Detecting the source language of text translated by state-of-the-art Machine Translation models

Chih-Hsiang Hsu, Chung-Hao Liao, Antoni Maciąg, Jen-Tse Wei
National Taiwan University

Abstract

We investigate the possibility of detecting the source language of text machine-translated from Arabic, Chinese, Indonesian and Japanese. We create four different models, based on vanilla BERT, BERT on part-of-speech tags, SVM document-level features, and a neural network on dependency trees, respectively. The experiments show that relatively small models based on syntactic information only can achieve accuracies close to 60%.

Source language detection

Although there is some preexisting work on tasks very similar to the one we chose for the assignment, it is not a standard NLP task and to the best of our knowledge, there is no work with exactly the same problem formulation. The formulation is: given a text machine-translated into English from a known set of source languages, detect the source language. We will refer to the problem as Source Language Detection (SLD).

We are motivated by the following: in human translation, clues as to the original language in the form of both syntactic and semantic information tend to get unconsciously carried over to the translated text [1], making it possible to detect the original language. We are curious if that is also the case for currect state-of-the-art models for Machine Translation, and if so, what kinds of models will make SLD possible and what features of the translated text will be salient for detection. If the translation models are good enough, our obtained accuracy should not be considerably higher than random guessing.

In addition, we are curious if there is a difference between translation models which were trained multilingually and ones that were not. We hypothesize that for the former, the task might be more difficult because such models have learned on languages with various syntactic structures, which could make them less likely to carry the syntactic features of a particular source language over into the translation.

Related work

The closest work to what we attempted to do was done by Nguyen-Son et al. [2] who detect, for a given English text, whether it was translated or orignally written in English, and if translated, the correct one out of a set of possible source language - translator tuples. The possible languages are Russian, German and Japanese. They use the round-translation method, utilizing the phenomenon by which, while repeatedly translating a text back and forth between two languages, each round-trip changes the text less than the previous one. Thus, given an English text T which we know was translated from either Russian or German, if we generate round-trip $En \rightarrow Ru \rightarrow En$ and $En \rightarrow Ge \rightarrow En$

translations of the text, the similarity to T will be higher for the translation through the language that was the original language of T. Therefore, the authors generate round-trip translations of a given text through a number of languages and translators, and choose the translation with the highest similarity to T. Its associated language-translator tuple has its own subclassifier, which is further used to determine if the text was translated or originally English. The authors prove the ability of such a model to generalize (i.e. still detect the source language) to texts translated by translators not included in training. However, a shortcoming of such an approach is that it is computationally expensive, both while training (due to training a separate subclassifier for each language-translator pair), and during inference (due to generating multiple round-trip translations).

Kurokawa et al. [4] create a model based on Support Vector Machines (SVM), capable of determining whether a text was originally English, or translated from French. They find that certain n-grams were more frequent in translated text (semantic information), but syntactic features turn out ot be powerful as well, for example there is a "higher presence of the definite article *the* and prepositions in text translated from French", and "good classification accuracy was obtained even when texts were reduced to part-of-speech sequences".

Lynch & Vogel [1], similarly, construct an SVM based on document-level features such as the number of nouns, average sentence length, word unigrams, part-of-speech (POS) bigrams. Proper names are excluded from word unigrams, as "any character or place-names could unambiguously distinguish a text". Again, the word *towards* turns out to be particularly common in translation from German, and *that's* (rather than *that is*) - in translations from Russian. The document-level features suggest it might be easier to detect the source language of a text if it is longer, which makes such features more reliable (reduces their variance).

Approach

Chosen languages

As mentioned before, the SLD task involves detecting the source language from a given set of possible source languages. We chose Arabic, Chinese, Indonesian and Japanese to be our set. All of those are non-Indo-European languages with grammar fairly dissimilar to English, so we hypothesize that their syntactic features are carried over during translation to a bigger extent, making classification easier. For example, we hope that the rigid word order of Japanese and Chinese, which are known to be highly configurational languages, will manifest itself in the translations.

Dataset creation

The creation of the dataset can be divided into two stages: gathering text in the original languages, and machine-translating it into English.

In the first stage, we scraped Wikipedia articles in each language. We wrote a script which operated as follows: a pool of articles was maintained, initialized with a single article. Until a given number of text chunks was accumulated, the script selected a random, previously unvisited article from the pool. The contents of the article, regardless of their original division into sections, paragraphs etc. were concatenated into a single piece of text. From this piece, if it was sufficiently long, one or two chunks were selected, each composed of a sequence of whole sentences. Each chunk's length (in words) was 256 or more, but would have been less than 256 without the last sentence. Then a set number of links from the article, if available, was selected at random and added to the pool.

The random link sampling, and limitation of at most two chunks from each article, were included so as to avoid the situation in which the data for a given language includes particularly many proper names related to a given topic. In such case, a model could learn to "recognize a given language" by, in fact, learning that the language is associated with that topic. These measures resulted in a very diverse dataset, with - for example - the first few chunks in the Indonesian data featuring retina cells, millenials, a Christian holiday, an Italian company, and squirrels. There was also an overlap between languages, for example Arabic data featured Christians and squirrels as well.

We decided not to remove proper names due to these measures and the fact that directly removing them from the text (or, for example, replacing them with random nouns) would have resulted in incoherent text, especially after translation, and could have ruined syntactic information.

It should be noted that we made a mistake in the scraping script, by which some of the scraped paragraphs were duplicates. We suppose this was caused by Wikipedia redirections, which make previously unseen links direct the script to articles which in fact have been seen and scraped before. This was not a significant problem, for example about 2% of the Indonesian paragraphs were duplicates. These were removed before training.

Once the data in original languages was gathered, it had to be translated into English. We decided to generate, for each language, about 6400 chunks for the train set and further 700 for the test set. To avoid the situation in which a model learns the style of a particular translator, instead of the source language, we used a few different translators for each language.

Parts of the test set were translated using the Facebook large multilingual BART translator, since, as mentioned before, we were curious if there would be a difference in accuracy between text translated by a multilingual translator and not. In addition, since BART uses the same decoder for all languages, we suppose the translations it generates might be similar in style regardless of their source language, making it harder for the model to learn to recognize text in a given language by recognizing the styles of its associated translators.

The final dataset composition for each language:

- Arabic:
  - Train set: 1600 translated chunks from each Google Translate, the Helsinki-NLP/opus-mt-ar-en translator, and the fine-tuned one Shularp/krirk-finetuned-Helsinki-NLP-opus-mt-ar-en, Facebook large multilingual BART translator.
  - Test set: 350 translated chunks from the same Helsinki and MBART translators each with 700 chunks in total
- Chinese:
  - Train set: 2700 translated chunks from Google Translate, 2700 chunks from Helsinki-NLP/opus-mt-zh-en translator, and 2700 chunks from Face book large multilingual BART translator.
  - Test set: 350 chunks from Helsinki-NLP/opus-mt-zh-en translator and 350 chunks from Face book large multilingual BART translator.
- Indonesian:
  - Train set: 1600 chunks translated by Google Translate, the Helsinki-NLP/opus-mt-id-en translator, and the Facebook large multilingual BART translator each; 252 by the DeepL translator; 995 by the Microsoft translator API; and 330 by LibreTranslate; making 6377 chunks in total,
  - Test set: 350 chunks translated by the same Helsinki and MBART translators each - 700 chunks in total.

  These numbers were gathered before removing duplicate chunks, so the final numbers are, as mentioned, slightly different. In addition, due to maximum length limitations in the Helsinki translator, sentences with length (after tokenization) greater than 256 were removed.
- Japanese:
  - Train set: 1355 chunks translated by Google Translate, 2495 chunks from staka/fugumt-ja-en, 1043 chunks from Facebook large multilingual BART and 1649 chunks from Helsinki-NLP/opus-mt-ja-en. Make 6542 chunks in total.

  - Test set: 350 chunks translated from staka/fugumt-ja-en and 350 chunks from Facebook large multilingual BART with 700 chunks in total.

Models

Four models have been created:

- A model based on a pretrained BERT, as a straightforward but possibly powerful solution and a reference point for the others,
- A BERT on POS sequences, inspired by the effectiveness of POS sequences as mentioned in [4],
- An SVM, inspired in turn by [1],
- A neural network based on dependency tree convolutions and, again, POS tags.

More precise descriptions of each model are provided in the sections below.

Pre-trained model	Accuracy on wikipedia test data (%)	Accuracy on news test data (%)
Bert-base-cased	95.3	63.3
Twitter-Roberta-base	94.7	54.4

Table 1: Accuracy for different pre-trained model.

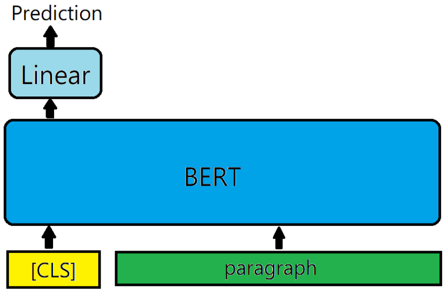

Figure 1: Bert model architecture.

Bert

First, we try a straightforward method, fine-tuning a Bert pre-trained model on this classification task. The architecture of the model is shown in Figure 1. The bert model takes tokenized paragraph as inputs, and the pooled output of it is fed into a linear model to predict the class.

We fine-tuned two pre-trained models, which are bert-base-cased and twitter-roberta-base-sentiment. The pre-training data of bert-base-cased includes BooksCorpus and English Wikipedia, which may has similar document properties to the training and test data we collect. On the other hands, the model twitter-roberta-base-sentiment is pre-trained on a lot of tweets, which is quite different from the training data and test data we collect.

The result has shown that both models reached more than 90% overall accuracy on our test data collected from wikipedia. However, the models take original text as inputs, which means that they can learn not only grammatical or structural features but also subjects related to training data in each language. Hence, we can not confirm that the models are globally great solely from their performance on wikipedia test data.

To verify whether the models learn the structural features related to original language or the subjects of our training data, we collect another test dataset. The new test data is collected from 80 news article in different languages translated by Google translator. The performance of both models on new test data dropped drastically. Bert-base-cased gets 63.3% of accuracy, and twitter-roberta-base gets 54.4% of accuracy. However, the accuracy is still much higher than that of random guessing. Hence, we can conclude that the Bert model is able to grab some of the structural of grammatical features related to original language. In addition, the performance of bert-base-cased is much better than that of twitter-bert-cased, which may result from their different pre-training data. The overall accuracy of both models are shown in Table 1.

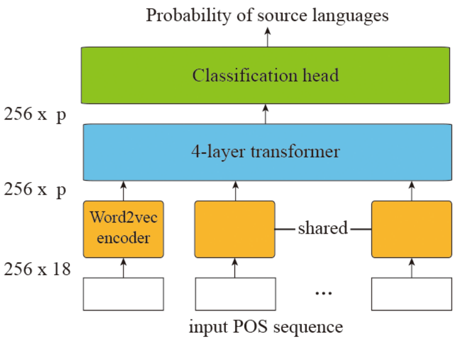

Figure 2: Model architecture on POS feature.

Model on POS sequence

Since the languages we chose are very different in grammar structure, we hope to identify the source language from the part-of-speech (POS) feature. This method is considered to be effective in [4]. Their SVM model benefits from the POS feature and achieves satisfying accuracy on detecting translated text.

We first transform our dataset to word-level POS sequences by vblagoje/bert-english-uncased-finetuned-pos, which is fine-tuned BERT on English POS tagging task. For example, a sentence "I want to eat an apple." will become a sequence "PRON VERB PART VERB DET NOUN PUNCT". There are 16 categories for POS tagging with an extra tag "X" for those unable to be recognized. Then we would like to see if this sequence can give enough information to source language detection.

This task can be formulated as text classification. We come up with using a pre-trained RoBERTa-based model on text sentiment classification. After several try, we find that the results are all barely better than guessing. The reason is mainly on the size of the model. Since there are only about 20 categories of POS tagging, the model seems to be too large for dealing with this and rapidly overfits the training set.

Thus we construct a smaller transformer-based model. The model can be split into three parts: a word-to-vector encoder, a 4-layer transformer, and a classification head as shown in Figure 2. We truncate and pad the POS sequence into fixed length of 256 words. Then one-hot encoded it into an 18 dimensional vector sequence $V \in \mathbb{R}^{256\times 18}$. Next, we use a multi-layer perceptron to encode the sequence to p dimensional embedding and get an embedded sequence $\hat{V} \in \mathbb{R}^{256\times p}$. Then the embedded sequence becomes probabilities on the 4 source language type after passing through the 4-layer transformer and the classification head. The result is shown in Table 2.

Embedding dim. p	Accuracy (%)	Parameters
16	56.6	14229
64	57.2	160581
256	53.4	2312709

Table 2: Accuracy on test set for different model structure.

We may see that a model with small amount of parameters can do source language detection well. Also, the POS feature provides enough information and allows our model to achieve a satisfying accuracy on this task.

SVM

Inspired by [1], we think that there are some clues hidden in the documents. Therefore, we mimic 14 documents-level features and transform these features into vectors. These 14 documents-level features are listed in Table 5. After doing the data pre-processing, we have a vector consist of 14 elements for each paragraph. We create a linear kernel SVM to do the classification.

Besides, we set up different parameter like constant C in linear kernel. Constant C means that how much the margin size of hyperplane will be set up. If the constant C becomes smaller, the margin size becomes larger which means that it will be easier to misclassify data. Table 3 lists the influences of constant C in linear kernel. However, the larger constant C will increase the calculation time. Therefore, it is a trade-off between constant C and training time.

Additionally, we use cross-validation technique to avoid of overfitting. The result is listed in Table 4. We can find out there is a little improvement in accuracy.

C	Accuracy (%)
1	48.95
2	50.14
5	50.42

Table 3: The accuracy for different constant C

C	Accuracy (%)
1	50.97
2	51.75
5	52.26

Table 4: The accuracy with 10-folds cross validation

Dependency tree CNN

Due to the effectiveness of source language detection based solely on syntactic information as encoded by POS sequences, we hypothesize that dependency trees, which are a rich source of such information and can encode long-range syntactic relationships missed by sequential processing, may prove powerful for SLD.

If a Deep Learning model is to be used, first it needs to be decided how to feed it the information about the dependency tree, i.e. how to represent it with a vector. Ma et al. [5] propose applying a convolution to the tree, as follows: first, vector representations of each word's k ancestors are concatenated to its own representation in the order of ancestry (with k being a hyperparameter). The vectors are padded if there are not enough ancestors. Thus we obtain a sentence representation in l vectors, with l being the length of the sentence. Then, n (which is also a hyperparameter) learnable convolution filters are applied to each vector, with each filter applying the dot product between its own learnable vector and the input, adding learnable bias, and applying a non-linear activation. Thus we obtain l features for each filter. Max-pooling is applied, i.e. the maximum feature is chosen for each filter. This way, we finally obtain an input representation with n features, which can then be fed to a neural network. In addition, the authors test the effect of including siblings in the convolution, but their results show this only results in a small increase in accuracy.

It might be worth noting that a similar approach, i.e. basing the encoding of syntactic information on node-to-root paths in dependency trees, is applied by a researcher from our University, in [7].

Zhang et al. [6] propose an alternative method, by which word representations are fed straight into the network, but the network is not fully-connected and its existing connections are used to represent the graph structure. A connection exists between neuron i in a layer and neuron j in the next layer only if $i = j$ or the i-th and j-th words are connected in the dependency tree. Additionally, the activations each neuron obtains from the neurons in the previous layer are normalized by the in-degree of that neuron.

One may notice that neither of these approaches encodes information about the nature of dependencies between words, only the structure of the dependency tree. In fact, Zhang et al. mention that including such information does not lead to any improvement in performance.

We based our model on the one in Ma et al. [5]. However, we also applied some modifications. Most importantly, as our word embeddings, we apply one-hot vectors representing POS tags, motivated by the succcess of earlier POS-based models, mentioned above, and by the objective to analyze syntax only, allowing us to drop semantic information. The dimension of these vetors is $P = 16$. We do not include dependency tree siblings in the convolution. We also simplify the convolution as our experiments show that such simplification actually leads to a slight increase in accuracy. In detail, the model works, as follows:

First, for each word in the input sentence, a vector representation is obtained just as in [5], by concatenating k ancestor vectors. k is set to 8, as, the dependency trees of most sentences from the dataset do not exceed 8 in height. If there are not enough ancestors, representations are padded with zeroes. Thus, the dimensionality of each representation is $D = P \cdot k = 16 \cdot 8 = 128$. The vectors are combined into a matrix $M \in \mathbb{R}^{D,L}$, with the vector representing the i-th word becoming the i-th column, and zeroes used for padding to length L. This matrix is then

Number	Documents-level feature	Description
1	Avgsent	Average length of sentences
2	ARI	Readability metric
3	CLI	Readability mertic
4	word ratio	word-type word(without numerals and others) : total words
5	num ratio	numerals : total words
6	verb ratio	verbs : total words
7	noun ratio	nouns : total words
8	pronoun ratio	pronouns : total words
9	prep ratio	prepositions : total words
10	conj ratio	conjunctions : total words
11	open ratio	open-class words : total words
12	closed ratio	closed-class words: total words
13	lemma ratio	lemmas : total words
14	grammlex	open-class words : closed-class words

Table 5: Description of each documents-level feature.

multiplied by a single, learnable convolution vector $c \in \mathbb{R}^L$, and a learnable bias vector $b \in \mathbb{R}^L$ is added to obtain the final representation $r \in \mathbb{R}^D$. Note that unlike in [5], the i-th feature of r is calculated as the dot procuct of c with the i-th features of all vector representations (plus bias), and not with one, whole vector representation. r is then fed to a *ReLU* nonlinearlity, followed by a fully-connected affine layer, another *ReLU* and a final affine transformation to logits.

The model, as described here, is trained and operates on a sentence basis. For fair comparison with the other models, inference for a whole paragraph is performed by running it separately for every sentence and then simply summing the output logits. The label corresponding to the highest logit sum is chosen as prediction.

Results

The testing results for each model-language-translator tuple, representing the percentage of samples from the test dataset corresponding to each tuple for which the source *language* was correctly recognized (we stress that the model was not tasked with recognizing the translator) are, as follows:

- BERT: (overall accuracy 95.3%, 63.3% on smaller dataset as described above)
  - arabic (overall accuracy 98.1%):
    - Helsinki: 99.4%
    - MBART: 96.8%
  - chinese (overall accuracy 95.0%):
    - Helsinki: 95.7%
    - MBART: 94.3%
  - indonesian (overall accuracy 95.7%):
    - Helsinki: 95.7%
    - MBART: 95.7%
  - japanese (overall accuracy 92.6%):
    - Staka: 93.1%
    - MBART: 92.0%
- POS (overall accuracy 57.2%):
  - arabic (overall accuracy 77.6%):
    - Helsinki: 78.8%
    - MBART: 76.4%
  - chinese (overall accuracy 59.6%):
    - Helsinki: 58.0%

    - MBART: 61.1%
  - indonesian (overall accuracy 44.6%):
    - Helsinki: 46.7%
    - MBART: 42.5%
  - japanese (overall accuracy 47.1%):
    - Staka: 53.1%
    - MBART: 41.1%
- SVM (overall accuracy 50.14%):
  - arabic (overall accuracy 57.57%):
    - Helsinki: 50.99%
    - MBART: 64.27%
  - chinese (overall accuracy 57.94%):
    - Helsinki: 55.58%
    - MBART: 60.37%
  - indonesian (overall accuracy 50.65%):
    - Helsinki: 51.00%
    - MBART: 50.29%
  - japanese (overall accuracy 30.00%):
    - Staka: 39.43%
    - MBART: 20.57%
- Dependency tree CNN (overall accuracy 59.4%):
  - arabic (overall accuracy 74%):
    - Helsinki: 66%
    - MBART: 82%
  - chinese (overall accuracy 60.7%):
    - Helsinki: 61.1%
    - MBART: 60.2%
  - indonesian (overall accuracy 85.8%):
    - Helsinki: 87.3%
    - MBART: 83.6%
  - japanese (overall accuracy 16.7%):
    - Staka: 19.4%
    - MBART: 14.3%

Aside from the BERT model when tested on the original dataset, the results are not dramatically different between the models, with accuracies between 50% and 60% for the POS, SVM, and D-CNN models, and 63% for the BERT model if testing on the smaller dataset. While this is a considerably worse result than that obtained in [2], we consider it acceptable, since it was achieved by fairly small models (e.g. 59.4% for the D-CNN with 17220 parameters) at a task which can be challenging even for humans.

Contrary to our expectations, there is no observable difference in detection accuracy between text translated by multilingual models and by non-multilingual ones.

Some languages are recognized with better accuracy than others. In general, Arabic is relatively easy, while Japanese - relatively hard to recognize. Big differences between the models (e.g. D-CNN does well on Indonesian and poorly on Japanese, while POS does slightly better on Japanese than on Indonesian) suggest that an ensemble of these models, which we have not tried creating, would outperform each of them.

Finally, the fact that BERT's accuracy highly depends on the dataset clearly shows that for tasks similar to ours, there is a serious risk of the results being falsified by the models learning the topic distribution in the dataset, rather than features of translated text.

Conclusion

We have shown that the syntactic features of text translated by SOTA Machine Translation models are enough to achieve a decent accuracy in detecting the text's original language, even with comparatively small models. Syntactic clues are enough, while not eliminating the possible influence of semantics is risky from the point of view of result veracity. While there are differences in accuracy between languages, the multilinguality of the translator does not seem to have a substantial impact.

We have a few suggestions for future work related to this project. One of them would be to create a model ensemble like [2], selecting models particularly good for respective languages, although that would increase model size, especially with a growing number of test languages. An accuracy comparison to humans who speak the test languages would be interesting to see, for example comparing an ensemble to a group of professional translators of the test languages.

To further investigate the role of syntax in detection, ablation studies could be conducted, for instance removing the parse-tree-based information from the D-CNN model.

One more shortcoming we would like to acknowledge is that we still included text translated by the same translators in the train and test datasets, creating a risk of the models learning the style of a particular translator rather than translation from a particular language. While this is mitigated by the fact that we used a few different models for the train set, the possibility that this falsified the results to some extent cannot be ruled out.

Work distribution

- Chih-Hsiang Hsu:
  - Arabic language dataset part, POS-BERT model, and their respective sections in this report.
  - Reviewing related literature.
  - Improvements to scraping.
- Chung-Hao Liao:
  - Chinese language dataset part, BERT model, and their respective sections in this report.
- Antoni Maciąg:
  - Formulation of the task.
  - Indonesian language dataset part, dependency tree CNN model, and their respective sections in this report.
  - Reviewing related literature.
  - Initial versions of the scripts for scraping, and translating with the MBART, Helsinki and Google translators.
  - Writing this report, except for the sections related to specific dataset parts and models.
- Jen-Tse Wei:
  - Japanese language dataset part, SVM model, and their respective sections in this report.
  - Reviewing related literature.

References

[1] Gerard Lynch and Carl Vogel. Towards the Automatic Detection of the Source Language of a Literary Translation. Proceedings of COLING 2012

[2] Hoang-Quoc Nguyen-Son, Tran Phuong Thao, Seira Hidano, Ishita Gupta, and Shinsaku Kiyomoto. Machine Translated Text Detection Through Text Similarity with Round-Trip Translation Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics, 2021

[3] Hoang-Quoc Nguyen-Son, Tran Phuong Thao, Seira Hidano, and Shinsaku Kiyomoto. Detecting Machine-Translated Text using Back Translation. Proceedings of the 12th International Conference on Natural Language Generation, 2019

[4] David Kurokawa, Cyril Goutte and Pierre Isabelle. Automatic Detection of Translated Text and its Impact on Machine Translation. In Proceedings of Machine Translation Summit XII, 2012

[5] Mingbo Ma, Liang Huang, Bowen Zhou, Bing Xiang. Dependency-based Convolutional Neural Networks for Sentence Embedding. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th

International Joint Conference on Natural Language Processing, 2015

[6] Yuhao Zhang, Peng Qi, Christopher D. Manning. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018

[7] Yun-Nung Chen, Dilek Hakkani-Tur, Gokan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. Knowledge as a Teacher: Knowledge-Guided Structural Attention Networks, 2016