# Unicode-based Data Processing for Text Classification

**Akash Sedai** *akash.sharma@ucl.ac.uk*
*Institute of Finance & Technology*
*University College London*
*London, WC1e 6BT, UK*

**Ben Houghton** *benhoughton@quantexa.com*
*Quantexa Ltd*
*London, SE1 7ND, UK*

**Abstract**

In this paper we demonstrate a Unicode based text data processing approach for machine learning classification. The fields are first converted to Unicode, and then the features are generated by splitting the characters by vowels or any custom set of delimiters contained within fields. The fields are labelled into classes and the model outputs the class predictions for each field. It provides a simpler approach for text preprocessing that can maintain high accuracy. It will be useful to database managers or researchers who work with large unlabeled datasets that needs to be labelled into several classes.

**Keywords:** NLP, Classification, Text Processing, Machine Learning, Labeling.

## 1. INTRODUCTION

Text classification, also known as text categorization, is a classic problem in natural language processing (NLP) and information retrieval. It aims to assign labels or tags to textual units such as sentences, queries, paragraphs, and documents. It has a wide range of applications including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation, and so on. Text data can come from different sources, including web data, emails, chats, social media, tickets, insurance claims, user reviews, and questions and answers from customer services, to name a few. Text classification is a sophisticated process involving not only the training of models, but also numerous additional procedures, e.g., data pre-processing, transformation, and dimensionality reduction (Mironczuk and Protasiewicz, 2018). It remains a prominent research topic, utilizing various techniques and their combinations in complex systems. Furthermore, researchers are either developing new classification systems or improving the existing ones, including their elements to yield better results, i.e., a higher computational efficiency (Wang et al., 2013; Mironczuk and Protasiewicz, 2018).

Text classification can be performed either through manual annotation or by automatic labelling. With the growing scale of text data in industrial applications (Statista, 2022) and academic research (Kumar et al., 2021; Zhu and Lei, 2022), automatic text classification is becoming increasingly important. Approaches to automatic text classification can be rule based or machine learning (data-driven) based models (Asahiah, 2021; Bhattacharyya et al., 2021). Rule-based methods classify text into different categories using a set of pre-defined rules and require a deep domain knowledge while machine learning based approaches learn to classify text based on observations of data (Asahiah, 2021; Bhattacharyya et al., 2021). Using prelabelled examples as training data, a machine learning algorithm learns inherent associations between texts and their labels. While rule-based methods can supplement our approach by filtering out the more obvious entries, especially through keywords and regex methods, in this paper we will demonstrate only the outputs obtained from machine learning models. We will show the results for neural network, logistic regression, extra trees, random forest, QDA, and Gaussian naive bayes.

## 2. LITERATURE REVIEW

While the earliest use of statistical NLP dates to the 2000s, recent variations of NLP based on supervised machine learning and annotated data began thriving just over a decade ago. The improvement in computational power/parallelization (Raina et al., 2009; Coates et al., 2013) and big data has allowed for successful processing and implementation of sophisticated models with deep architectures (Otter et al., 2020).

Computational linguistics methods require encoding of text into numbers. Machines cannot process raw text as inputs and outputs by the ML models. The process we describe converts text into meaningful numeric or vector representation such that the context and relationship among characters in sentences are preserved and is sufficient for the machine to pick the signals associated with different combinations of letters, digits, non-numeric characters, and context in sentences. Several text encoding methods are suggested in the literature. Some of the most popular ones include index-based encoding, bag of words, TF-IDF encoding, word2vector encoding (Mikolov et al., 2013) and BERT encoding (Devlin et al., 2019).

Data labelling deals with the process of identifying the class of raw data and adding meaningful contextual labels. Researchers and industry practitioners use software, processes and data annotators to clean, structure and label data. Compared to unlabeled data (which is sufficient for unsupervised models but not supervised models), labelled data is more difficult to achieve. E.g. indicating whether a photo contains an animal or an object, sentence delivered by a person was positive or negative, or if an x-ray contains a tumor. Although labelled data provide more precise predictions and better usability, its drawbacks include being time consuming, expensive, and prone to human-error.

There is an increased interest in the exploration of ontologies and corpus-based statistical knowledge. Within the text classification domain, standard kernel functions such as linear kernel can be replaced with customized kernel functions exploiting this background knowledge, which then allows for a better performance when used with kernel-based method SVM (Altınel et al., 2015). A semantic smoothing kernel for SVM based on a meaning measure calculates the meaningfulness of the terms in the context of classes. Another common approach with the domain is based on Dissimilarity Representation and multiple classifier systems to counter the issue of high dimensionality (Pinheiro et al., 2017). Each dissimilarity space reduces the dimensionality, feature-to-instance ratio, and sparseness within the original space. A variation of the problem is a multi-label text classification problem whereby each document can belong to any number of classes (labels). Multi-label classification assigns multiple labels to each document contemporaneously. MLC is divided into two classification types: flat and hierarchical. In flat classification, a set of predefined labels are classified without considering the hierarchy of the relationship between the labels (Duwairi and Al-zubaidi, 2011). In a hierarchical multi-label classification (HMC), on the other hand, a single instance may have multiple labels consecutively, and the labels are interrelated by a categorical hierarchy (Brucker et al., 2011) Many real-world classification problems can be naturally structured in a hierarchy and tend to employ high-dimensional label spaces where each instance may belong to multiple labels. The corresponding classification algorithm considers hierarchical relationships between labels allowing for the prediction of multiple labels, and this can present more complexities compared to the flat classification (Aljedani et al., 2021).

There is also a growing interest in the literature to include text classification solutions for languages other than English. E.g., multi-label text classification for the Arabic language was explored by Aljedani et al. (2021).Xin and Zhang (2020) ran experiments on the description of 500,000 Chinese products with label using a convolutional neural network based multilabel text classification of Chinese text. Similarly, based on data that included hostile and non-hostile texts collated from social media platforms, Joshi et al. (2021) investigated hostile text detection in the Hindi language, further dividing the hostile posts into overlapping classes of fake, offensive, hate, and defamation. For other literatures in text identification for non-Latin languages see (Altamimi and Teahan, 2019;

Alkhazi and Teahan, 2019; Sumamo and Teferra, 2018; Meshesha and Solomon, 2018; Patil and Patil, 2016) for languages with Latin text other than English see (Mercelis, 2021; Chorozoglou et al., 2021; Malema et al., 2018, 2016).

## 3. METHODOLOGY

Our methodology is an inductive one that develops and tests a text data processing method for machine learning classification using publicly available data. The steps are as follows:

1. Convert text to Unicode. This makes each training point a long string of numbers where each pair of numbers represent a certain letter, digit or characters. E.g, the unicode of ”a”, ”-”, ”1”, and ” ” correspond to ”97”, ”45”, ”49” and ”32”. All fields are converted to unicode as seen in table 1

```
def unicodeConversion(field):
  data = []
  for i in range(len(field)):
    data.append(([ord(j) for j in field[i]]))

  data = [list(map(str, i)) for i in data]
  return [''.join(i) for i in data]
```

**TABLE 1:** Conversion examples.

| Latin | Unicode |
|---|---|
| Gower Street, London | 7111111910111432831161141011011164432761111101001111 10 |
| 0123456789 | 48495051525354555657 |
| +(44)712345xxxx | 434052524155495051525312012012012 0 |

2. Primary feature creation. We pick a set of characters and use them as keys to split the unicode into features. In our examples below we used a list of unicode characters which correspond to vowels, space, dot, and hyphen.

$k = '97', '101', '105', '111', '117', '32', '45', '46'$. For each time one of the keywords occurs in an unicode entry, the character splits from the point of occurrence, creating a new feature. The number of splits can be set as any number greater than 0. An appropriate number of splits will depend on the data that we want to classify but generally this would be correlated with the length of string entries in the dataset.

3. Secondary features creation. This include features such as character counts, space counts, and counts of any other keywords that are correlated with a particular field type. The usage of this further improved robustness of the models. The application of this methodology is demonstrated with three examples: field identification, review classification, and fake news identification. We report the outputs from best performing model for each task.

Step 1 (converting to unicode) and step 2 (primary feature creation) in the list above was also tested in reverse order. When the text was first split and then converted to unicode the results were inferior. A unicode output of a string has two or three strings. This may cause some of the text to be ”wrongly” split in the post conversion split. E.g. in the first example in 1, when splitting the text using vowel delimiters, the first word, ”Gower”, will be split at ”o” and ”e”. However, if the split is performed post-conversion, the equivalent delimiters are ”111” and ”32”. Since the unicode

of ”G” ends with ”1” and that of ”w” begins with ”11”, together the first three strings make up ”111111”, leading to double splitting. Despite such occurrences, this order of feature extraction performed better. This is likely the result of information being salvaged from correlation among some words that appear together.

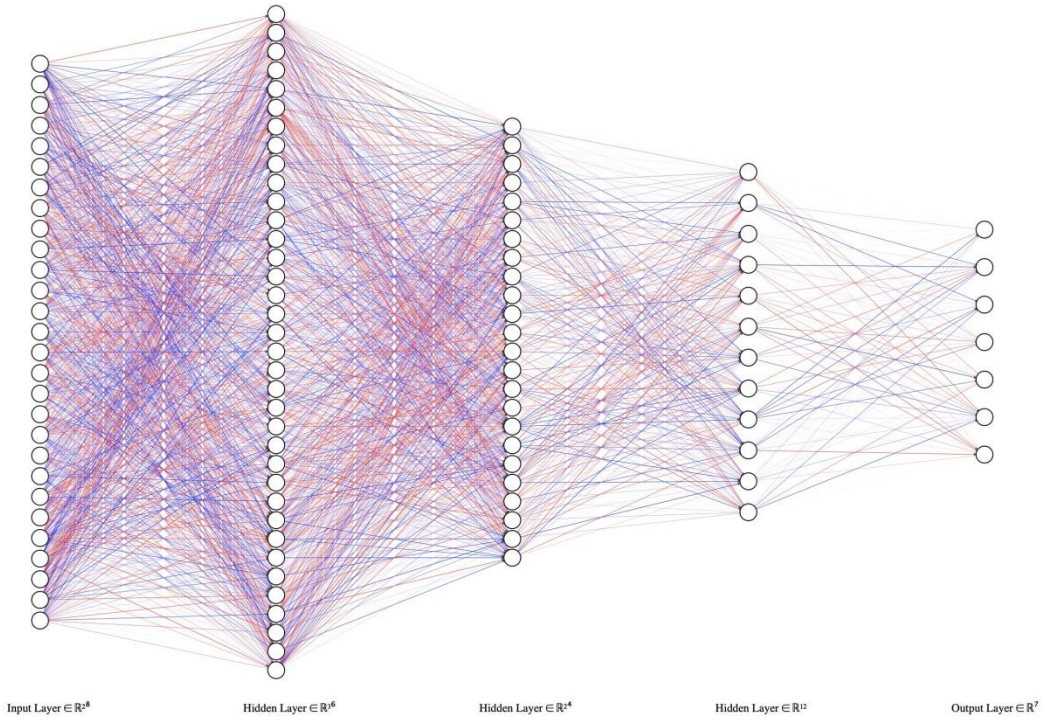


**FIGURE 1:** Neural network architecture.

The input data $X \in R^{ND}$, contains $N$ observations, with $D$ = 28 features. For performing hyper parameter tuning of neural network we use 20% of in-sample data as validation sample. The training sample is used to train several models with different architectures and specifications, while the validation set is used for selecting the best parameter setup and avoid overfitting in the training dataset. We used the ADAM algorithm for optimization which is a computationally efficient optimizer requiring less memory space. It is also known to work well on problems with noisy or sparse gradients as well as large datasets and large parameters.

We used cross-entropy for loss function. It is one of the widely used functions for classification problems that minimizes the distance between the predicted and actual distributions. E.g. if a prediction probabilities of three classes is given by $[p_1,p_2,p_3]$ where $p_1+p_2+p_3 = 1$ and the true outcome is [1,0,0], we want the predicted probabilities to be close to this original probability distribution. Cross entropy ensures the difference between the two probability distributions are minimized.

In developing our model specifications, we also examine an extended set of variables that describe the properties of field group in the sample. These include different numbers of splits and string delimiters other than vowels such as consonants, numbers and periods. We also explored other features of the variables such as whether they are numeric, alphanumeric, number of spaces, etc or some combination of these properties.

## 4. EXPERIMENTAL PERFORMANCE

The hardware environment used for this experiment is intel i7-1165G7 and 16GB DDR4 memory. The model training is supported by python 3.5 running on Windows 10 pro. For the implementations, Tens or flow library with Keras API was used. The data are collected from various sources including offshore leaks, Bahama leaks, Panama papers, Paradise papers, OpenStreetMap (OSM), U.S. Small Business Administration (SBA), Federal Deposit Insurance Corporation (FDIC), Kaggle, and Python Faker library. The details of the sources are given the table below. OSM data includes some addresses in non-Latin script and ”Others” include data from countries with counts <1000.

**TABLE 2:** Data sources.

| Field type (target class) | Location | source |
|---|---|---|
| Address | UK, US, Germany, Others | Offshore leaks, Panama papers, Paradise papers, SBA, OSM |
| Individual names | UK, US, France, Germany, Italy, China, India, Others | Offshore leaks, Panama papers |
| Business names | Mixed | Bahama leaks, SBA, FDIC |
| Dates | N/A | Faker |
| Email/phone/individual | UK / US | Faker |
| Unclassified | N/A | Faker, SBA, Kaggle, OSM |

The classification results of each field are shown in table 2. The outputs are from the deep learning model which is one of our best performing models on par with the tree-based models. The overall accuracy and computation time of six models are shown in table 3. Table 2 shows the sample for each label which indicates their uneven distribution. The numbers on the ”size” represents the dataset which was used for testing which is 20% of the entire dataset. As can be seen from table 2, the accuracy DL and other models are at 99% while logistic regression had a test accuracy of 97%.

**TABLE 3:** Field identification confusion matrix.

| | address | business | dates | email | individual | phone | unclassified | test size |
|---|---|---|---|---|---|---|---|---|
| address | 99.62 | 0.35 | 0 | 0 | 0.02 | 0 | 0.01 | 316466 |
| business | 3.18 | 94.65 | 0.01 | 0.42 | 1.74 | 0 | 0 | 70384 |
| dates | 0 | 0 | 100 | 0 | 0 | 0 | 0 | 54785 |
| email | 0 | 0.01 | 0 | 99.99 | 0 | 0 | 0 | 59886 |
| individual | 0.07 | 1.24 | 0.01 | 0 | 98.68 | 0 | 0 | 62841 |
| phone | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 60022 |
| unclassified | 0 | 0 | 0.05 | 0 | 0 | 0.02 | 99.93 | 44413 |

Results from neural network model

**TABLE 4:** Test accuracy and training times.

| Model | Test accuracy | Train time |
|---|---|---|
| Deep learning (Neural network) | 99% | 380 secs |
| Logistic regression | 97% | 297 secs |
| ExtraTrees | 99% | *<75 secs* |
| RandomForest | 99% | *<75 secs* |
| QDA | 82% | *<75 secs* |
| GaussianNB | 73% | *<75 secs* |

Most NLP preprocessing methods including those adopted by spaCy, gloVe, BERT, word2vec, etc require several steps in data processing which may include include tokenization, punctuation, stop words, contractions, alpha numeric characters, stemming/lemmatization. Most of these steps are not required in our method explicitly, except we use a variation of the tokenization which essentially can include some aspects of punctuation, stop words, contractions in one step. Stemming and Lemmatization are widely used in tagging systems, indexing, SEOs, Web search results, and information retrieval. Both stemming and lemmatization are not required in our approach and have not been used to model the examples but can be incorporated if preferred. We also do not include any steps to account for part-of-speech tagging or collocation/combination of phrases that are often used in NLP models. Besides being a stand-alone method to process texts data, our approach can also be used to improve existing methodologies through creation of further inputs to any supervised machine learning classification model. It may also be further augmented with existing processing methods such as bag of words which can potentially improve performance in some domain or when the dataset in question is more complex.

## 5. CONCLUSION

Text data requires approach that is different to those used in numerical data in machine learning modelling. This is because text data can have large dimensions (words and phrases); the English language has around 100,000 words in common use. However, datapoints in most text datasets often contain less than 100 words. Text categorization systems are designed to classify documents into a fixed number of predefined categories. Bag-of-words is one of the most used approaches to represent a document. However, it generates high-dimensional sparse data matrix with a high feature-to-instance ratio. An aggressive feature selection can alleviate these drawbacks, but such selection degrades the classifier's performance. The approach suggested in this paper mitigates this issue by simply splitting the characters by pre-specified global keywords. The advantage of the proposed model lies in the ease of preprocessing, suitability with simpler models, and high accuracy. This approach can reduce the number of steps required to prepare data for model training and maintain high accuracy even when using simpler models. It will be useful to database managers or researchers who work with large number of unlabeled data that needs to be labeled into several classes. For future work we consider incorporating a reinforcement learning for the search of best set of unicode delimiters. The delimiters used in the model presented in this paper is based on vowels of English characters; an intuitive choice based on their occurrence in the English language. A more comprehensive set of delimiters could be used that can optimize representation different classes. This can potentially be more useful when working with less familiar languages or languages that do not explicitly have a paradigm that classifies the characters.

## 6. REFERENCES

Aljedani, N., Alotaibi, R., and Taileb, M. (2021). Hmatc: Hierarchical multi-label arabic text classification model using machine learning. *Egyptian Informatics Journal*, 22(3):225– 237.

Alkhazi, I. S. and Teahan, W. J. (2019). Compression-based parts-of-speech tagger for the arabic language. *International Journal of Computational Linguistics*, 10:1–15.

Altamimi, M. B. and Teahan, W. J. (2019). Arabic dialect identification of twitter text using ppm compression. *International Journal of Computational Linguistics*, 10:47–59.

Altınel, B., Can Ganiz, M., and Diri, B. (2015). A corpus-based semantic kernel for text classification by using meaning values of terms. *Engineering Applications of Artificial Intelligence*, 43:54–66.

Asahiah, F. O. (2021). Comparison of rule-based and data-driven approaches for syllabification of simple syllable languages and the effect of orthography. *Computer Speech Language*, 70:101233.

Bhattacharyya, R. P., Jung, S., Kruse, L., Senanayake, R., and Kochenderfer, M. J. (2021). A hybrid rule-based and data-driven approach to driver modeling through particle filtering. *CoRR*, abs/2108.12820.

Brucker, F., Benites, F., and Sapozhnikova, E. (2011). Multi-label classification and extracting predicted class hierarchies. *Pattern Recognition*, 44(3):724–738.

Chorozoglou, G., Zacharis, N. Z., Papakitsos, E. C., Galiotou, E., and Giovanis, A. (2021). Review of parsing in modern greek - a new approach. *International Journal of Computational Linguistics*, 12:1–8.

Coates, A., Huval, B., Wang, T., Wu, D., Ng, A., and Catanzaro, B. (2013). Deep learning with cots hpc systems. *30th International Conference on Machine Learning, ICML 2013*, pages 2374–2382.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Duwairi, R. and Al-zubaidi, R. (2011). A hierarchical k-nn classifier for textual data. *The International Arab Journal of Information Technology*, pages 251–259.

Joshi, R., Karnavat, R., Jirapure, K., and Joshi, R. (2021). Evaluation of deep learning models for hostility detection in hindi text. *CoRR*, abs/2101.04144.

Kumar, S., Kar, A. K., and Ilavarasan, P. V. (2021). Applications of text mining in services management: A systematic literature review. *International Journal of Information Management Data Insights*, 1(1):100008.

Malema, G. A., Motlhanka, M., Okgetheng, B., and Motlogelwa, N. (2018). Setswana noun analyzer and generator. *International Journal of Computational Linguistics*, 9:32–40.

Malema, G. A., Motlogelwa, N., Okgetheng, B., and Mogotlhwane, O. (2016). Setswana verb analyzer and generator. *International Journal of Computational Linguistics*, 7:1–11.

Mercelis, W. (2021). Developing ai tools for a writing assistant: Automatic detection of dt-mistakes in dutch. *International Journal of Computational Linguistics*, 12:9–23.

Meshesha, M. and Solomon, Y. (2018). English-afaanoromo statistical machine translation. *International Journal of Computational Linguistics*, 9:26–31.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Mironczuk, M. M. and Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54.

Otter, D., Medina, J., and Kalita, J. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21.

Patil, N. N. and Patil, J. B. (2016). Robust text watermarking technique for authorship protection of hindi language documents. *International Journal of Computational Linguistics*, 7:12–22.

Pinheiro, R. H., Cavalcanti, G. D., and Tsang, I. R. (2017). Combining dissimilarity spaces for text categorization. *Information Sciences*, 406-407:87–101.

Raina, R., Madhavan, A., and Ng, A. (2009). Large-scale deep unsupervised learning using graphics processors. volume 382, page 110.

Sumamo, J. S. and Teferra, S. (2018). Designing a rule based stemming algorithm for kambaata language text. *International Journal of Computational Linguistics*, 9:41–54.

Statista (2022). Natural language processing market revenue worldwide. https://www.statista.com/statistics/607891/worldwide-natural-language-processing-market-revenues. Accessed 03-07-2022.

Wang, D., Wu, J., Zhang, H., Xu, K., and Lin, M. (2013). Towards enhancing centroid classifier for text classification—a border-instance approach. *Neurocomputing*, 101:299– 308.

Xin, Y. and Zhang, Z. (2020). An improved chinese text multi-label classification method based on CNN. *Journal of Physics: Conference Series*, 1619(1):012017.

Zhu, H. and Lei, L. (2022). The research trends of text classification studies (2000–2020): A bibliometric analysis. *SAGE Open*, 12(2).