Who said that? Comparing performance of TF-IDF and fastText to identify authorship of short sentences

Thomas van Tussenbroek[1] , **Tom Viering**[1] , **Stavros Makrodimitris**[1] , **Arman Naseri Jahfari**[1] , **David Tax**[1] , **Marco Loog**[1]

[1]TU Delft

Abstract

Authorship identification is often applied to large documents, but less so to short, everyday sentences. The ability of identifying who said a short line could provide help to chatbots or personal assistants. This research compares performance of TF-IDF and fastText when identifying authorship of short sentences, by applying these feature extraction techniques to the television series *Friends*' transcripts. TF-IDF outperforms fastText in every measurement, but its performance is only marginally better than randomly guessing the original character, reaching an accuracy of 28 percent when making a distinction between 6 characters. Accuracy increases linearly at the same rate for both techniques as the minimum word count per sentence set on the test data increases. TF-IDF's confidence remains constant as this limit is set on either the test or training data, whereas fastText's confidence decreases and increases, respectively. Cross-entropy loss, however, remains constant for fastText and decreases for TF-IDF as the minimum word count set on the test data increases.

1 Introduction

Authorship identification, also known as authorship attribution, has been of great influence in combating phishing (Khonji et al., 2011), fighting unrightful authorship claims (Mosteller & Wallace, 1963), and helping to solve forensic cases (Khan et al., 2012). To identify authorship of a document, features are extracted from the unidentified text. They are matched to features which have been extracted from documents the machine already knows the author of. Many researchers achieved high accuracy rates when identifying authorship of large text documents such as books or e-mails, but little research has been conducted on identifying authorship of everyday, short sentences (Van der Knaap & Grootjen, 2007). Identification of such short, everyday sentences, however, could be beneficial in many fields. An example of this would be online chatbots, which could provide personalized responses based on very brief conversations. Alternatively, this would benefit personal assistants in a family home such as smart speakers, which could personalize actions depending on which family member made a request.

The difficulty with identifying authorship of short sentences is that they lack features in which identification models normally try to find a pattern. This is because everyday speech introduces abbreviations, reduced grammar, and a lack of word count per sentence (Van der Knaap & Grootjen, 2007). Two commonly used feature extraction techniques are TF-IDF and fastText, a type of word embedder (Waykole & Thakare, 2018). The first assigns each word with a score based on how often it occurs with respect to all words in a document, the second represents words with similar meanings and context the same way by matching them to a prelearned dictionary. Feature vectors extracted by these techniques can be matched to one another if the sentences contain a variety of word choices or subjects. This is something which is lacking in everyday speech. It is therefore important to know which technique performs best.

This research paper compares the performance of two feature extraction techniques, TF-IDF and fastText, used to identify authorship of everyday, short sentences. In order to answer which technique is best suited for short sentences, this paper explores which mistakes are made by the model, and whether setting a minimum sentence length has any effect on performance. The two extraction techniques are applied to the American sitcom television series Friends' transcripts, which contains a large amount of everyday conversations, albeit theatrical. Moreover, the dataset has a limited pool of characters to train a classification model on, which allows for better evaluation of results.

The remainder of this paper is structured as follows. In section 2, we explain how TF-IDF and fastText extract features from text, and describe their possible advantages and disadvantages. Section 3 discusses which metrics were chosen to compare the classification of the produced feature vectors. Next, section 4 describes the experimental setup, the ethical and results of which are shared in section 9. Next, these results are discussed in section 6. Section 7 provides a conclusion, and possible future research is recommended in section 8. Lastly, section 9 addresses ethics and reproducibility.

2 Feature extraction

In order to be able to identify who said a particular line, features should be extracted from the text to create a machine readable representation of the sentence. Feature extraction can be either hard-crafted or learned. Hand-crafted feature extraction represents words by how important they are with respect to other words in the corpus. Learned feature extraction matches words in text to pre-trained dictionaries by looking at a word's context. A classification model therefore predicts people based on either how often they say a word, or which subjects they talk about, respectively.

For hand-crafted feature extraction, this research uses term frequency–inverse document frequency (TF-IDF), one of the most popular hand-crafted feature extraction techniques (Beel et al., 2016; Waykole & Thakare, 2018). This technique shall be compared to fastText, which provides learned feature extraction by means of embedding words (Bojanowski et al., 2017), which is also frequently used in natural language processing (Waykole & Thakare, 2018).

2.1 TF-IDF

TF-IDF consists of two parts: term frequency (TF) and inverse document frequency (IDF) (Rajaraman & Ullman, 2011). TF is computed by taking the frequency f_{ij} of a word i in sentence j, divided by the maximum frequency of all words k in the same sentence, the formula of which is the following:

$$TF_{ij} = \frac{f_{ij}}{max_k f_{kj}}. \tag{1}$$

IDF is computed by taking the logarithm of the total amount of sentences N divided by the amount of sentences n_i containing word i. This results in the following formula:

$$IDF_i = \log \frac{N}{n_i}. \tag{2}$$

The score of word i in sentence j is then computed by:

$$\text{Score} = TF_{ij} \times IDF_i. \tag{3}$$

This scoring is applied to each sentence j in the database. The columns of the resulting vector represent all unique words in all sentences, where each row represents one sentence. A cell contains the TF-IDF scoring if the a column's corresponding word is part of the respective sentence. The resulting matrix is sparse, since for each row, there will only be as many cells with a value as there are unique words in the corresponding sentence; all other cells will contain the value zero. The total amount of words could be in order of hundreds of thousands, whereas each sentence might only contain ten words. Words that appear often across all sentences, such as stopwords, will weigh less than words that occur infrequently.

Since this technique relies on words that are saved in its dictionary, a classification model could have problems finding a pattern in the dataset. This could be particularly problematic if the model tries to classify a sentence which contains none of the words present in its dictionary.

On the other hand, this extraction technique could prove to be successful if the model tries to classify sentences which contain many words already present in the dictionary.

2.2 fastText

The learned feature extraction technique used in this research, fastText, is a type of word embedder. fastText is an adaptation of the well-known word-embedder word2vec (Goldberg & Levy, 2014). Word2vec was initialy also evaluated in this research, however, since fastText's results were comparable to or slightly better than word2vec's, only fastText's results were included. Word embedders represent words as a probability distribution, where words with similar context have a similar distribution. This probability distribution is learned from large training datasets, in which a word embedding model analyses which words often appear near one another and which ones do not (McCormick, 2016). The probability distribution of ”apple” and ”banana” would therefore be rather similar, whereas their representation would differ greatly from that of ”castle”.

Besides analysing each word, fastText also captures a word's n-grams in its probability distribution. N-grams are a word's partial substrings of size n. If n were set to 3, the n-grams for ”apple” would become ”app”, ”ppl”, ”ple” (for demonstration purposes, fastText's use of boundary symbols < and > is ommitted in this example). Taking into account a word's n-grams for embedding allows for capturing the meaning of pre- or suffixes (Subedi, 2018). As an example, n-grams could help create similar probability distributions when extracting features from ”prehistory” and ”history”.

A classification model using fastText's representation should be better than TF-IDF's at finding patterns in sentences which contain words not present in the training data, since it is able to match context.

If the context of two sentences is the same, fastText would assign them with similar representations, even if the choice of words is different. Using different words can be an indication of such sentences being said by different people, and cannot be recognized, which could be a disadvantage of using fastText.

3 Measuring performance

In order to identify authorship of sentences, a classification model needs to be trained, for which logistic regression is used. Logistic regression is chosen because it is computationally inexpensive. K-nearest neighbor was also evaluated as a classification model, but was not used in this research due to its relatively worse performance and longer computation time. We will be using cross-validation for hyper parameter tuning and the data will be split into a train and test set to be able to register performance of unseen data.

To decide on which feature extraction technique performs best, there are two subquestions that need to be answered. The subquestions are listed hereafter, together with the metrics that were used to find the answer:

- **Which characters are often mistaken for one another** can be answered by examining the confusion matrix.
- **How does performance change when there is a constraint on the minimum sentence length** can be answered by examining the model's accuracy, confidence in predictions, and cross-entropy loss.

Finding out which characters are often mistaken for one another

In order to show which mistakes are made by the classification model, this research used a confusion matrix. A confusion matrix is a table which shows how many lines have been correctly classified to belong to a certain character and how many have not. Each column represents the actual character, and each row represents the character predicted by the model. The main diagonal displays correct predictions (true positives), and all other cells count how many classifications were incorrectly classified (false positives). It is therefore most favourable to have high values in the main diagonal.

The confusion matrix shows which characters are often mistaken for one another. Therefore, it can provide insight on whether the model thinks characters have a similar speech pattern. By doing so, it can be determined whether there are common speech patterns amongst the characters.

The effect of a minimum sentence length on performance

In order to show whether setting a minimum sentence length changes performance, three metrics are evaluated: accuracy, the model's confidence in its predictions, and cross-entropy loss.

For each of these metrics, performance is measured by specifying a range of minimum sentence lengths. This minimum sentence length is applied to the train and test set separately. Therefore, when evaluating whether performance changes when a minimum word count is set on the *train* data, the *test* data does not change, and vice versa.

- **Accuracy** is the percentage of correct predictions made by the classification model. It is a necessary measurement as the model must choose one character who might have said the line, which is either correct or incorrect. For this research, it is also the most important performance indicator, since we want the model to make as many correct predictions as possible.
- **Confidence in prediction** is the probability assigned to the predicted character. For every sentence, the logistic regression model creates a probability distribution, assigning a probability to each of the six characters. The higher the probability, the more likely the model thinks the sentence belongs to that given character. The confidence in the prediction is the probability assigned to the character the model thinks most likely said the line (the highest probability).
- **Cross-entropy loss**, also log loss, shows how close the probability distribution made by the classification model is to picking the true character (Nbro, 2019). The logistic regression model used in this research minimizes the cross-entropy loss in order to achieve the highest result, and is therefore a good indication of performance. The cross-entropy loss is decided by the following formula:

$$\text{Cross-entropy loss} = \sum_i p(i) \log q(i). \tag{4}$$

  Where $p(i)$ and $q(i)$ are the wanted and actual probability of sentence i, respectively. As follows from the formula, a lower cross entropy loss means a better performance.

With each increase of the minimum word count, the data size decreases. Therefore, it could be that a change in performance is due to this change of sample size rather than due to us setting a minimum word count. In order to rule out that this change of performance is due to the change of data size, additional metrics shall be created for accuracy, confidence in predictions, and cross-entropy loss. For each minimum word count we recorded the data size, and we randomly selected a sample with a size equal to what was recorded.

If the performance of randomly selecting samples is equal to the performance of setting minimum word counts, we know that the performance change is due to a decrease of the data size, and not due to us setting a minimum word count. The opposite holds when the performance is different from one another.

In order to give an indication of the reliability of the experiment, we have calculated the variance of the accuracy per minimum word count. The variance was calculated by means of the following formula:

$$\text{Var}[a_i] = \frac{a_i(1-a_i)}{n_i}, \tag{5}$$

where a_i is accuracy and n_i is data size when setting minimum word count i (heropup, 2014).

4 Experimental setup

In order to determine whether TF-IDF or fastText performs best when identifying authorship of short, everyday sentences, we use an experimental setup consisting of the following five steps:

1. Retrieving data
2. Parsing and pre-processing data
3. Extracting features
4. Classifying features
5. Evaluating metrics

These steps are described in more detail in this chapter. When using algorithms provided by libraries, default parameters were used unless stated otherwise. The full list of libraries and a description of what they were used for, as well as the hardware and software used, can be found appendix A.

4.1 Retrieved data

The television series Friends consists of 236 episodes, the transcripts of which were obtained from the internet-forum *Forever Dreaming* (Forever Dreaming Transcripts, 2018) in HTML format.

4.2 Parsed and pre-processed data

Lines said by multiple characters were parsed per individual character. The transcripts were parsed in the following order:

1. **Scene directions were removed.**

2. **Contractions were expanded.** Words containing an apostrophe that were not expanded by the used package are displayed in table 1. Since all punctuation was removed in a later step, these words have been expanded as well.

Table 1: Words that were substituted to not include apostrophes

Original	Substitute
’ (U+2019)	' (U+0027)
c’mon	come on
o’clock	o clock
y’know	you know
‘em	them

3. **Lines were split on a full stop (.), question mark (?), or explanation mark (!).** This paper researches identification of authorship of individual lines, which is the reason why multiple sentences in one line need to be divided and treated separately. Some words which naturally contain these punctuation marks have been altered such that they are not needlessly split into separate lines. These are displayed in table 2.

Table 2: Words that were substituted to not include a full stop

Original	Substitute
c.h.e.e.s.e.	cheese
t.g.i.friday’s	tgifridays
dr.	doctor
f.y.i.	fyi
p.m.	pm
a.m.	am
s.a.t.s.	sats

4. **Punctuation was removed.** One word which was substituted before removal of all punctuation was the transcripts use of *s*x*, instead of *sex*. This way, this word could be processed properly.
5. **Lines consisting of one word only were removed.** Lines with one word contain too little information to count as a proper sentence.
6. **Lines said multiple times by the same person were reduced to occur only once.** Learning the pattern of one sentence multiple times would create an unbalanced classification model, and therefore should only be evaluated once.
7. **Lines said by side-characters were removed.** There are many side characters in the data set, all of which say few lines relative to the main characters. This could lead to class imbalance problems, meaning that there is an unequal distribution of classes. Therefore, lines said by non-main characters are removed all-together.
8. **Lines said by multiple people were removed.** This research focuses on finding patterns in speech unique to one person, and therefore, if the exact same sentence is said by multiple people, it should not be evaluated.

After parsing, the dataset consisted of 57.393 unique lines. Pre-processing options that have been evaluated but have not been used in this research are spell correction, stemming, and the removal of stopwords. The use of these techniques on their own and their combinations did not improve the accuracy of the model, and in some cases even worsened it.

4.3 Extracted features

Next, features were extracted from the parsed lines by means of fastText and TF-IDF, creating a vector representation for each sentence.

TF-IDF

Applying dimensionality reduction to reduce the width per vector to 300 to match fastText’s output vector size by using PCA was considered. This resulted in lower accuracy and longer computation time, and was therefore disregarded.

fastText

Naturally, fastText is a word embedder. Since this research classifies sentences rather than words, we applied fastText to each word in a sentence and represented the sentence as a mean of the embedded words (Takeshita, 2019). Words were matched to the default fastText dataset, trained on Common Crawl and Wikipedia, with character n-grams of length 5. The output per sentence is a vector of 300 features (Bojanowski et al., 2017).

Note that TF-IDF’s vector representation’s size depends on the amount of unique words in the training data, whereas fastText’s vector representation is always 300. Therefore, if the training data changes, TF-IDF’s features need to be extracted again, which is not necessary for fastText.

4.4 Classified features

The datasets were split into a 80:20 train and test set with the random state `1515` to allow reproducibility of the research. The classification model was trained with the training set, and all evaluations were done with the test set.

For parameter tuning, we used grid search. Cross-validation was performed in 5 fold.

Parameters that were evaluated using grid search were the inverse of reguralization `C`, with values `[1e(-10), 1e-9, ..., 1e(9), 1e(10)]` to combat overfitting, and the maximal amount of iterations `max_iter`, with arbitrary values `[250,500,750,1000]` to combat the inability to converge.

4.5 Evaluated metrics

The minimum word count is a range from 2 up until 30. The reason for the starting value of 2 is that sentences with only one word were removed during pre-processing of data. The latter value of 30 is an arbitrarily chosen large number. From a minimum of 30 words, the train and test set contain less than 323 and 68 lines respectively, which is less than 1 percent of the amount of starting data, which is 43728 and 10955 respectively. The dataset therefore becomes too unreliable to draw conclusions from.

The logistic regression model required retraining every time the minimum word count of the *training* dataset was changed. This is necessary because the classification model's internal weights, and therefore its predictions, changes depending on the training input. Following the same logic, the classification model does not need retraining when the *test* dataset changes minimum word count. TF-IDF's features needed to be extraction again every time the minimum word count of the *training* dataset was changed as well, as explained in subsection 4.3.

5 Results

Table 5 and 6 in appendix B show the optimal parameters found by applying grid search to the logistic regression model without and with minimum word count set on the training data respectively. Appendix C shows the table with the amount of data per minimum word count. This was used for the metrics shown in appendix F, showing performance when randomly selecting data equal to the size of setting a minimum word count to either the test or training data. The following subsections give a visualisation and explanation of the results.

Confusion matrix

Figure 1 and 2 show the confusion matrix (CM) of the logistic model applied to the data created by fastText and TF-IDF respectively.

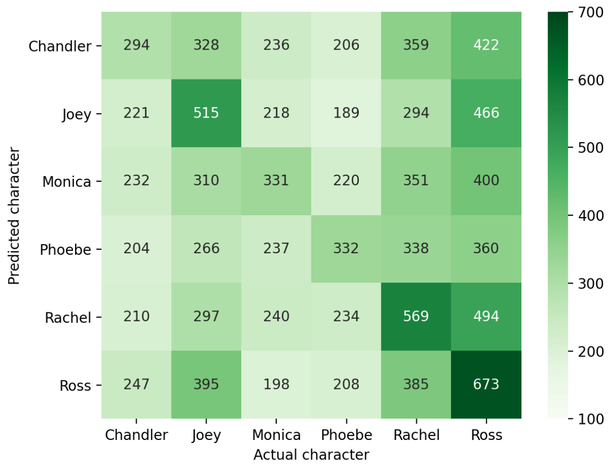

Figure 1: Confusion matrix when extracting features using fastText

Figure 1 shows the confusion matrix when classifying fastText's data. From this figure, we see that only three of the six cells with the highest values are in the main diagonal. Additionally, more lines said by Monica and Phoebe are predicted to be said by Rachel and Ross than by Monica and Phoebe themselves. It is also notable that lines said by Joey are often confused to have been said by Ross.

Figure 2 shows TF-IDF's data's confusion matrix. The six cells with the highest values are in the main diagonal, which indicate the true positives, and therefore shows better performance than fastText. Lines said by Joey, Rachel, and Ross are

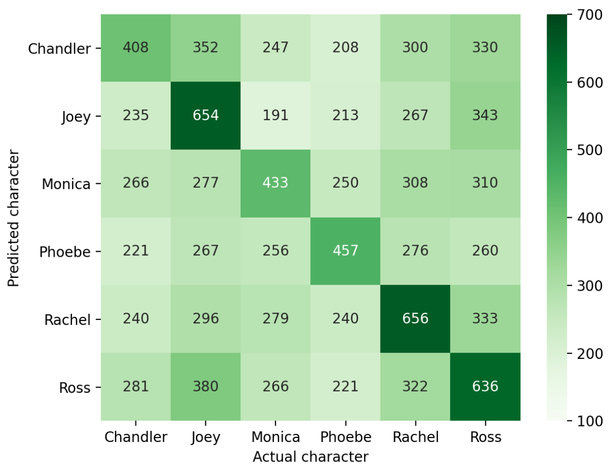

Figure 2: Confusion matrix when extracting features using TF-IDF

far more often correctly predicted than lines said by Chandler, Monica, and Rachel.

From these two figures we can deduce that appplying the classification model to TF-IDF's data results in less confusion between characters than when it is applied to fastText.

Accuracy

The average accuracy of the model per minimum word count set on test and train dataset are shown in figure 3 and 7 respectively, the latter of which can be found in appendix E. Appendix F shows the accuracy when randomly selecting data in figures 9 and 10. The variance of the accuracy has been included in appendix D. The largest variance is $2.40e-03$.

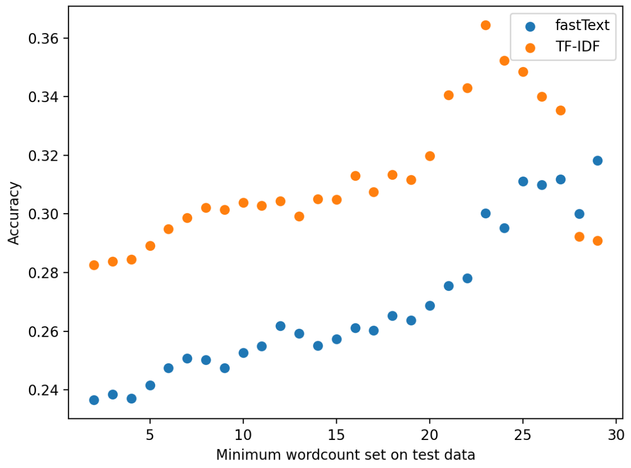

Figure 3: Average accuracy per minimum word count set on test dataset

From figure 3, which shows the accuracy when the minimum word count is set on the *test* data, we can see that TF-IDF's accuracy is continuously 5 percentage points more accurate than fastText's, up untill a minimum word count of 25. Both TF-IDF's and fastText's accuracy rises almost linearly up untill that point, but after a minimum word count of 25, TF-IDF's accuracy drops sharply, even below fastText's.

The linear increase is not apparent in figure 9, so this increase in accuracy is due to the increase of minimum wordcount. The drop and spike of TF-IDF's and fastText's performance after a minimum wordcount of 25 however, is visible in figure 9, and so this is due to the change of dataset size.

From this figure, it is clear that using TF-IDF's data results in higher accuracy than when fastText was used to extract features. Moreover, increasing the minimum word count of the test set does increase performance.

Figure 7, in which the minimum word count is set on the *training* data, and 10, where data is randomly selected, follow a similar trend to one another. This change in performance is therefore attributed to the change of data size, and not due to setting a minimum word count. Consequently, this result has not been evaluated more thoroughly and can be found in appendix E.

Confidence in prediction

Figures 4 and 5 show the confidence of the two techniques that their predictions are the correct ones. Appendix F shows the confidence when randomly selecting data in figures 11 and 12.

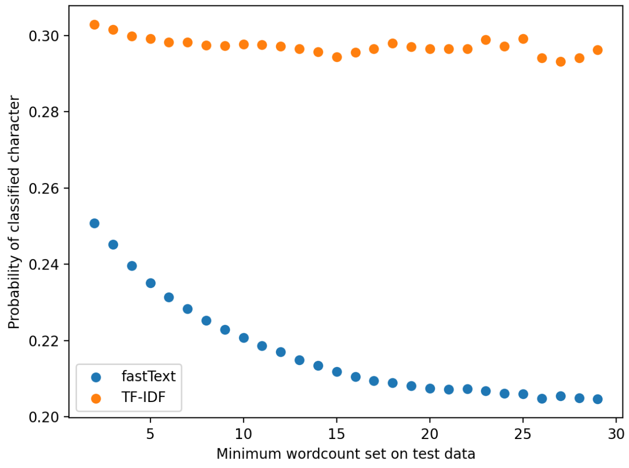

Figure 4: Average confidence in predicted character per minimum word count set on test dataset

Figure 4 displays the confidence when the limit of word count is set on the *test* data. From this illustration, we can see that TF-IDF's confidence starts higher than fastText's and remains at a constant 30%. The performance when randomly selecting data, as can be seen in figure 11, is similar, and so setting a minimum word count on the test set has no influence on TF-IDF.

fastText's confidence starts at 25%, but decreases over time. Its confidence decreases less sharply as the minimum word count increases, however.

Figure 5 shows the model's confidence when the limit of word count is set on the *training* data. TF-IDF's has a higher confidence than fastText when the minimum word count is set

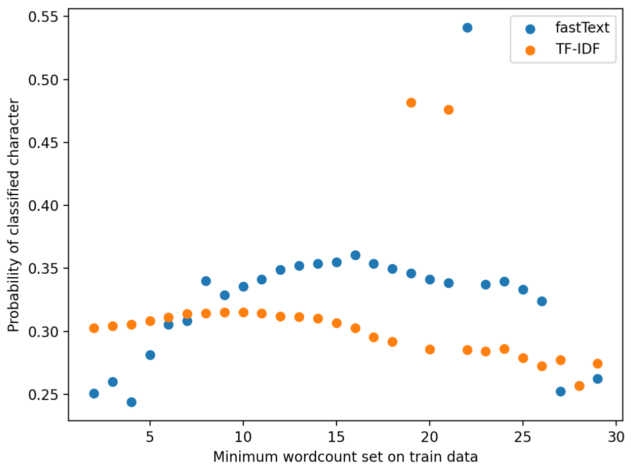

Figure 5: Average confidence in predicted character per minimum word count set on train dataset

to two. However, fastText's confidence rises faster than TF-IDF's, and is more confident than TF-IDF if the minimum word count is larger than eight. TF-IDF's confidence only slightly increases, then decreases as the minimum word count increases. After a minimum word count of 15, both TF-iDF's and fastText's confidence declines at the same pace. There are two outliers for TF-IDF and one for fastText around a minimum word count of 20. This figure is not similar to figure 12, an so we can attribute this change of performance to having set a minimum word count to the train data.

From these plots, we can say that TF-IDF's confidence stays almost constant when increasing the word count of the training dataset. Combining these figures with the metrics measuring accuracy, we can see that fastText, becomes less confident of its choices the more accurate it becomes, and becomes more confident the less accurate it becomes.

Cross-entropy loss

The average cross-entropy loss per minimum word count set on the test and the traning dataset are displayed in figure 6 and 8 respectively, the latter of which can be found in appendix E. Appendix F shows the cross-entropy loss when randomly selecting data in figures 13 and 14.

Figure 6 shows the cross-entropy loss when a minimum word count is set on the *test* data. It is apparent that TF-IDF's loss is lower than fastText's regardless of the minimum word count. TF-IDF's loss drops almost linearly as the test set's minimum word count increases. After a slight initial drop, fastText's loss remains almost constant.

From this measurement, we can say that TF-IDF's cross-entropy loss is more favourable that fastText's. When the minimum word count of the test data increases, TF-IDF's loss decreases, whereas fastTexts remains almost constant.

Aside from the outliers, the cross-entropy loss when setting a minimum wordcount on the *train* data, and when randomly selecting data, as can be seen in figure 8 and 14, follow a trend similar to one another. This means that the change in performance is due to a decrease of training data, and not due

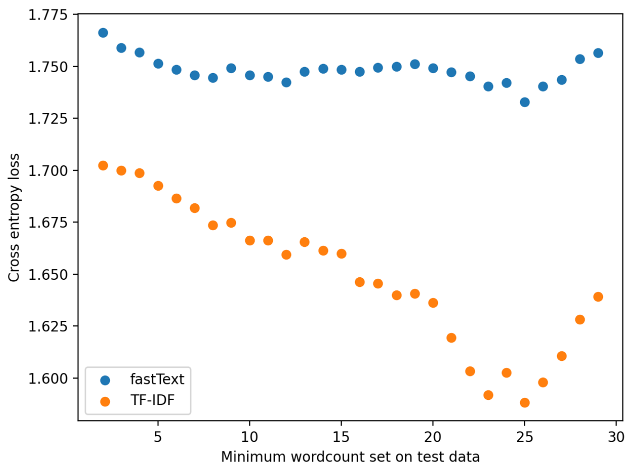

Figure 6: Average cross-entropy loss per minimum word count set on test dataset

to an increase of the minimum word count. This metric has therefore not been included in the results, but can be found in appendix E instead.

Data example

For discussion purposes, we also gathered some information on the classified data. Table 3 shows the amount of lines per character mentioning "Chandler", together with how often these lines are predicted to have been said by the characters and how many of those predictions are correct.

Table 3: Amount of times characters say a line containing the name "Chandler", along with how often the prediction models think these lines containing "Chandler" are said by the characters and how many of these predictions are correct

Character	Actual	fastText		TF-IDF	
		Pred.	Corr.	Pred.	Corr.
Monica	48	62	18	64	22
Joey	30	47	11	53	17
Ross	29	7	1	6	2
Rachel	26	20	6	20	6
Phoebe	23	21	4	14	4
Chandler	8	7	0	7	2

6 Discussion

This research answers the question which feature extraction technique, TF-IDF or fastText, performs best on authorship identification of short, everyday sentences from the television series Friends. This is done by looking at the performance of a classification model applied to feature vectors created by these techniques. A limit was put on the amount of words per line to look at how performance changes based on the input length, and the techniques have also been compared on which characters are often confused for one another. TF-IDF confuses people less often for one another than fastText does, and in all three metrics which compare performance when a minimum word count is set on either the test or training data, TF-IDF also clearly performs better than fastText. The cause of the deviations in figures 5 and 8 were not found. We can say that for the short sentences found in the dataset, it is apparently better to make a prediction based on how often words occur, rather than on how frequently a subject is being discussed.

What strikes from the results is how low the overall accuracy of the model is. Even though this model achieves better results than if the character was randomly guessed, we can only say that the model achieves 28 percent accuracy when TF-IDF's feature representation is used and no minimum word count is set. Looking at the dataset, however, we may be able to speculate why the model's performance is so limited. Many sentences which the model has to predict authorship of are very common, such as "Oh yeah sorry about that", "OK I want to", or "Oh can I come?". These lines could have been said by any of the characters as they contain such little personalized information.

Only if sentences contains words that strongly indicate that they belong to someone, is the model capable of correctly predicting the original author. As an example, the sentence "Chandler's my husband" is correctly predicted by both techniques to belong to Monica, because the model knows that Monica's character often speaks about Chandler. However, when other characters use Chandler's name in a sentence, the model also often wrongly predicts that the sentence belongs to Monica, and example of which is the sentence "I knew I should have married Chandler", which is said by Phoebe instead of by Monica. As can see in table 3, Monica mentions Chandler's name most, and is predicted to be mentioning Chandler more often than she actually says by both models. Moreover, even though Ross mentions Chandler's name almost as often as Joey, who is predicted second most by both model's, Ross is almost never chosen. Therefore, even though the model does use the knowledge it has gained through training to base its predictions on, the model is not able to find a pattern, and the predictions are therefore only slightly better than guessing.

This could also explain why it is clear that an increase of the minimum word count of the to be predicted lines is correlated to an increase of accuracy. The increase of feature complexity or variety in which the classification model can find a pattern could aid the model into making better predictions.

Following the way cross-entropy loss is calculated, when confidence in a prediction remains constant as accuracy increases, the cross-entropy loss should decline. Alternatively, if accuracy increases and confidence decreases, the cross-entropy loss should remain constant. This can also be seen in the results. When TF-IDF's accuracy rises and its confidence in predictions remains constant as the minimum word count of the *test* set increases, it's cross-entropy loss decreases. At the same time, fastText's confidence in its predictions decreases, but its cross-entropy loss remains constant.

With this research, we can say that feature extraction by TF-IDF results in better authorship identification of everyday, short sentences than if fastText were used. We have also shown that an increase of length of the to be predicted lines results in better performance of both techniques.

7 Conclusion

In this study, we compared the performance of TF-IDF and fastText when applied to authorship identification of short, everyday sentences. After examining accuracy, confidence, and cross-entropy loss of the classification model, we can conclude that TF-IDF performs better than fastText in all categories. Moreover, we showed that features created by TF-IDF makes the classification model less confused about who said a line than fastText's feature representations. Where previous papers focused mostly on identifying authorship of large text documents, we have shown that identifying authorship of everyday, short sentences using TF-IDF's and fastText's feature representations perform better than randomly guessing the character, albeit not with a significantly improved performance. Performance does increase as the sentence length of the to be identified line increases.

TF-IDF's and fastText's accuracy increases linearly as the minimum word count set on the test set changes. Accuracy decreases for both if a minimum word count is set, but this is because the train set contains fewer data, and not due to the word count limit. TF-IDF's confidence in its prediction remains constant with an increase of minimum word count, whereas fastText's confidence decreases and increases when the minimum word count increases on the test and training set, respectively. The cross-entropy loss remains constant for fastText as the minimum word count increases, whereas it declines for TF-IDF, up until a minimum of 24 words.

8 Future research

Future research could focus on distinguishing characters in an associative manner, rather than the discriminative manner as done in this research. Characters might adapt different language treats depending on who they talk to, similar to how someone might speak with different words depending on whether they talk with a toddler or their boss. If a pattern could be found in someone's speech depending who they are speaking with, this might help achieve better performance when identifying authorship of everyday, short sentences. This research has already shown that there is potential in distinguishing characters in an associative manner, as lines mentioning a character's name can give information on who said it.

Since the classification model used in this research was not able to find a pattern in feature vectors, the speech patterns of the characters might be too similar. Because of this, a multi-label classification model might be more suitable for future research, rather than the multinomial model used in this research. A multinomial classification model chooses the character it has most confidence in, whereas a multi-label classification model is able to choose multiple characters based on whether the confidence in characters reaches above a chosen threshold. Such a model will not provide a singular answer to who said a line, but the chance of the correct character occurring amongst the predictions will be higher. Multi-label authorship identification has already been applied to large, collaborative documents (Boumber et al., 2018; Dauber et al., 2017). Experiments applying such multi-label authorship identification could opt to include duplicate lines or lines said by multiple characters instead of removing them from the dataset as done in this research.

9 Responsible research

This paper was written for the CSE3000 course provided by the Technical University Delft. This experiment was performed without funding and there was no conflict of personal interest. The data retrieved came from an external, independent internet forum. No data has been purposely removed unless a justified explanation was provided, and the data has only been handled objectively. Moreover, outliers have not been omitted from the results.

All libraries and packages used are listed in this paper in appendix A, to allow for reproducibility. The code used for the research can be found on the public GitHub repository `https://github.com/thomasvant/Character-classification`. Since probability did play a role in the experimental setup, some results might come out slightly different when the experiment is repeated. However, the results should not differ significantly, as can be seen in the variance of the accuracy in appendix D. Moreover, this deviation has been limited as much as possible by stating the parameters used. The general conclusions drawn from results should therefore not be any different when the research is repeated.

References

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338. doi: 10.1007/s00799-015-0156-0

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Boumber, D., Zhang, Y., & Mukherjee, A. (2018). Experiments with Convolutional Neural Networks for Multi-Label Authorship Attribution. In *Proceedings of the eleventh international conference on language resources and evaluation* (pp. 2576–2581).

Dauber, E., Overdorf, R., & Greenstadt, R. (2017). Stylometric authorship attribution of collaborative documents. In *International conference on cyber security cryptography and machine learning* (pp. 115–135).

Forever Dreaming Transcripts. (2018, February). *FRIENDS Transcripts Index - Forever Dreaming.* Retrieved April 30, 2020, from https://transcripts.foreverdreaming.org/viewforum.php?f=845

Goldberg, Y., & Levy, O. (2014, February). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method.

heropup. (2014, February). *Variance of average of bernoulli variables.* Cross Validated. Retrieved from https://stats.stackexchange.com/q/86425

Khan, S., Nirkhi, S., & Dharaskar, R. (2012, April). Author Identification for E-mail Forensic. *Proceedings of National Conference On Recent Trends In Computing NCRTC*, 29–32.

Khonji, M., Iraqi, Y., & Jones, A. (2011). Mitigation of spear phishing attacks: A Content-based Authorship Identification framework. In *2011 International Conference for Internet Technology and Secured Transactions* (pp. 416–421).

McCormick, C. (2016, April). *Word2Vec Tutorial - The Skip-Gram Model.* Retrieved May 17, 2020, from http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Mosteller, F., & Wallace, D. (1963). Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *Journal of the American Statistical Association*, *58*(302), 275–309.

Nbro. (2019, March). *What is cross-entropy?* Retrieved May 27, 2020, from https://stackoverflow.com/questions/41990250/what-is-cross-entropy

Rajaraman, A., & Ullman, J. (2011). *Mining of Massive Datasets*. USA: Cambridge University Press.

Subedi, N. (2018, July). *FastText: Under the Hood - Towards Data Science.* Retrieved May 17, 2020, from https://towardsdatascience.com/fasttext-under-the-hood-11efc57b2b3

Takeshita, S. (2019, December). *Super Easy Way to Get Sentence Embedding using fastText in Python.* Retrieved May 25, 2020, from https://towardsdatascience.com/super-easy-way-to-get-sentence-embedding-using-fasttext-in-python-a70f34ac5b7c

Van der Knaap, L., & Grootjen, F. (2007, January). Author Identification in Chatlogs using Formal Concept Analysis.

Waykole, R., & Thakare, A. (2018, April). A Review of Feature Extraction Methods for Text Classification. *International Journal of Advance Engineering and Research Development*, *5*(4), 351–354.

A Hardware and software used

The research was conducted on a Dell XPS 13 9350 with 8GB of RAM and an Intel Core i5-6200U CPU running 64-bit Windows 10 Home version 1903. The Python version used was 3.7.0.

Table 4: Libraries used to perform this research including version number and a description of what they were used for

Package	Version	Description
autocorrect	1.2.1	Autocorrection
BeautifulSoup	4.9.0	Parsing transcripts' HTML structure
matplotlib	3.2.1	Creating plots
NLTK	3.5	Removing stopwords and stemming
numpy	1.18.3	Numeric operations
pandas	1.0.3	Storing, retrieving information, obtaining random samples, and altering data
pathlib	1.0.1	File management
pycontractions	2.0.1	Expanding contracions
requests	2.23.0	Downloading transcripts
scikit-learn	0.22.2.post1	Creating TF-IDF feature extraction, splitting train and test data, classification using logistic regression, performing grid search, obtaining log loss and accuracy
seaborn	0.10.1	Creating confusion matrix
sister	0.1.7	Embedding using fastText

B Parameters used

Table 5: Result of performing grid search with no minimum word count set on training data

Min word count	fastText		TF-IDF	
	`C`	`max_iter`	`C`	`max_iter`
-	10.0	250	1.0	500

Table 6: Result of performing grid search with minimum word count set on training data

Min word count	fastText		TF-IDF	
	`C`	`max_iter`	`C`	`max_iter`
2	10.0	250	1.0	500
3	1.0	250	1.0	500
4	0.1	250	1.0	500
5	1.0	500	1.0	250
6	10.0	250	1.0	250
7	1.0	250	1.0	250
8	10.0	500	1.0	250
9	1.0	500	1.0	250
10	1.0	500	1.0	250
11	1.0	250	1.0	250
12	1.0	250	1.0	250
13	1.0	500	1.0	250
14	1.0	250	1.0	250
15	1.0	500	1.0	250
16	1.0	250	1.0	250
17	1.0	250	1.0	250
18	1.0	250	1.0	250
19	1.0	250	10.0	250
20	1.0	250	1.0	250
21	1.0	250	10.0	250
22	10.0	250	1.0	250
23	1.0	250	1.0	250
24	1.0	250	1.0	250
25	1.0	250	1.0	250
26	1.0	250	1.0	250
27	1e-05	250	1.0	250
28	1e-05	250	1e-05	250
29	1e-05	250	1.0	250

C Data size

Table 7: Data size when setting minimum wordcount

Min word count	Data size	
	Test set	Train set
2	10955	43728
3	10113	40453
4	8907	35629
5	7587	30348
6	6386	25439
7	5269	21143
8	4338	17500
9	3550	14556
10	2992	12103
11	2506	10075
12	2075	8407
13	1740	7002
14	1453	5856
15	1233	4939
16	1018	4128
17	852	3450
18	709	2927
19	588	2438
20	508	2049
21	417	1724
22	343	1445
23	298	1198
24	241	1019
25	200	837
26	170	681
27	130	572
28	110	476
29	86	384

D Variance

Table 8: Variance of accuracy when applying a minimum word count to the test and train set

Min word count	Var. test set		Var. train set	
	TF-IDF	fastText	TF-IDF	fastText
2	1.85e-05	1.65e-05	4.64e-06	4.13e-06
3	2.01e-05	1.80e-05	5.02e-06	4.46e-06
4	2.29e-05	2.03e-05	5.70e-06	5.05e-06
5	2.71e-05	2.42e-05	6.62e-06	5.96e-06
6	3.26e-05	2.92e-05	7.90e-06	7.06e-06
7	3.98e-05	3.57e-05	9.45e-06	8.50e-06
8	4.86e-05	4.33e-05	1.14e-05	1.04e-05
9	5.93e-05	5.25e-05	1.36e-05	1.24e-05
10	7.07e-05	6.31e-05	1.62e-05	1.49e-05
11	8.42e-05	7.58e-05	1.91e-05	1.78e-05
12	1.02e-04	9.31e-05	2.28e-05	2.13e-05
13	1.21e-04	1.10e-04	2.69e-05	2.50e-05
14	1.46e-04	1.31e-04	3.17e-05	3.03e-05
15	1.72e-04	1.55e-04	3.71e-05	3.58e-05
16	2.11e-04	1.90e-04	4.40e-05	4.25e-05
17	2.50e-04	2.26e-04	5.20e-05	5.03e-05
18	3.03e-04	2.75e-04	5.91e-05	5.92e-05
19	3.65e-04	3.30e-04	7.14e-05	7.18e-05
20	4.28e-04	3.87e-04	8.45e-05	8.40e-05
21	5.39e-04	4.79e-04	9.90e-05	9.80e-05
22	6.57e-04	5.85e-04	1.15e-04	1.13e-04
23	7.77e-04	7.05e-04	1.40e-04	1.37e-04
24	9.47e-04	8.63e-04	1.60e-04	1.60e-04
25	1.14e-03	1.07e-03	1.94e-04	1.93e-04
26	1.32e-03	1.26e-03	2.38e-04	2.34e-04
27	1.71e-03	1.65e-03	2.80e-04	2.56e-04
28	1.88e-03	1.91e-03	3.07e-04	3.07e-04
29	2.40e-03	2.52e-03	4.02e-04	3.81e-04

E Unused results

Accuracy

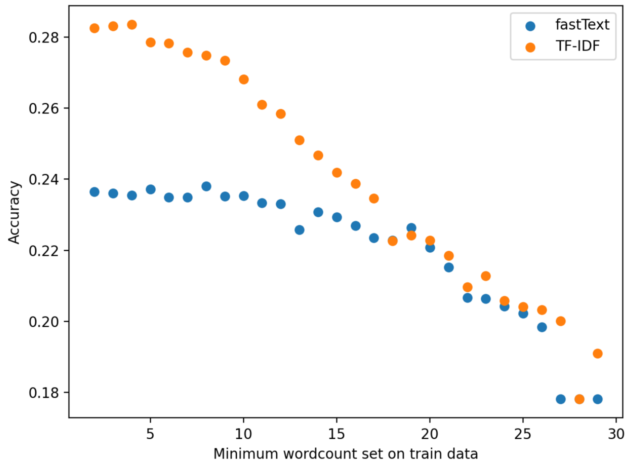

Figure 7: Average accuracy per minimum word count set on train dataset

In figure 7, in which the minimum word count is set on the *training* data, we can see that TF-IDF's accuracy starts higher than fastText's, but also declines faster. FastText's accuracy remains almost constant, until around a minimum word count of 20. After this minimum word count of 20, TF-IDF's and fastText's accuracy become equal and both continue declining at the same pace. This result is similar to figure 10, where random data is selected, and therefore, an increase of the minimum word count is not the cause of the change of performance.

Cross-entropy loss

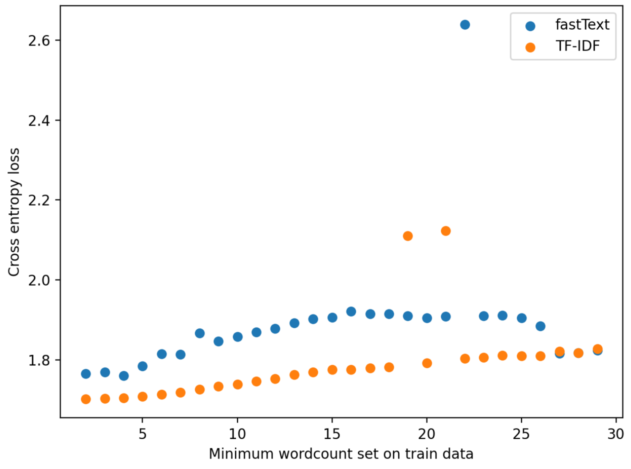

Figure 8: Average cross-entropy loss per minimum word count set on training dataset

The cross-entropy loss when setting a minimum word count on the *train* data can be seen in figure 8. FastText's loss starts higher than TF-IDF's. Both fastText's and TF-IDF's loss increase linearly initially, but fastText's loss drops slightly from 20 words and eventually has the same loss as TF-IDF around 25 words on. There is a large negative outlier in fastText's measurement, and two slight outliers in TF-IDF's around 20 words. This metric is similar to figure 14, and the change in performance is therefore due to a decrease of the training dataset.

F Metrics when randomly selecting data

Accuracy

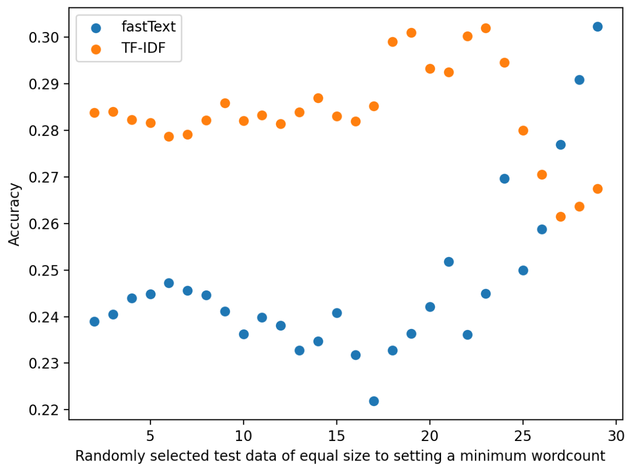

Figure 9: Average accuracy per random data of equal size to minimum word count set on test dataset

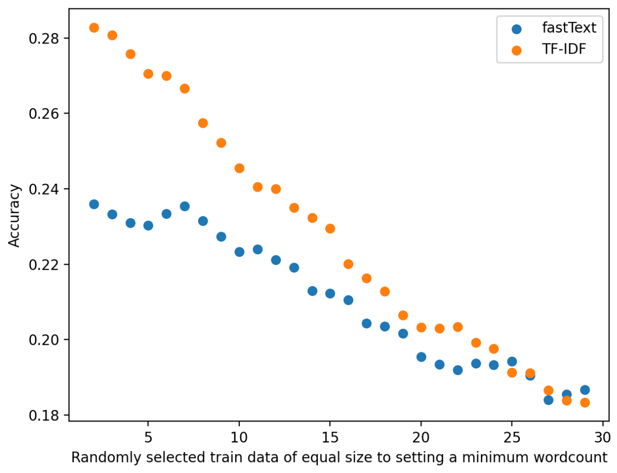

Figure 10: Average accuracy per random data of equal size to minimum word count set on train dataset

Confidence in prediction

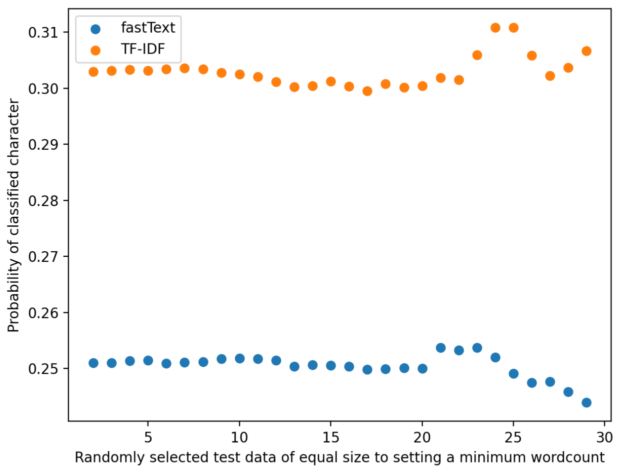

Figure 11: Confidence in prediction per random data of equal size to minimum word count set on test dataset

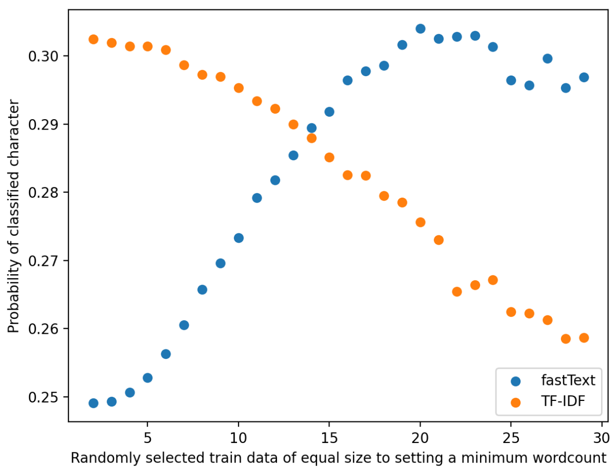

Figure 12: Confidence in prediction per random data of equal size to minimum word count set on train dataset

Cross-entropy loss

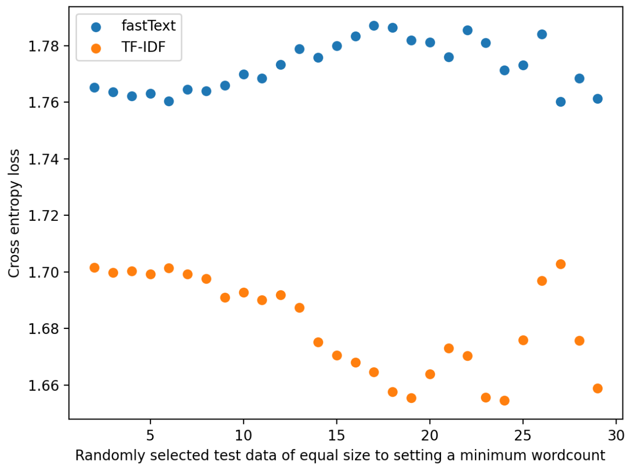

Figure 13: Cross entropy loss per random data of equal size to minimum word count set on test dataset

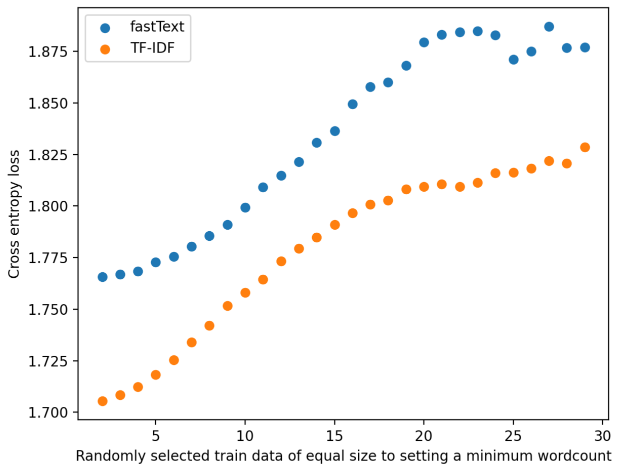

Figure 14: Cross entropy loss per random data of equal size to minimum word count set on train dataset