Poet Attribution for Urdu: Finding Optimal Configuration for Short Text

Tafseer Ahmed[1] Adil Rao[2]

Abstract

This study presents a machine learning system to identify the poet of a given poetic piece consisting of 2 lines (i.e. a couplet) or more. The task is more difficult than the general task of author attribution, as the number of words in verses and poems are usually less than the number of articles present in author attribution datasets. We applied classification algorithms with different sets of feature configurations to run several experiments and found that the system performs best when support vector machine using a combination of unigram and bigram are used. The best system (for 5 Urdu poets) has the accuracy of 88.7%.

Keywords: Poet Attribution, Author Attribution, Ngrams, Classification, Urdu.

1 Introduction

A poem is defined as "a piece of writing in which the words are chosen for their sound and the images they suggest, not just for their obvious meanings. The words are arranged in separate lines, usually with a repeated rhythm, and often the lines rhyme at the end"[1]. Poetry is a creative expression of words in which words most of the time do not reflect their true meaning.

Urdu is spoken majorly in Pakistan and India, and also in rest of the world [2]. It is an Indo Aryan language that has strong influence of Persian and Arabic in vocabulary and script. There are different forms of poetry in Urdu e.g. Ghazal, Qasida, Marsiya, and Masnavi etc. [3].

Author attribution is the problem of identifying the author of a given text [4]. It is usually modeled as a multiclass classification task. A model is trained by using the texts (articles or letters etc.) written by the authors, and then a new unseen text is processed by the trained system to give the most appropriate author of the given text.

Poet attribution appears similar to the task of author attribution. However, there are some differences between author identification and poet attribution. The poems are usually shorter than texts (e.g. articles etc.) that are used in training and prediction of the author attribution system. We may have to identify the poet of a couplet (i.e. two lines or sentences) that consists of only 10 to 20 words. Moreover, some words of the poetic lines appear for rhyming requirements, so we are left with a lesser number of words to model the writing style of the poet.

There is very little research present about this topic, so it has never been presented conclusively what should be the feature set for such a system. How much data is required to make a stable system and most importantly in how many lines a poem should be split in the training phase to get good results. In this work, we tried to get answers to all of these questions. To achieve

[1]*Muhammad Ali Jinnah University, Karachi, Pakistan | tafseer.ahmed@jinnah.edu*
[2]*DHA Suffa University, Karachi, Pakistan | adil.rao@dsu.edu.pk*

this goal, multiple poet attribution systems for Urdu language using different machine learning algorithms are implemented and results are compared. Systems have been trained and tested on different feature sets and different length of poems.

2 Literature Review

The work under the computational linguistics for the poet classification or the author attribution in poetic text showed very thin contributions. Some of these techniques, specially for the languages other than English, are presented in the following:

The techniques broadly catered in most of the these works are the term-based such as a bag-of-words, n-gram; probabilistic approaches such as linear interpolation; and supervised Machine Learning (ML) algorithms such as Support Vector Machines, Naive Bayes, and decision trees such as C4.5 and ID3 (with further enhancement of employing ensemble techniques, for example random forest). The lately mentioned techniques, i.e. supervised ML, are used with different preprocessing steps of term-weighting (for example tf-idf).

Author attribution was employed by using a variational ensemble of PCFGs [5]. The task was to identify multiple text categories, i.e., football, cricket, business, travel, and poetry. The system models the author's writing style. There were two important studies for the Turkish poetry of the Ottoman period [6][7]. They modeled the poetic style by employing supervised ML approaches.

Researchers focused on the investigation of the dependencies of word count and choices of words for building word-vectors e.g. [8]. The Burrow's Delta Method [9] is employed in this study. There are other works [10][11] that used the extension of Burrow's Delta Method.

Another study [12], in the same regard, contributes to find the minimal data requirements for the author attribution. The study involves the text in the Latin script (English, German, Polish, and Hungarian novels). [13] is an important work on authorship attribution for the Arabic poets. The experiments involve the feature extraction based on the prosodic rules and probabilistic approaches.

Bangla language is genetically related to Urdu [3]. A novel technique for the feature extraction for Bangla poetry was employed in [14]. The extracted features were orthographic, phonemic, and synthetic, features which are more than ordinary lexical feature selections. They used these features with supervised ML techniques on the dataset of four distinguished poets of Bangla, namely, Rabindranath Tagore, Jibanananda Das, Kazi Nazrul Islam, and Sukumar Roy.

We find only one contribution related to poet attribution for Urdu [15]. They modeled the poetry of Allama Iqbal in the training phase. While in the testing phase, they used the poetry Mirza Ghalib, and Nasir Kazmi alongwith poetry of Allama Iqbal. They showed results with the bi-gram and tri-gram level matching, and probabilistic linear interpolation.

As mentioned above, most of the systems used supervised machine learning to model Author/ Poet identification. Following are the major techniques that are used to solve the multiclass classification problem.

Support Vector Machine (SVM) is a supervised learning algorithm which trains itself from labeled data and gives an optimal hyperplane which categorizes new examples [16]. The optimal hyperplane is the one which has the maximum margin from the nearest points of all the classes. The points nearest to the hyperplane are Support Vectors. These are the extreme points in the data set and these points are used to find the optimal hyperplane.

Naive Bayes [17] is a probabilistic classification model for supervised learning that works on the Bayes theorem. It works on the probabilities of the features of training examples, assuming that the value of each feature is not dependent on the values of other features which means that all the features have equal effect on the outcome.

LSTM neural network is a special type of Recurrent Neural Network [18]. The special thing about LSTM is it has some contextual state cells which acts for it as long-term and short-term memory. This additional power of keeping contextual history makes neural networks able to make good predictions on sequential data because now it maintains history over the sequence of input. This history adds value to the current input and provides the context for the current prediction. As time passes it becomes very unlikely that current prediction would not depend upon very old input so with time LSTM also learns when to forget and when to remember.

3 Methodology

As briefly mentioned in the introduction (section I), our goal is not only the poet attribution, but we also want to compare different configurations of the pottery identification models and then we want to recommend the best configurations for Urdu poet attribution and other similar tasks.

For this reason, we designed multiple experiments which are explained in III.C. Before describing these experiments, we present the data and its pre-processing.

A Data

Data was scraped form rekhta.org and then poems are extracted using the python package Beautiful Soup [19] which is used for parsing HTML and XML documents. Rekhta is an online resource to get Urdu poetry in Urdu, Hindi and Roman scripts. Data of Urdu poetry in Urdu script was collected for this experiment. Details about the data is given below:

Table 1: Information About Data

Total Count of Poets	**1969**
Count of poems	32667
Total Lines of Text	525835
Average words per line	1969

For the sake of simplicity, top 5 poets with the greatest number of poems were selected for the experiment. Poets chosen for the experiment and count of their poems in our dataset is mentioned in Table II.

Table 2: Poets and Count of their poems

	Poet Name	No. of Poems
P1	Meer Taqi Meer	433
P2	Mirza Ghalib	426
P3	Nazeer Akbar Abadi	261
P4	Faiz Ahmed Faiz	252
P5	Allama Muhmmad Iqbal	191

B Preprocessing

The average length of (poetic) line is 8.92 words. In preprocessing, special characters and all the other Urdu non-alphabets were removed from the data. Then data was tokenized using python library NLTK (Natural Language Toolkit) [20].

We did not remove the stop words as the functions words are important in modeling the style of the author. It is different from the topic oriented text classification, in which only content words are used and stop words are removed.

C Experiment

We executed several sets of experiments to find the best system for poet attribution. We focused on the following, which we termed as learning options.

The first learning option is the classification algorithm. As Naive Bayes (NB) works well for text classification problems, and support vector machine (SVM) works well generally including text classification problems, we chose these two as the classification algorithms. Moreover, we chose a deep learning model, LSTM as a classification algorithm.

The second learning option was number of (poetic) lines used in a training example. We chose the values from 2 to 10 in different experiments to see whether less number of lines e.g. 2 or 4 in each training example can give us results comparable to training by using a larger number of lines in each training example.

Then we have the choice of ngram used . We can use unigram, bigram, trigram or higher ngrams as features. We chose three configurations: (a) only unigram (b) only bigram (c) both unigram and bigram combined.

We also want to know whether we should use all the ngrams as features, or we will use some percentage of high frequency ngrams . As a lot of ngrams, especially bigrams, are created when a large corpus is processed, it is better to choose important ones (and reduce the size of the featureset), if possible. We used percentage = 10%, 20%,.....,90%,100% for selecting high frequency ngrams.

The size of training examples is also important . We know that larger numbers of training examples give better results in machine learning. However, we want to discover the smallest size of training examples that is sufficient to give fair results. As in many low (annotated) resource datasets, we do not find many training examples, so we can have a rough idea of the smallest number of examples that can give fair or good results. We have maximum number of poems used = 50, 75, 100, ..., 225 or 250 as the different options for the size of training examples.

Every poem creates different number of examples on the basis of number of lines option. If we are creating 2 line examples, then the (upper limit) of training examples will be 4 times the case when we choose to create 8 line examples.

We wanted to run experiments using all of the above mentioned learning options, however all these experiments cannot be created and run manually one by one. On the basis of learning options mentioned above (excluding LSTM), we have to run 2x9x3x9x10=4,680 experiments. We wrote programming code to automate the process of creating, running and storing results of all these experiments.

The SVM and NB algorithms used the bag of word approach, hence we created feature sets out of words, and lost the information about the sequence of the words. In contrast, LSTM works on sequences of word features, hence most of the learning options mentioned above are not relevant to the LSTM. The only relevant learning option is number of lines , and we plan to run the LSTM on 2,3,...10 lines in different experiments.

4 Results and Discussion

As some extensive training was done and there were different systems to compare on the bases of different configurations. The table III gives the configuration of top 4 systems on the basis of accuracy.

Table 3: Top 4 & 10th Poet Attribution systems

Model	ngrams	lines	max poems	% of ngrams	Accuracy
SVM	uni+ bigram	10	100	20,30, 40%	0.8871
SVM	uni+ bigram	9	225	100%	0.8860
SVM	uni+ bigram	6	50	60%	0.8829
SVM u	ni+ bigram	7	125	40,50%	0.8720
.....	.....	.....	.....	.....	.....
NB	uni+ bigram	10	175 3	0-100%	0.8400

It is evident from the above table that all the configurations using SVM and combination of most frequent unigrams and bigrams outperformed the rest of the configurations that used only most frequent uni-grams or most frequent bi-grams. We achieved the highest accuracy of 88.7% with average accuracy of 81% using SVM and highest accuracy of 84% with average accuracy of 77% using Naïve Bayes classifier.

Further we visualized the relation of average accuracy with maximum number of poems by each poet, number of lines and other learning options. The graphs and discussion are present below.

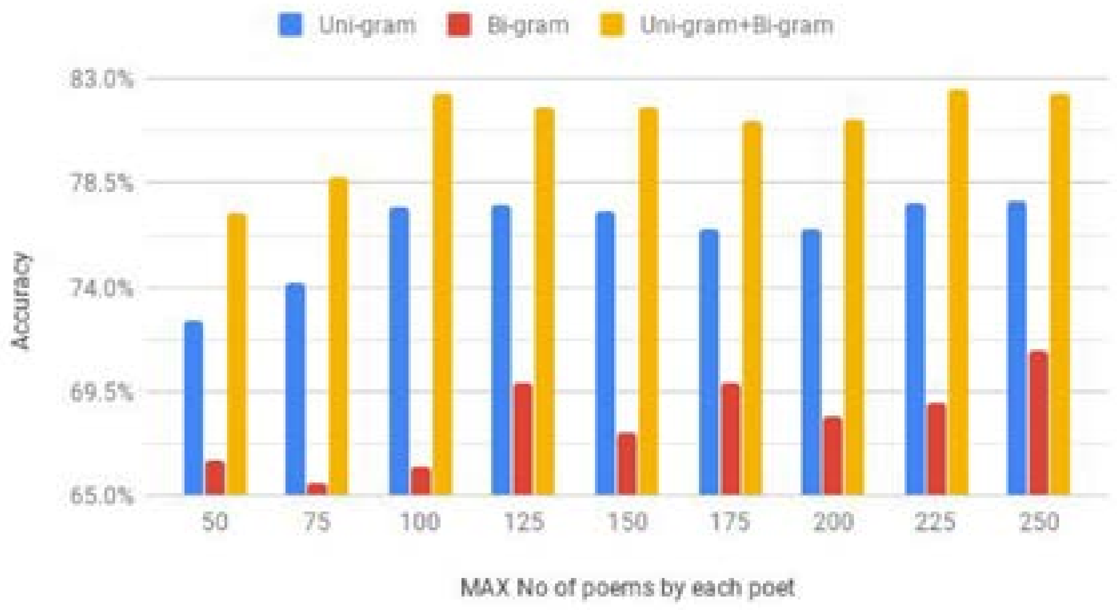

Figure 1: Accuracy for max. number of poems by each poet using SVM

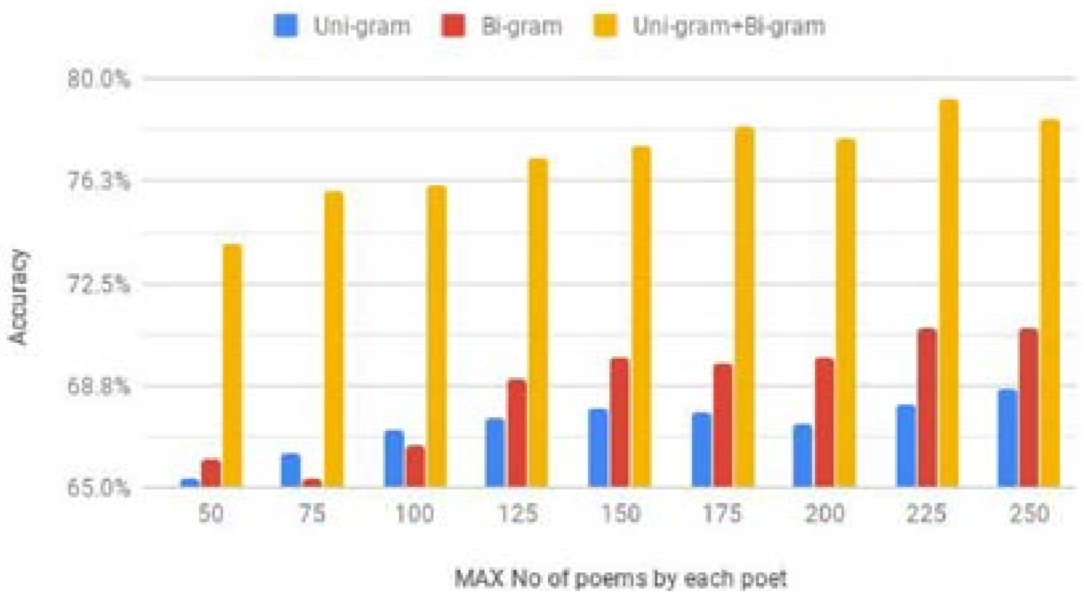

Figure 2: Accuracy for max. number of poems by each poet using NB

The figure 1 and 2 shows the accuracy corresponding to the different (maximum) count of poems per poet used in the experiment. The bar graph shows the accuracy when only ungram, only bigram, or a combination and unigram and bigram are used.

It is clear from these graphs and subsequent graphs that the combination of unigram and bigram outperforms the use of only unigrams or only bigrams. In the unigram+bigram setting, the individual words (As unigram) and multiwords (as bigram) voth can get suitable weight/ probability for the classification task.

Similarly the graphs in figure 1 & 2, and also in figure -3-6 shows that Support Vector Machine (SVM) is a better classifier than Naive Bayes (NB), as former give higher accuracy in almost all the cases.

In most of the cases, the accuracy increases or get uniform with the increase in maximum number of poems used. (The exception is bigram accuracy with SVM.) Hence, we found that using more examples is better. However, accuracy remains almost the same after 100 poems. So 100 poems is also a suitable size for any future work.

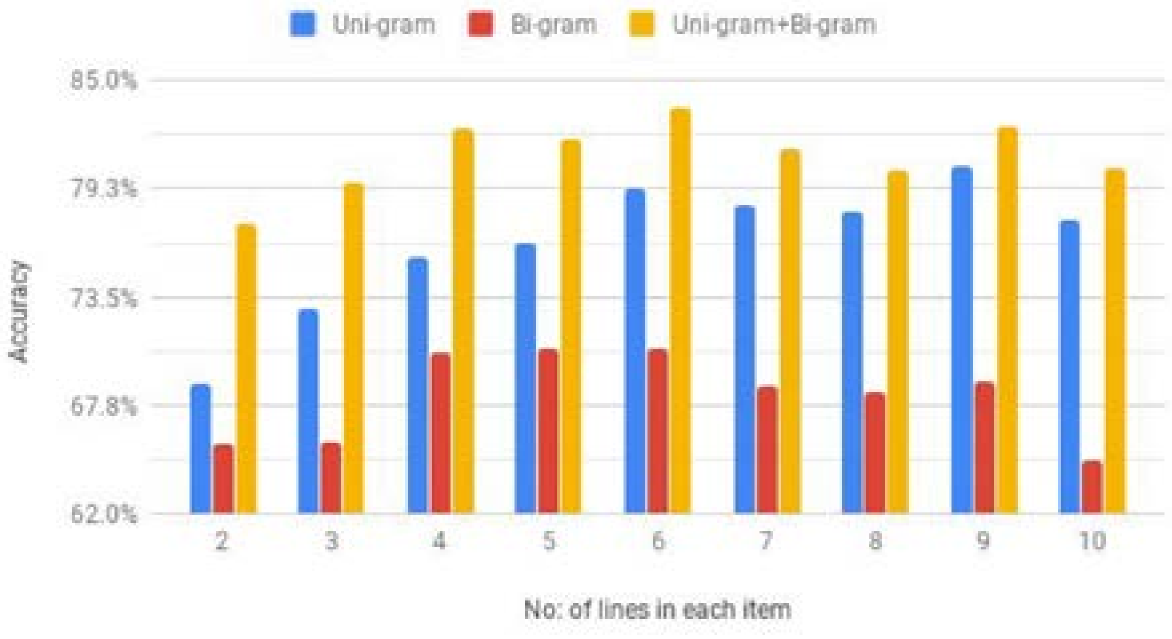

Figure 3: Accuracy for number of lines in each poem using SVM

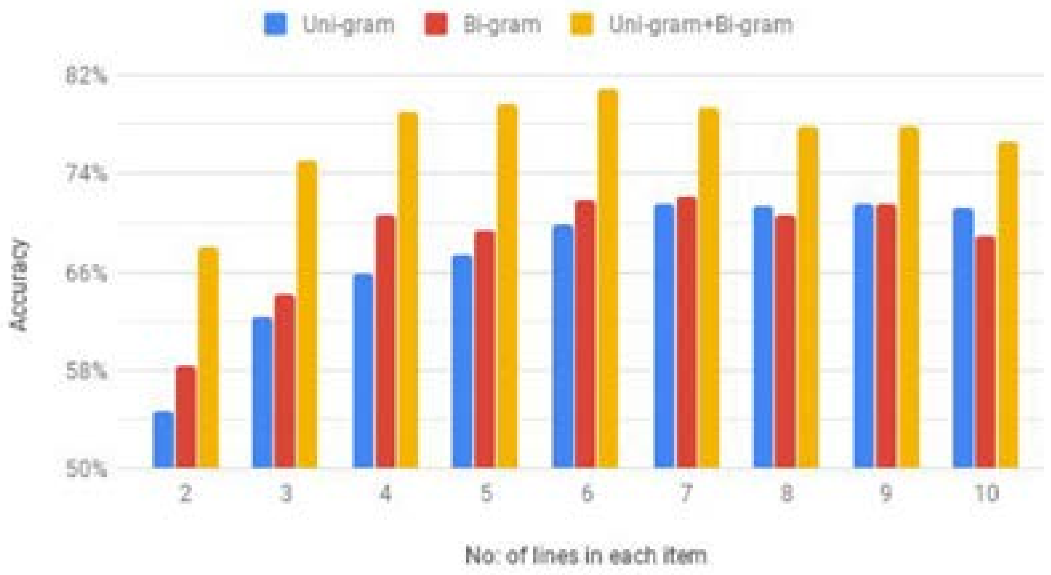

Figure 4: Accuracy for different number of lines in each item using NB

In figure 3 & 4, we present the accuracy corresponding to the number of (poetic) lines used in the training examples. The accuracy is lower for two lines (i.e. one couplet شعر.(However, it gets good for 4 or 6 lines. It means that our machine learning system needs more than one couplet to correctly predict the poet. As we are modeling the writing of the poet, we need more words.

As mentioned earlier, the average word size of the line is 8.92. So, we can infer that the use of more than 50 words (6*8.92= 54) in training examples gives a better model for writing style attribution. It must be noted that the usual text classification tasks involve modeling the frequency of keywords. However in author and poet attribution, the function words become more important. As function words are used by all the writers, their relative frequency usually suggest the difference in different author's/poet's style. For relative frequency, we need bigger text i.e. more number of words in the training examples.

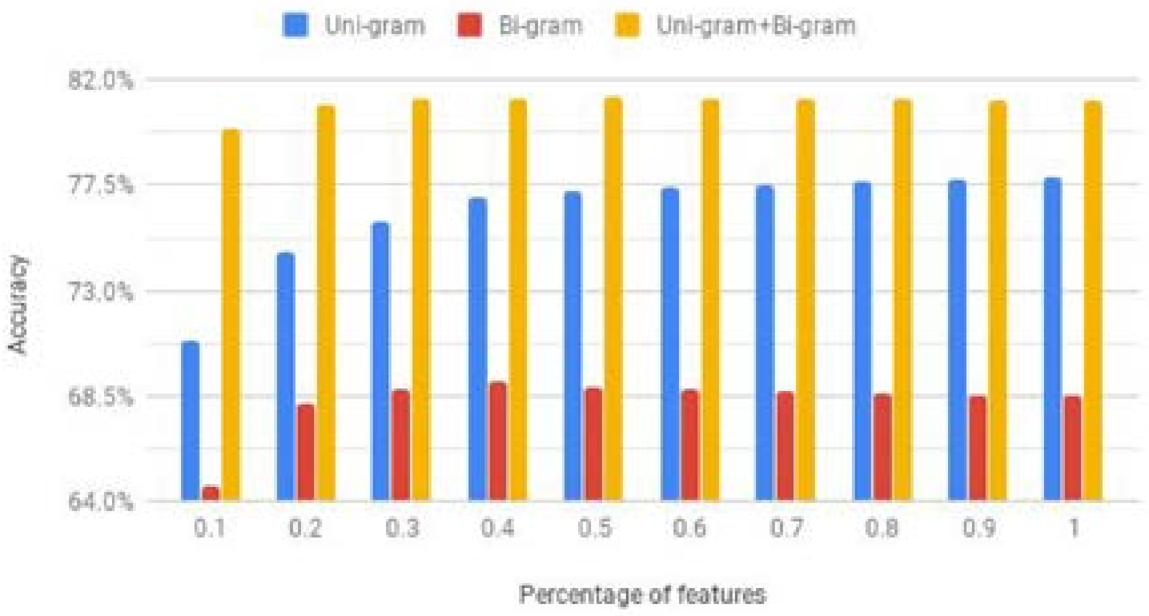

Figure 5: Accuracy for percentage of features using SVM

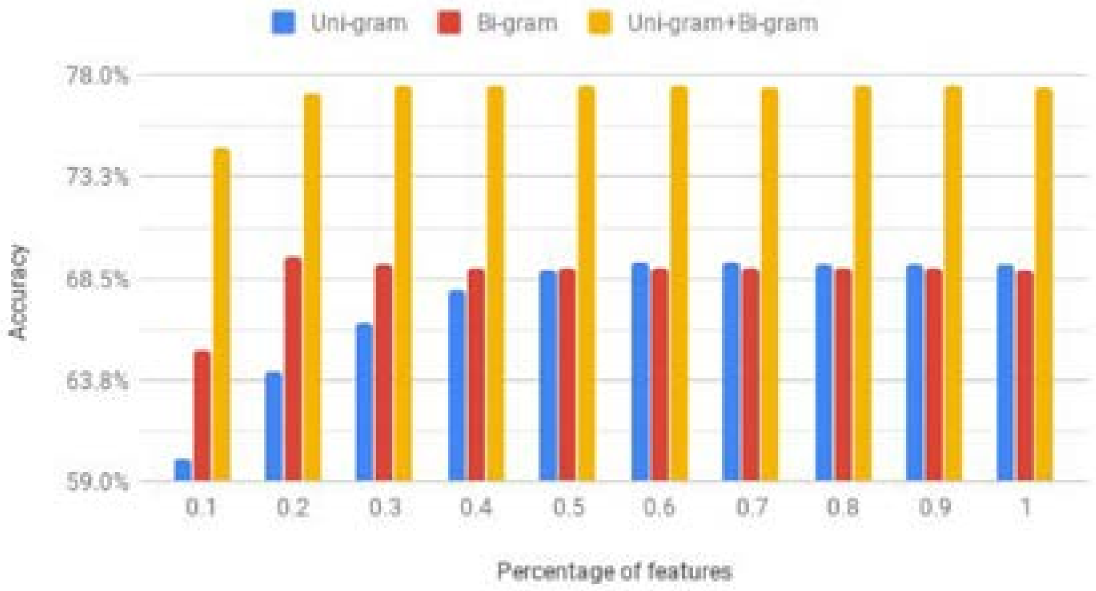

Figure 6.: Accuracy for percentage of features using NB (Naïve Bayes)

In figure 5 & 6, we present the accuracy corresponding to the percentage of ngrams used as features. We may not need to use the frequency of all ngrams as the features in the training. Zipf's Law tells that few high frequency words correspond to the majority of tokens in the text. Hence, we chose different percentages of high frequency words as features. The value 0.1 in the above chart means that we sorted the list of ngram on the basis of high to low frequency, and chose the top 10% ngrams from this list. Similarly the value 0.9 at y-axis shows that top 90% high frequency words were chosen.

The graphs in figure 5 & 6 shows that accuracy becomes almost uniform after 0.2 i.e. top 20% of ngram features. Hence, we do not need to use all the ngrams as features. Around 20% high frequency ngrams can give good accuracy.

Hence, our general conclusion about bag of word approaches is that one should use support vector machine (SVM), combination of unigram and bigram, 6 or more lines (50 or more words) and 20% ngrams for training a good machine learning system for poet attribution. We may guess that this configuration may be beneficial for author attribution of other types of text too.

We also created a deep learning based system. As LSTM does not need feature creation, we do not need to create ngram and choose some percentage of those ngrams. We did not get good accuracy, as LSTM work well when huge data is provided, and our training data is too small to fulfill this requirement. The top three systems are listed in the following table. The best accuracy is almost half of the SVM based system.

Table 4: Top 3 LSTM system

No of lines	Error	Accuracy
2	1.4988	0.4578
5	1.9080	0.4290
3	1.6748	0.4166

5 Conclusion

We created a dataset for 5 Urdu poets and executed multiple experiments with different classification algorithms, training examples and feature configurations. We found that the SVM based classifier that uses a combination of unigrams and bigrams gives the best result. The best system gives 88.7% accuracy. These experiments show that we can create author attribution systems for short texts.

References

[1] Oxford Leaner's Dictionaries. https://www.oxfordlearnersdictionaries.com/definition/english/poem?q=poem

[2] G.F. Simons, & C.D. Fennig, "Ethnologue: Languages of Asia". SIL International, Dallas, 2017.

[3] T. G. Bailey, "A History of Urdu Literature". Association Press (Y.M.C.A.), 1932.

[4] H. Love, "Attributing Authorship: An Introduction", Cambridge University Press, 2002.

[5] S. Raghavan, A. Kovashka, & R. Mooney, "Authorship attribution using probabilistic context-free grammars". In Proceedings of the ACL 2010 (pp. 38-42), 2010.

[6] E. F. Can, F. Can, P. Duygulu, & M. Kalpakli, "Automatic categorization of ottoman literary texts by poet and time period", In Computer and Information Sciences II (pp. 51-57), Springer, London, 2011.

[7] D. O. Sahin, O. E. Kural, E. Kilic, & A. Karabina, "A Text Classification Application: Poet Detection from Poetry", arXiv preprint arXiv:1810.11414, 2018.

[8] P. W. Smith, & W. Aldridge, "Improving authorship attribution: optimizing Burrows' Delta method", Journal of Quantitative Linguistics, 18(1), (pp 63-88), 2011.

[9] J. Burrows, "'Delta': a measure of stylistic difference and a guide to likely authorship", Literary and linguistic computing, 17(3), (pp 267-287), 2002.

[10] D. L. Hoover, "Testing Burrows's delta", Literary and linguistic computing, 19(4), (pp 453-475), 2004.

[11] D. L. Hoover, "Word frequency, statistical stylistics and authorship attribution", In What's in a Word-list? (pp. 55-72), Routledge, 2016.

[12] M. Eder, "Does size matter? Authorship attribution, small samples, big problem", Digital Scholarship in the Humanities, 30(2), (pp 167-182), 2015.

[13] A. F. Ahmed, R. Mohamed, B. Mostafa, & A. S. Mohammed, A. S, "Authorship attribution in Arabic poetry", In 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA) (pp. 1-6), IEEE, 2015.

[14] G. Rakshit, A. Ghosh, P. Bhattacharyya, P., & G. Haffari, "Automated analysis of bangla poetry for classification and poet identification", In Proceedings of the 12th International Conference on Natural Language Processing (pp. 247-253), 2015.

[15] A. A. Raza, A. Athar, & S. Nadeem, "N-gram based authorship attribution in Urdu poetry". In Proceedings of the Conference on Language & Technology (pp. 88-93), 2009.

[16] M. A. Hearst, "Support Vector Machines", IEEE Intelligent Systems 13, 4 (pp 18–28), 1998. DOI:https://doi.org/10.1109/5254.708428

[17] S. Russell, & P. Norvig, "Artificial Intelligence: A Modern Approach", 3rd ed., Pearson, 2003.

[18] S. Hochreiter, & J. Schmidhuber, "Long short-term memory", Neural Computation, 9(8) (pp 1735-1780), 1997.

[19] L. Richardson, "Beautiful soup documentation", 2007.

[20] S. Bird, E. Klein, E. Loper, 2009, "Natural language processing with Python: analyzing text with the natural language toolkit", O'Reilly Media, Inc., 2009.