N-Gram Based Authorship Attribution in Urdu Poetry

Agha Ali Raza, Awais Athar and Sajid Nadeem
National University of Computer and Emerging Sciences, Lahore, Pakistan
{ali.raza, awais.athar, l070815}@nu.edu.pk

Abstract

Authorship attribution is an interesting problem in Computational Linguistics. Traditional author recognition systems for electronic text rely on techniques which train the system to the specific vocabulary and writing style of the writer and apply stochastic methods to judge a given text at byte, letter or word levels. In this paper we have developed a software system to apply one existing and one modified authorship attribution technique to Urdu poetry. This paper discusses the results of this exercise and also presents the issues and procedures related to the language model training and word level n-gram based recognition system development.

1. Introduction

Automated authorship attribution is the problem of identifying the author of an anonymous text, or text whose authorship is in doubt [1]. The foremost use of this technique is identification of the author of a disputed text. One such example is the *Federalist Papers*, of which twelve are claimed to have been written by both Alexander Hamilton and James Madison [2]. Other applications include plagiarism detection, criminal investigation based on clues like emails, letters or other forms of evidence about the suspect [3], signature-based detection of malicious code in antivirus software systems [4] and spam identification [5]. This technique has special importance in the age of web development in fields like Information Retrieval for classification, search systems and data mining. The various techniques developed for author identification are also applicable to topic/genre classification, sentiment classification and language and encoding identification.

The next section describes some of the related work already done in the domain of author attribution. Section 3 explains the problems faced during the data corpus collection and methods of their resolution. Section 4 lists down the methodologies followed in our software systems and Section 5 discusses the results. The conclusion is presented in Section 6 followed by possible future research directions in Section 7.

2. Previous work

Various methods can be used to analyze a text to identify a specific author. The available methods range from character and byte level matching techniques to letter and word level stochastic systems. These include Parts of Speech (POS) tagging based approaches [3] which develop a probabilistic model based upon word classes and compare the given text with a previously tagged corpus of the author in question. [6] and [4] suggest a character based recognition technique which is applied at byte level using n-gram frequencies, called *author profiles*. Character level processing makes their method very language independent. The author profiles are generated from the given text of the author. Similar profiles are developed for the test data as well and then a stochastic approach is used to compare the two profiles for calculating dissimilarity.

Their approach is different from the standard perplexity and cross-entropy based n-gram matching technique which is utilized in [7] and is quite often used in speech pattern matching for speech recognition. Simple maximum likelihood estimation (MLE) [8] based techniques can also be used. However, these often suffer from problems related to text length and ambiguous threshold. Finally, there are also techniques specifically for any given language that are based upon the frequency comparison of certain chosen word sets from the vocabulary of that language. A comparison of three such techniques has been presented in [9].

3. Data collection and cleaning

Complete Urdu works of Allama Mohammad Iqbal are available at the Digital Urdu Library [10] in Unicode text format. The text was downloaded to local files and 13 poems, consisting of 79 couplets, were separated to serve as the test data. The rest of the corpus was utilized to train the system. All the extra

characters like redundant line breaks, extra spaces, under-line characters used for separating poems etc. were deleted manually. Further processing of text was done due the following reasons, most of which are specific to Arabic and Arabic like scripts.

3.1 Extra spaces after non-joiner characters

In Arabic script, the non joiner characters like ا،د،ر،ذ،ڈ،و etc. present a problem in processing. These characters appear same visually when typed with or without spaces succeeding them. As a result, words that contain these characters are often mistyped with spaces after the non joiner characters, instead of being typed as single words. This is because the error feedback to the typist is only visual in nature. While this is not a problem in appearance, it poses difficulty when the text is processed for separating words where spaces are used as one of the word boundary tokens. As a result, words like خواب (dream) and سرمایہ (wealth, possession) are wrongly tokenized as (خو)(ا)(ب) and (سر)(ما)(یہ) (where the parenthesis indicate word boundaries). Such problems had to be solved manually and incorrectly inserted spaces were removed from the words in the corpus.

3.2 Omitted spaces after non-joiner characters

Another problem of similar nature arises when a non-joiner character occurs at word boundaries. The space is often missed which poses segmentation problems that are also not visually detectable by the typist. For example,

(1) یا مجھے ہمکنار کر یا مجھے بے کنار کر

which should be recognized as

(یا) (مجھے) (ہمکنار) (کر) (یا) (مجھے) (بیکنار) (کر)

is recognized as

(یامجھےہمکنارکریامجھےبےکنارکر)

i.e. a single word, if space characters are missed out after the non-joiner characters. Hence, in order to resolve this problem, we manually inserted all such missing spaces in the corpus where required.

3.3 English punctuation marks

If the corpus is analyzed only for counting words and discovering of the frequency of singletons and non-singletons, one can use punctuation marks as end-of-word boundary markers and omit them from analysis altogether. However, in the case of language model development, we allowed these marks because their usage is also one of the characteristics of any writing style (e.g. usage of commas, exclamation marks, quotes, lengthy comma separated sentences etc.). Since a lot of Urdu text available online includes punctuation marks from English as well as Urdu character sets, we had to normalize them.

All the English punctuation marks which have an Urdu equivalent were converted to their corresponding Urdu characters. e.g. comma , (U+002C) was converted to ، (U+060C), period . (U+002E) to ۔ (U+06D4) and Interrogation mark ? (U+003F) to ؟ (U+061F).

Another problem was to deal with the punctuation marks which have no counter part in Arabic Unicode, e.g. exclamation mark !, single quote ‘, parenthesis () etc. It must be mentioned here that we have dealt only with the punctuation marks which were present in available poetry corpus. The solution can, however be easily extended generally as well. We chose characters U+0600 to U+0603 to map the four Latin marks mentioned.

! (U+0021) was mapped to ؀ (U+0600)

' (U+0027) was mapped to ؁ (U+0601)

((U+0028) was mapped to ؂ (U+0602)

) (U+0029) was mapped to ؃ (U+0603).

3.4 Latin numerals

Dates and numbers contain Latin numerals (0, 1, 2, 3…) which become problematic when we are performing *collation* (section 3.5). Therefore, we manually replaced all Latin Numerals (U+0030 to U+0039) with Indic Numerals (U+06F0 to U+06F9) (۰, ۱, ۲, ۳…).

3.5 Normalization

Normalization was required for those combinations which also have a one character representation in Unicode, for example, ٰ + ا can be written as آ as well. This part was completely handled by the Urdu Collation Utility [11] which is available as an open source project from CRULP.

3.6 Diacritics

Problems also occur due to the optional nature of diacritics in Urdu which result in words like میں (me) and میں (in) being treated as the same. More examples of such behavior are words like پَر (over) and پُر (to fill), اِس (this) and اُس (that) etc. However, dealing with such problems was beyond the scope of our project and future work maybe done to either complete the corpus for diacritics or use techniques like POS tagging to categorize words into these different types.

The optional nature of the diacritics also prevents compound words that are joined together by *zair-e-izafat* (ِ), to be treated as one for example قائداعظم would be treated as two words if the writer forgets to put in the *zair*. Similar problems occur with other compounding rules as well. We solved this by manually cleaning the corpus and joining such words, which make sense only as compounds, into one.

3.7 Compound words

A decision had to be made about the compound words regarding whether to collapse them all to a single word status or keep them as separate words. In Urdu, two primary methods of compounding exist. The و based compounds (مرکبِ عطفی e.g. شب و روز and چنگ و رباب) and the ِ or کا، کے، کی based compounds (مرکبِ اضافی e.g. اسرارشہنشاہی and تیغ بےنیام). We decided to treat the constituent words in a compound as separate words (except for those categories of compounds, which were discussed in section 3.6) as this will allow at a later stage to perform analysis like "Number of و based compounds?". This would not be possible if we collapse all the compounds to single words. Moreover, by definition, a compound word is made up of separate words, so it was prudent to keep their separate status. Thirdly, the pattern of compounding and the use of compounds contain information about the writer's specific style of writing, so they should remain as compound words and not become joined single words.

3.8 Introduction of new markers

One major problem which arises while processing text is parsing the poetry. In case of prose, phrases and sentences are separated and marked by punctuation markers like period, comma, colon, semicolon etc. However, in case of poems such separation is not necessary. As a result, when we develop any language model based upon history (i.e. relationship between words), we face the problem of identifying the poem title and verse boundaries. This can be elaborated with the following example.

Table 1 : Line breaks and verse boundary issues

فرشتے آدم کو جنت سے رخصت کرتے ہیں کی عطا ہوئی ہے تجھے روزوشب کی بیتابی نہیں کہ تو خاکی ہے یا کہ سیمابی خبر تری نوا سے ہے پردہ زندگی کا ضمیر کہ تیرے ساز کی فطرت نے کی ہے مضرابی	فرشتے آدم کو جنت سے رخصت کرتے ہیں۝ عطا ہوئی ہے تجھے روزوشب کی بیتابی؋ خبر نہیں کہ تو خاکی ہے یا کہ سیمابی۔ تری نوا سے ہے پردہ زندگی کا ضمیر؋ کہ تیرے ساز کی فطرت نے کی ہے مضرابی۞

The right column of Table 1 shows two couplets from a poem of Iqbal. The first line is the title of the poem. If we develop n-grams using this text as such, we find that the n-grams will span across the verse boundaries as we are removing spaces and line breaks in the first step. For instance, in verse 1, کی بیتابی خبر will be treated as a trigram where we can see that کی بیتابی belongs to the first verse and خبر belongs to the second one. Due to the absence of boundary markers we are unable to differentiate between words belonging to the same verse, couplet (شعر) or poem, and those belonging to separate ones.

In order to solve this problem we introduced new markers into the corpus. The marker characters and their roles are given in Table 2.

Table 2: New verse, couplet and poem markers

Marker	Role
؋ (U+060F)	*Misra* (verse) marker. Marks the ending of a poetic line.
۔ (U+060E)	*Sheyr* (couplet) marker. Marks the end of a poetic couplet (which is also the end of a line)
۝ (U+06DD)	Title marker. Marks the end of title sentence or verse.
۞ (U+06DE)	*Maqta* marker. Marks the end of the last line of a poem (this is also a line and couplet ending)

After applying these symbols to the whole corpus, we can distinguish between words of the same unit and those belonging to separate units. The poem text written in the right column of Table 1 gets changed to that written in the left column after tagging.

4. Implemented techniques

The problem that we are addressing in this paper is different from most of the available methods in various aspects. Firstly, the available methods deal mostly with prose or technical writing where sometimes vocabulary alone can be a sufficient clue. However, in case of poetry, due to the restrictions of rhythm, the positional significance of a word is sometimes more important than just vocabulary. Secondly, while working with Urdu, we cannot directly use language specific techniques like the *stylometric measures* suggested in [9], where frequencies of occurrence of common non-contextual words of English, or patterns of occurrence of certain selected words of English, are being employed for authorship attribution. Such methods, if used, require finding equivalent word lists and word patterns for Urdu which are not readily available. Thirdly, a corpus of the complete works of the poet is available in our case, whereas it might be a problem to obtain such a corpus for any other given poet, especially the contemporary ones.

Based upon these differences, we have tried an MLE (Maximum Likelihood Estimation) approach with sentence probability at word form level (where the inflected forms of a word are treated as separate words). We used bigrams and trigrams with Good-Turing discounting for smoothing. The initial results for this method were not encouraging and did not merit further pursuit. Therefore, we have implemented and analyzed the following two techniques:

1. Author Profile based frequency matching [6] adapted to word forms instead of characters on a unigram and bigram level.
2. A simple probabilistic method specifically suited for our problem.

In order to test these computational techniques we have used four books containing Urdu poems of Allama Iqbal.

- Bang-e-Dara (بانگِ درا)
- Bal-e-Jibreel (بالِ جبرئیل)
- Zarb-e-Kaleem (ضربِ کلیم)
- Armughan-e-Hijaz (ارمغان حجاز)

The techniques that we have used are based upon word forms instead of word classes, or characters. The hypothesis is that every writer uses a specific vocabulary at word and sentence level, which is a small subset of the vocabulary of the natural language being used by him. Therefore, in all the writings of a particular author, some words (and word-combinations) will occur with greater frequency than others. By analyzing these high frequency words, one can get some estimate about the authorship. The second clue comes from the lowest frequency words (singletons) which also reveal information regarding the author and the subject of the text. When this assumption is extended to multiple words i.e. to sentence level, we actually need to compare the combination of words used in a specific text with the corpus of sentences of a writer. If this process is simplified to make it computationally practical, we have to use probability of a sentence instead of actually comparing all sentences and the Markov assumption of finite history to obtain the n-gram based model. Author recognition is carried out using this n-gram based model.

4.1 Author profile matching

This method was presented in [6] for character level n-grams but we have modified it for word form based n-grams. In this method we generate the bigrams for the input corpus or the author's original sample text; this is called the *Author's Profile.* A similar profile is generated for the test data. Now, let $f_i\ (n)$ be the frequency of the *nth* bigram in the Author's Profile. Let $f_t\ (n)$ be the frequency of the *nth* bigram in the test data. We calculate the dissimilarity between the two using the following formula:

$$\sum_{n \in profile} (f_i(n) - f_t(n))^2 \qquad (4.1.1)$$

Normalizing using the average frequency for a given n-gram, we get:

$$Sum = \sum_{n \in profile} \left(\frac{f_i(n) - f_t(n)}{\frac{f_i(n) + f_t(n)}{2}} \right)^2 \qquad (4.1.2)$$

$$Sum = \sum_{n \in profile} \left(\frac{2.(f_i(n) - f_t(n))}{f_i(n) + f_t(n)} \right)^2 \qquad (4.1.3)$$

Finally, to make it independent of the input text length we present the dissimilarity as (*Sum / (Profile Length*4)*), which is 0 in case of a perfect match and 1 in case of completely dissimilar texts. The test results are given in terms of similarity which is 1 - (*Sum / (Profile Length*4)).*

4.2 Linear interpolation of probabilities

In this simple technique proposed by us, we find the unigram, bigram and trigram probabilities P_u, P_b and P_t, where

$$P_u = \frac{\text{Number of test unigrams found in the corpus}}{\text{Total number of unigrams in the test data}} = \frac{N_{ut}}{N_u} \quad (4.2.1)$$

$$P_b = \frac{\text{Number of test bigrams found in the corpus}}{\text{Total number of bigrams in the test data}} = \frac{N_{bt}}{N_b} \quad (4.2.2)$$

$$P_t = \frac{\text{Number of test trigrams found in the corpus}}{\text{Total number of trigrams in the test data}} = \frac{N_{tt}}{N_t} \quad (4.2.3)$$

P_u can be interpreted as the matching between the vocabulary (of the test data and the corpus) and P_b and P_t indicate the matched style. We combine these values by linear interpolation as:

$$P = \lambda_u P_u + \lambda_b P_b + \lambda_t P_t \ , \qquad \lambda_u + \lambda_b + \lambda_t = 1 \quad (4.2.4)$$

Since while analyzing poetry, positional significance of words is more important than mere vocabulary, the trigrams should carry more weight ($\lambda_u < \lambda_b < \lambda_t$). Keeping in mind this hypothesis, we set the probability weights to the following values.

$$\lambda_u = 0.2, \lambda_b = 0.3, \lambda_t = 0.5 \qquad (4.2.5)$$

5. Results

The system was trained on a corpus of Iqbal's which consisted of 8808 verses. The techniques described earlier were applied to the test data belonging to the following categories:

- 158 verses (13 poems) of Iqbal that were not part of the training corpus.
- 180 verses (15 poems) of Mirza Ghalib
- 154 verses (9 poems) of Nasir Kazmi

Similarity between Iqbal's couplets and each couplet of the test data was computed and the results were averaged over each author. Table 3 summarizes the results found after calculating the author profile based bigram level similarity (BLS), the author profile trigram level (TLS) similarity and the novel linear probability interpolation (LPI) based on the methods described earlier. The breakdown of the unigram, bigram and trigram level matching for the LPI method is given in Table 4.

Table 3 : Similarity of authors with Iqbal

	Author Profile Matching		LPI (%)
	BLS (%)	TLS (%)	
Iqbal	96.24	12.64	37.33
Ghalib	97.21	7.09	28.01
Nasir	94.89	15.42	29.47

Table 4 : Unigram, bigram and trigram probabilities

	P_u	P_h	P_t	P
Iqbal	91.71	43.36	11.96	37.33
Ghalib	83.28	31.28	3.94	28.01
Nasir	83.66	34.35	4.86	29.47

A good similarity measure should give higher score for the couplets from Iqbal and lower score to the ones from other authors. It is evident from the results that the Author Profile Matching based technique does not perform well and is not able to distinguish between couplets by Iqbal and other authors. This can be explained by the fact that there is a significant difference between the size of the training and testing data.

On the other hand, the LPI based method performs a lot better and we can observe a relative difference of more that 20% between the couplets of Iqbal and other authors. Analysis of the individual n-gram probabilities suggests that while the percentage overlap decreases as we move from unigrams to trigrams, this difference becomes more pronounced, thus confirming our initial hypothesis that trigram probabilities should carry more weight.

6. Conclusion

This paper applies two n-gram based techniques of author attribution to Urdu poetry. A data corpus consisting of all the Urdu works of Allama Mohammad Iqbal was chosen and the corpus was cleaned up using different string replacement rules and an existing collation software. Some poems were separated as the test data, and the rest of the data was used as input to train the system. We applied one existing and one modified method based on n-gram probabilities. Upon observing the results of the system on couplets of Iqbal and two other poets, it was found that the author profile matching based method performed poorly. The simpler method based on interpolation of n-gram probabilities proposed by us successfully distinguished between the works of Iqbal and other poets.

7. Future work

Future research directions on this problem could focus on determining the role of singletons in authorship attribution. It would also be interesting to apply an optimization technique like Expectation Maximization to calculate the optimal weights for the

linear interpolation of n-grams. Furthermore, it would be worth investigating if we can give weight to factors like rhyme and rhyming words in case of Urdu poetry for identification as those are often the salient distinguishing features of certain poems and poets.

8. References

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

[2] D.I Holmes,. and R.S. Forsyth, The Federalist Revisited: New Directions in Authorship Attribution. Literary and Linguistic Computing, 1995. 10(2): p. 111-127.

[3] C.E Chaski,., Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations. International Journal of Digital Evidence, 2005. 4(1): p. 1-13.

[4] T. Abou-Assaleh, , et al. N-gram-based detection of new malicious code. 2004.

[5] J. Diederich,., et al., Authorship Attribution with Support Vector Machines. Applied Intelligence, 2003. 19(1): p. 109-123.

[6] V Keselj,., et al. N-gram-based author profiles for authorship attribution. 2003.

[7] V. Siivola, and B.L. Pellom. Growing an n-Gram Language Model. 2005: ISCA.

[8] D . Jurafsky,., et al., Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2000: MIT Press.

[9] J. Schaalje, B.A.J.L.H.G.B., Comparative Power of Three Author-Attribution Techniques for Differentiating Authors. Journal of Book of Mormon Studies, 1997. 6(1).

[10] Digital Urdu Library. 2008 [cited 2008 July]; Available from: http://www.urdulibrary.org/.

[11] CRULP. Urdu Collation Utility. 2008 [cited; Available from: http://www.crulp.org/software/langproc/urducollation.htm