# Classification of Mainland and Taiwanese Mandarin

Charles Sørbø Edvardsen

April 2019

## Abstract

Dialect identification is the challenging task of discriminating between varieties of the same language. This paper describes an effort at such a discrimination task by employing a number of machine learning (ML) and deep learning (DL) models to the task of distinguishing between Mainland and Taiwan Mandarin on the basis of single script versions of a balanced corpus of newspaper articles from the two places. Along the way some properties of Mandarin as well as some of the challenges of language variety identification are delineated briefly. The various models used are portrayed in brief and their performance accuracy noted, compared and discussed. The peak accuracy scores for the various models were very similar. An weighted voting classifier registered the highest peak accuracy at 88.5% on character bigrams, but the accuracy of ML models drops quite rapidly when n-gram size increases. The DL models consistently return good scores in the 80-range. My POS tagger on the other hand, which was only applied in combination with other ML models, only hovered in accuracy around 60%, probably much due to the complexity of Mandarin morphology. But all in all the performance of the models shows that even relatively simple models can achieve respectable results when tackling a challenging identification task.

## 1 Introduction

Chinese is the most widely spoken language in the world. It comes in many varieties, some of them so different that they can easily be considered separate languages from a purely linguistic point of view. Of these *Mandarin* is the most prestigious and widespread. It has official status in both Mainland China and in Taiwan, and is the variety on which Standard Chinese is based. The standard is referred to as Pǔtōnghuà (普通话 - common language) in China, and Guóyǔ (國語 - national language) in Taiwan. These are variants based on the Mandarin dialect of Beijing. The most striking difference between the Mandarin of China and Taiwan is the two different character sets employed in writing. In Taiwan *traditional characters* are used, whereas *simplified characters* are used

| | **Traditional** | **Simplified** | **Meaning** |
|---|---|---|---|
| **Same** | 水 | 水 | Water |
| **Modified** | 愛 | 爱 | Love |
| **Completely changed** | 聽 | 听 | Listen |
| **Multiple merged into one** | 1: 後，2: 后 | 1&2: 后 | 1:After, 2:Queen |

Table 1: Comparison of traditional and simplified characters

in China. The latter script was devised by the Chinese government in the 1950s and 60s, and subsequently adopted in all of Mainland China, in an attempt to improve literacy among the Chinese. While many of the characters remained unchanged or were modified only slightly, others were changed so radically that they bear little or no resemblance to the originals. In addition, characters with the same or similar pronunciations have from time to time been merged into single characters. A few examples of traditional and simplified characters are shown in table 1.

There are also differences in vocabulary. A number of words are written with identical characters, but have different meanings in the two areas. A case in point is is tǔdòu (土豆), which means *peanut* in Taiwan, but *potato* in Mainland China. And quite a few things and concepts are commonly referred to by distinct names. A tomato is, for instance, normally called xīhóngshì (西红柿) on the Mainland, but fānqié (番茄) in Taiwan. Pronunciation differences are also common. They will not be considered here, however, since the task presented by the organisers of VarDial 2019, and on which this paper is is based, only takes written language into account. For their discrimination task they have, therefore, prepared two versions of the same balanced cross-variety corpus; one exclusively in simplified script and the other exclusively in traditional characters, and asked the participants to predict descriminative labels for text instances in one or both of the tracks separately. The use of character form in the identification task is therefore prohibited. Had it been provided for, the identification task would have been rather trivial. Since the general problem of distinguishing between variants of the same language is the same for both versions of the corpus, I have opted to concentrate only on the version in simplified script. The task at hand is, therefore, to identify language variety in texts written in Mandarin Chinese; more specifically in sentences extracted from newspaper articles, each of which is to be automatically classified as of either PRC-Mainland or Taiwan provenance. The task is, therefore, one of language variety differentiation rather than one of similar language identification. Though the two share much common ground, the distinction is significant because the more similar the variants, the more challenging it is for language identification systems (LDSes) to discriminate between them: Differences between varieties of the same language are more subtle in both lexicon and grammar. And despite increasing accuracy in discrimination methods, these challenges persist [1]. One clearly noticeable cross-variety difficulty is the increased presence of unknown words, test-set words that were unseen in training sets, and which are likely to cause problems. In Mandarin this difficulty is compounded by a rich derivational morphology with over 4000 productive affix characters, many of which are quite ambiguous in their part of speech details. In addition, the inventories and use of affixes vary somewhat between the mainland and the Taiwan variety. I expect these facts to be reflected in the prepared corpus, at least to some degree. As a consequence, the amount of training data for many individual linguistic features may very well be quite meagre. And among the extensive affixal ambiguity, POS-tagging, for instance, will pose a substantial challenge [2].

## 2 Method

In this project we train some machine learning (ML) and deep learning (DL) models to perform binary classification on sentences written in Mandarin. They comprise four traditional machine learning models, one weighted voting algorithm, and four different deep learning neural networks. The machine learning models considered are Naive Bayes (Multinominal and Bernoulli), Linear Support Vector Classifier, and Random Forest Classifier. In addition, a model I have called Weighted Prediction will be considered. This model is based on a combination of the four ML classifiers just mentioned, and it is further described in section 6. The features for the ML models are based on TF-IDF n-grams, both character level and word level n-grams. In addition, POS tags will be considered.

For the deep learning methods, a general Deep Neural Network (DNN, consisting only of Dense Layers), Convolutional Neural Network (CNN), Long short-term memory (LSTM), and Bidirectional LSTM (BILSTM) will be considered. The models are given short definitions below.

### 2.1 Naive Bayes

A family of simple probabilistic classifiers which are based on applying Bayes' theorem with a naive assumption of independence between features. This makes it a very fast and efficient algorithm.

### 2.2 Linear Support Vector

The Linear Support Vector machine is a supervised machine learning algorithm that can be employed for both classification and regression purposes. LSVMs are based on the idea of finding a hyperplane; the line that best divides a dataset into two classes.

### 2.3 Random Forest Classifier

The random forest classifier creates a large number decision trees, each created from a random subset of the training set. Then, each document in the test set is run through each tree, and each tree will vote for which class it thinks the document belongs to. Finally, the votes are aggregated to decide the most likely class.

### 2.4 Deep Learning Neural Networks

A deep neural network is a neural network with a certain level of complexity, in the sense that it is a neural network with more than two layers. Such networks are often described as having an input layer and an output layer and at least a hidden layer between, and the networks are particularly good at dealing with unstructured data.

The deep learning neural network framework is made up of three parts: an input embedding layer, a sentence encoding layer, and an output layer. Given an input sentence of n tokens, the input embedding layer represents each token as a vector. The sentence encoding layer creates a sentence representation on the basis of the word vectors. Finally, the sentence representation is fed through the hidden layers to obtain the output.

### 2.5 Convolutional Neural Network (CNN)

A convolution is basically a sliding window function applied to a matrix. A CNN network is a combination of several layers of such convolutions, supplemented by pooling layers, and fully-connected layers. CNNs are good at capturing abstract features, and they can identify them in the sentence regardless of their position.

### 2.6 Long short-term memory (LSTM)

Long Short-Term Memory network is a type of recurrent neural network that is capable of learning order dependence in so-called sequence prediction problems: prediction problems that involve sequence data. A Bidirectional LSTM use both a forward and a backward LSTM to process sequences.

## 3 Data

The simplified Mandarin version of the original corpus (which was drawn in equal measure from Taiwan and Mainland China) constitutes my data set. This set is further separated into training, validation, and testing sets, of sizes 18,770, 2000, and 2000 respectively. In the training and validation sets each sentence is assigned a label indicating its provenance: either “M” for Mainland or “T” for Taiwan. As the test set contains no labels, it cannot be used for evaluation. To compensate for this, the validation set is split in two; one half used for validation and the other for testing the (primarily for ML models).

Since punctuation removal and tokenisation had already been applied, only minimal preprocessing was needed. For easy manipulation the data was transformed to Pandas DataFrames. In addition, the labels “M” and “T” were replaced by 0 and 1, as the deep learning models required numerical labels. For character input, the spaces between the words were removed, so that the vectoriser could read each character directly one by one. It was also necessary to pad the sentences when working with the deep learning models, as several of them required sentences of the same length. For POS tagging, the Stanford Tagger was used.

## 4 Implementation

The machine learning algorithms were implemented using the scikit-learn library, while the deep learning models were implemented in Keras with Tensorflow backend. All models were trained and evaluated on the simplified character version of the corpus. The evaluation was with respect to accuracy. As the classes were even in number, a simple accuracy measure should give a sufficiently accurate picture of the models' performance.

### 4.1 Weighted Prediction

Examining the confusion matrices for the different models, I noticed that some models were better at correctly classifying texts from Mainland China, whereas others performed better on texts from Taiwan. This observation led me to devise the weighted_prediction classifier in an effort to capture the combined strengths of each model. It is essentially a voting classifier with weights, as all the ML models are run implicitly from Weighted Prediction. Simply put, the classifier runs through each sentence in the corpus once for each ML model defined by the code. Each model is then assigned two scores, an M_score and a T_score, proportional to its success in the classification of Mainland and Taiwanese sentences in the validation set respectively. The idea is that the better a model performs on Mainland text the greater say it will have when that model votes for a text being from the Mainland. Conversely, the poorer a model performs on Taiwanese text, the less weight it will carry when it votes for a sentence being Taiwanese.

More specifically, the weighted_prediction classifier iterates through the models, training each model on the training set, and running it on the validation set, before returning each models' M_score and T_score. The model and the scores are then appended to a list. When this has been completed for all models, each sentence in the test set will be run through every model. A dictionary is kept that keeps the score for both class M and T. If the model predicts that the text is from the Mainland, the score for M will be incremented by the M_score of that model. In the same way, the score for T will will be incremented by that model's T_score if it predicts the sentence to be from Taiwan. All predicted labels are appended to a list and transformed into a DataFrame, which is finally compared to the DataFrame containing the actual labels to obtain the accuracy.

### 4.2 POS classifier

The POS classifier is implemented as the pos_tagging_classification() function. This function creates the same kind of dataframe as before, but now it runs them through the pos_tag() function, which returns dataframes with the POS sequence of the sentence, rather than characters or words. It can return the sequences for both individual characters or for words, depending on whether we want to look at character or word level.

Next, the pos_tagging_classification() function creates a TF-IDF vectoriser similar to before, but instead of looking at each n on its own, n-gram range is set from 1 to 10, which will make it return only the size of n that optimises the accuracy. The vectorised POS n-grams are run through the Weighted Prediction classifier.

### 4.3 Deep Learning Networks

Each DL network is realised in a separate file. Except for the Dense Neural Network, the Keras tokeniser is used for vectorisation instead of the TF-IDF vectoriser used for the ML models. The sequential model is used, where appropriate layers are added to create the model for each DL network. “Adam” was used for the optimisation function, while “binary crossentropy” was used for the loss function. The models are trained for up to 10 epochs, but if there is no improvement for 4 epochs they will terminate. The models and model weights are saved so that they can be used again later without having to retrain the models.

## 5 Results

This section presents the results of the different models. They will be discussed further in the next section.

| | **MNB** | **BNB** | **LSVC** | **RFC** | **WeightedPrediction** |
|---|---|---|---|---|---|
| **Unigram** | 0.820 | 0.826 | 0.821 | 0.796 | 0.816 |
| **Bigram** | 0.875 | 0.881 | 0.857 | 0.794 | **0.885** |
| **Trigram** | 0.817 | 0.828 | 0.823 | 0.674 | 0.838 |
| **4gram** | 0.709 | 0.744 | 0.748 | 0.604 | 0.756 |
| **POS** | 0.588 | 0.579 | 0.620 | 0.588 | 0.601 |

Table 2: The accuracy of the different ML models on the test set on character level. Models from the left: MultinomialNB, BernoulliNB, LinearSVC, RandomForestClassifier, WeightedPrediction

| | **MNB** | **BNB** | **LSVC** | **RFC** | **WeightedPrediction** |
|---|---|---|---|---|---|
| **Unigram** | 0.873 | 0.879 | 0.850 | 0.850 | 0.877 |
| **Bigram** | 0.659 | 0.703 | 0.701 | 0.598 | 0.659 |
| **Trigram** | 0.547 | 0.546 | 0.547 | 0.541 | 0.533 |
| **4gram** | 0.515 | 0.510 | 0.515 | 0.515 | 0.505 |
| **POS** | 0.616 | 0.619 | 0.611 | 0.594 | 0.628 |

Table 3: The accuracy of the different ML models on the test set on word level.

| | **DNN** | **CNN** | **LSTM** | **BILSTM** |
|---|---|---|---|---|
| **Characters** | 0.879 | 0.832 | 0.794 | 0.810 |
| **Word** | 0.856 | 0.875 | 0.878 | 0.867 |

Table 4: The accuracy of the neural networks.

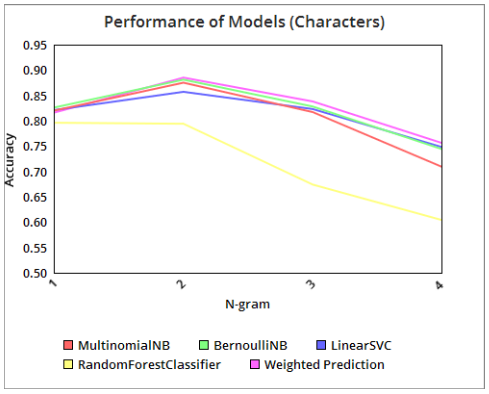

Figure 1: Accuracy of the machine learning models on character n-grams.

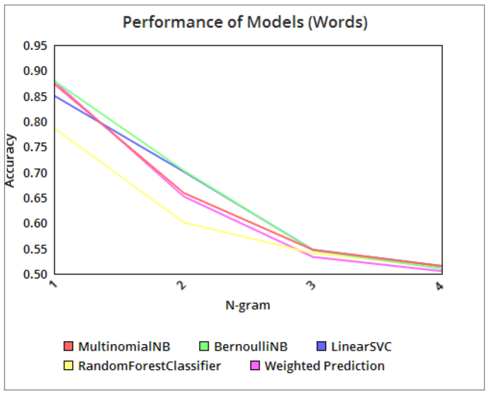

Figure 2: Accuracy of the machine learning models on word n-grams.

## 6 Discussion

As seen in figure 1 and 2, the overall trend is that the models tend to perform well on unigrams at the word level, but that their accuracy drops quite rapidly with increasing n-gram size. At the character level most of the models perform well when n-gram size is 3 or less. The performance is particularly good for bigrams. This dovetails with result found for unigrams at the word level. The result is unsurprising given the fact that Mandarin words are generally very short, averaging no more than 2.4 characters, and the majority of words have less than 3 characters [2].

The Weighted Prediction classifier reached an accuracy of 88.5% for character level bigrams. This was the highest accuracy recorded for any model, but only marginally better than the much faster and far less computationally expensive naive Bayes models. Of the machine learning models, the Random Forest Classifier appears to be the weakest one. It is also very slow, but its performance could possibly be enhanced somewhat by optimising the input parameters. Such optimisation would probably also benefit the Weighted Prediction model.

### 6.1 POS classifier

The POS tagger classifier is applied in combination with other ML models. Its accuracy score hovers around 60%, with a peak at 62.8% on word n-grams in tandem with the Weighted Prediction model. The high complexity of Mandarin morphology (which has no inflectional affixes, only a few dervational ones, and very rich compounding processes as well), is probably one of the main reasons for the modest accuracy score. POS tagging is considered a challenging task across languages, but there is probably ample room for improvement, partly in combination with DL models, and partly by discovering more sophisticated linguistically motivated features.

### 6.2 Deep Learning Models

There are only small differences between the accuracy scores the four DL models considered here. This is the case for both types of input, though the differences are a bit larger for character string input. The overall accuracy is also highest for word string input. The differences in performance between the two LSTM models are negligible in spite of the fact that the BiLSTM model is able to capture information from the context in opposite directions by being equipped with both forward and backward LSTM. It should, therefore, in principle be better equipped to pick up or even forget relevant grammatical information or other connections between words, but there is no evidence for this in the performance of my simple models. The CNN model performs just as well as the two LSTM models despite the fact that it just considers the current input. LSTMs, by contrast, consider both current and previously received input. They should, therefore, be superior to CNNs in handling sequences, and therefore superior to CNNs in matters of syntax and morphology. CNNs, on the other hand, are particularly good at capturing abstract features, and they can identify them in a sentence or word regardless of succession. This may very well be highly suitable for analysing the complex morphology of Mandarin and explain their performance vis-à-vis the LSTMs. Combining the two types in the right way, or experimenting with different layers, may very well yield better results in language variety identification.

## 7 Conclusion

The outcomes of the classification experiments indicate that there exist a number of fairly robust differences between the Mainland and Taiwan variety of Mandarin that can be exploited to distinguish between them. The results also clearly suggest that purely lexical differences are the strongest differentiators in the classification process, whereas grammatical properties have been utilised much more sparingly by the models I have considered.

## References

[1] Chris van der Lee and Antal van den Bosch. "Exploring Lexical and Syntactic Features for Language Variety Identification". In: (). URL: https://aclweb.org/anthology/W17-1224.

[2] Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. "Morphological features help POS tagging of unknown words across language varieties". In: *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing* (2005). URL: https://web.stanford.edu/~jurafsky/sighan_pos.pdf.