.898	0.968	0.981
5	0.822	0.835	0.756	0.636	0.897	0.975
3	0.732	0.746	0.618	0.459	0.816	0.930
1	0.582	0.538	0.427	0.326	0.649	0.704

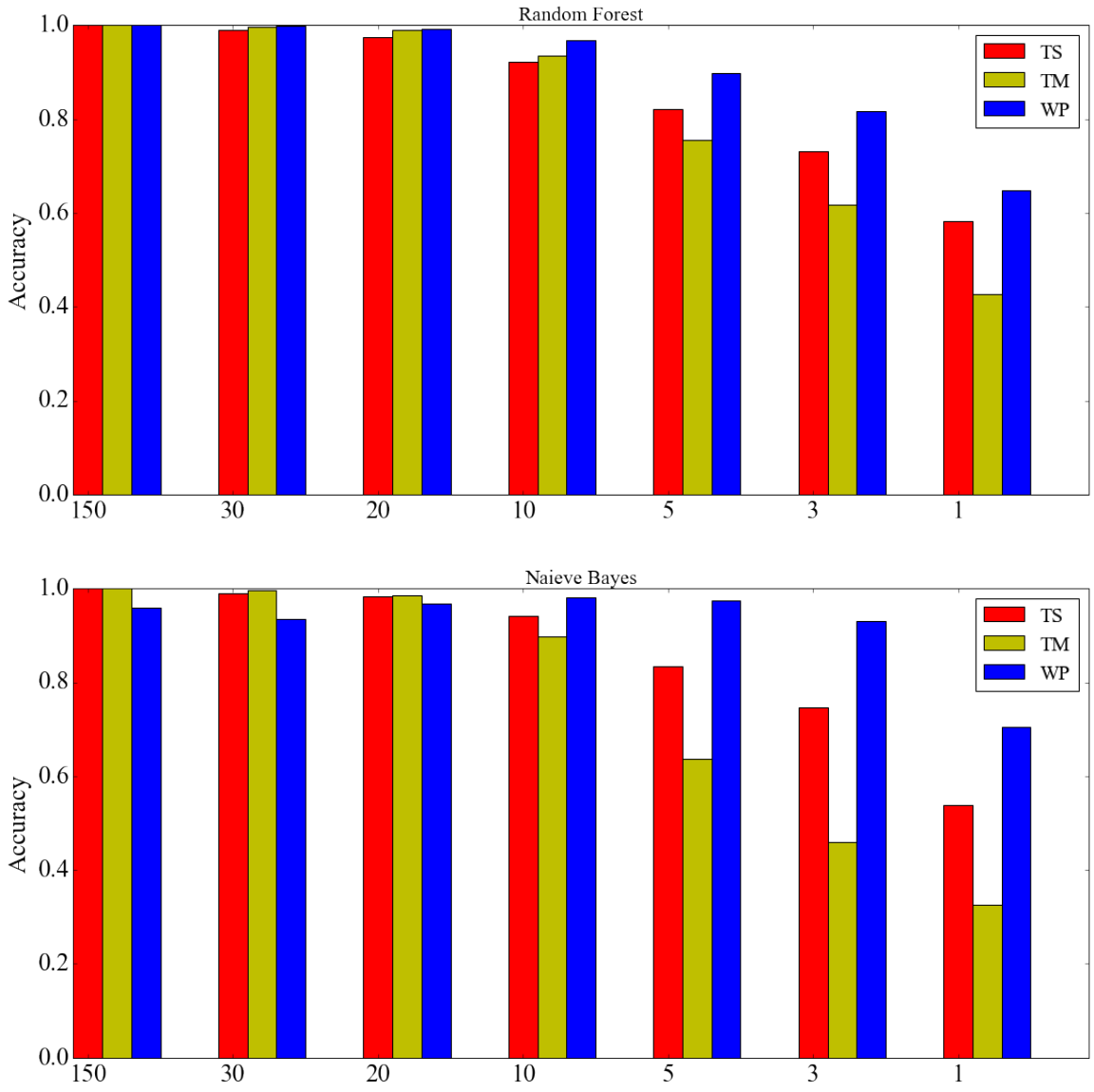

Figure 4.4: Classification accuracies of Translator set, Topic model, and word + POS features. The top graph is for the results of classification by Random forest and the bottom is for those of classification by Naive Bayes.

CHAPTER 5

CONCLUSIONS AND POTENTIAL FUTURE WORK

5.1 CONCLUSIONS

In this thesis, we investigate approaches to translator identification by testing different feature settings. Three features, word, POS tags, and the combination of word and POS tags, are selected for feature representation of texts with classification algorithms, Random forest and Naive Bayes. Also, the power of n-grams and stop word removal are checked in our experiments for better accuracy and analysis. Our experiments shows that the combination of unigrams, bigrams, and trigrams of extracted feature represented by word + POS produced the best results among the settings we have tested, by combining lexical and syntactic information. Our proposed method works as good as the ones previous studies proposed for long texts and even more it outperformed them in short text classification.

5.2 POTENTIAL FUTURE WORK

Our experimental results show that combining lexical and syntactic information improves classification results. Considering that, a potential future approach could be applying the classification models taking a sequence of lexical markers as an input and predicts with the marker order information, such as recurrent neural networks (RNNs).

We also could extend our experiments to other language pair translations. The sources of our dataset are novels written in Japanese and the documents in a dataset are normalized in terms of sentence lengths. To aim for the more realistic experimental scenario, the source should be obtained in the target domain such as the sources for translator detection of spam message should be spam messages.

REFERENCES

[1] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *ArXiv e-prints*, pp. 1–23, 2016.

[2] Efstathios Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[3] Patrick Juola, "Authorship Attribution," *Foundations and Trends R in Information Retrieval*, vol. 1, no. 3, pp. 233–334, 2006.

[4] Ahmed Abbasi and Hsinchun Chen, "Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace," *ACM Transactions on Information Systems*, vol. 26, no. 2, pp. 7:1–7:29, 2008.

[5] V. Suresh, Avanthi Krishnamurthy, Rama Badrinath, and C. E. Veni Madhavan, *A Stylometric Study and Assessment of Machine Translators*, pp. 364–375, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[6] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space," *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pp. 1–12, 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning, "GloVe : Global Vectors for Word Representation," *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[8] S Argamon-Engelson, M Koppel, and G Avneri, "Style-based text categorization: What newspaper am I reading?," *Proceedings of AAAI Workshop on Learning for Text Categorization*, pp. 1–4, 1998.

[9] Ying Zhao and Justin Zobel, "Searching with style: Authorship attribution in classic literature," in *Conferences in Research and Practice in Information Technology Series*, 2007, vol. 62, pp. 59–68.

[10] Xiaoyan Tang and Jing Cao, "Automatic Genre Classification via N-grams of Part-of-Speech Tags," *Procedia - Social and Behavioral Sciences*, vol. 198, no. Cilc, pp. 474–478, 2015.

[11] Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández, "Syntactic N-grams as machine learning features for natural language processing," *Expert Systems with Applications: An International Journal*, vol. 41, no. 3, pp. 853–860, 2014.

[12] David M Blei, Blei@cs Berkeley Edu, Andrew Y Ng, Ang@cs Stanford Edu, Michael I Jordan, and Jordan@cs Berkeley Edu, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[13] David Blei, Lawrence Carin, and David Dunson, "Probabilistic topic models," *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 55–65, 2010.

[14] Aylin Caliskan and Rachel Greenstadt, "Translate Once, Translate Twice, Translate Thrice and Attribute: Identifying Authors and Machine Translation Tools in Translated Text," *2012 IEEE Sixth International Conference on Semantic Computing*, pp. 121–125, 2012.

[15] Bing Liu, *Web Data Mining*, 2011.

[16] Eibe Frank and Remco R Bouckaert, "Naive bayes for text classification with unbalanced classes," *PKDD'06 Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, pp. 503–510, 2006.

[17] Peter Flach, *Machine Learning The Art and Science of Algorithms that Make sense of data*, Number X. 2012.

[18]

[19] Steven Bird, "NLTK: The Natural Language Toolkit," *Proceedings of the COLING/ACL on Interactive presentation sessions (COLING-ACL '06)*, pp. 69–72, 2006.

[20] Kim Luyckx and Walter Daelemans, "The effect of author set size and data size in authorship attribution," *Literary and Linguistic Computing*, vol. 26, no. 1, pp. 35–55, 2011.

[21] Moshe Koppel, Jonathan Schler, and Shlomo Argamon, "Authorship attribution in the wild," *Language Resources and Evaluation*, vol. 45, no. 1, pp. 83–94, 2011.

CURRICULUM VITAE

NAME: Keishin Nishiyama

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.Sc., Computer Science & Engineering
December 2017
University of Louisville, *Louisville, Kentucky*

M.A., Language and Culture, Shimane University
March 2011
Shimane University, *Shimane, Japan*

B.A., Education, Shimane University
March 2009
Shimane University, *Shimane, Japan*

PUBLICATIONS:

1. Faezeh Tafazzoli, **Keishin Nishiyama**, Hichem Frigui, "A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition", CVPR, Honolulu, HI, July 2017.

HONORS AND AWARDS:

1. 2nd place Graduate Research Award, 101st Kentucky Academy of Science, 2015