Predicting the author as per the writing style using Markov model of language

Aalok Sathe
Bhaskaracharya Pratishthana and Sir Parashurambhau College
Pune, India
aalok.sathe@gmail.com

Keywords: Markov model, author prediction, text classification, supervised machine learning, N-grams, computational linguistics

Abstract

Literary composition has always been an important activity in human culture. Many different pieces of literature have been composed over the years, each having a unique sense of style. We can associate such a style of literary creation with each creator. We can model and analyze the writing styles of authors using *Markov-chain Language Models* based on occurrences of word N-grams, or put differently, based on which words are more likely to occur after certain words. This enables us to determine the author of a given piece of text, provided that we have read other works by the same author. In this paper, we develop an algorithm to identify the author of an unseen text using Markov models. We study its efficiency and performance for multiple "Markov model orders" (the length of the N-grams used in the process) and compare the results for other variables such as training size. We also address the question of choosing the optimal Markov model order for this data paradigm (i.e., literary text). This will find use in various fields where accurate clustering of text-like data is required.

1 Introduction

When we communicate, we follow a certain system of rules having a semantic inventory that allows us to communicate – this is commonly referred to as *language.* Apart from grammatical and linguistic rules, we also rely on the semantics. Certain words are more appropriate in a certain context than others. We have multiple possible ways of saying or conveying a semantic idea. However, we choose a specific way over other ways to convey our thoughts. This decision depends on multiple variables which may appear irrelevant at a glance but are not so. The manner in which we choose a certain style of communication depends substantially upon the context of the conversation.

Most of the times, one person's decision procedure is different from that of some other person. Naturally, we talk, write and communicate differently. A style of communication can be associated with an individual. This is because we have a certain idea from experience of how specific individuals have communicated with us or others before. Hence we can *predict*, to an extent, how a person may or may not communicate given a specific context. This has been one of the recent topics of research and also commercialization in fields like *computational linguistics* and *language technology.* In fact, the keyboards in our smart-phones make use of similar methods to predict what we might type next. This is also necessary in speech recognition to improve the speed of speech to text conversion since it has to match our rate of speech delivery.

Association of the style of communication to an individual is valid even regarding literary creation. We can associate a *writing style* with an author which distinguishes their literature or writing from that of other authors. In this paper, we develop a process opposite to predictive text. Based on the *past exposure* of the creations of multiple authors, which is essentially exposure to other pieces of text or literary content created by the authors, we can decide which author, among the ones whose writings are studied by us, is most likely to have written or composed the unseen piece of text in question.

In this paper, we address this problem by using a probabilistic representation of the English language. In particular, we develop a Markov Language Model (MLM) of an author using probabilistic frequency tables of N-grams. The unknown piece of text is tested against such a model by comparing the probabilities of occurrence of all words with respect to the contextual k words occurring before them. For this, a quantified score is assigned to a piece of text with respect to the MLM of an author. By comparison of these scores across authors, we are able to determine the most likely author of the given sample. The accuracy and performance of this approach also depends on the number of words used as context and the number of words in the training text corpus. We discuss the performance of this method and its dependency on the order and the size of the training data.

In what follows, we present a brief review of the related literature, present our classification method and algorithm, describe the data used, present test results showing the performance of our method, and discuss future work directions.

2 Related work

In 1913, Andrey Markov (previously known as Markoff) demonstrated the use of Markov-chain to model the occurrence of vowels and consonants in the novel *Eugene Onegin* by Alexander Pushkin. Since then, Markov Models have been used in various fields to model sequentially correlated data via conditional probability. Similarly, Markov models can be used to understand text, since language is a sequentially correlated expression. If we take a bunch of words and arrange them randomly, it will not make sense, therefore we must analyse the context in which these words are known to, or permitted to occur.

Several attempts have been made to classify text. We try to mention some of the notable ones here. Classification of text into authors using character bigrams has been done before by applying Markov chains to individual characters instead of words (Khmelev and Tweedie, 2001). One approach treats the text classification task as predicting the best possible hidden sequence of classifiers based on the observed sequence of text tokens. (T. Mathew, 2006). This method considers the possibility that different sections of a large block of text may hint towards different yet related text categories and the Hidden Markov model predicts such a sequence of categories. Another paper proposed a Machine Learning approach to determine an unknown author's age and gender using previous corpus data. (K. Santosh et al., 2013). A study implemented an N-gram model to optimize the Naive Bayes model to classify text (Peng and Schuurmans, 2003). There have also been attempts to predict an author's age based on the author's text creation using a linear regression model (Nguyen et al., 2011). Age classification experiments have been conducted on a wide range of types of data, such as blogs (Schler et al., 2006), (Goswami et al., 2009) and phone conversations (Garera and Yarowsky, 2009). Medical texts were classified with the application of Hidden Markov Model (HMM) using prior knowledge in medical text classification. (Yi and Beheshti, 2009). Classifying text into categories using N-grams has been done before using character-N-grams but not using word-N-grams or MLMs. (Cavnar and Trenkle, 1994). Use of Markov-chain model for classification of genome sequences has been implemented. (L. Narlikar et al., 2013).

An extensive literature review in this field would be vast and is beyond the scope of this paper. We have thus mentioned select and recent developments in the field.

In this paper, we apply a Markov model to represent text. This implementation is far simpler than HMMs and various regression models, in terms of both, the complexity (time/space) of the model as well as computer programming and is also generalizable to an extent. It also shows results that are comparable with those of HMM or regression, if not better.

3 Mathematical background and method

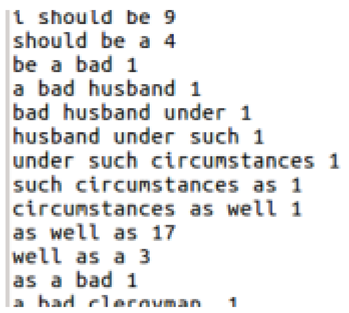
In this section, we discuss the mathematical background, the creation of an MLM (3.1), the method of classification of text and the implementation of it (3.2). We also discuss a scale to determine the closeness of a given sample of text with respect to an MLM (3.3).

3.1 Creating a Markov Language Model

Before attempting to assign a piece of text to an author, we must first have seen multiple pieces of text from various authors. We formalise this 'seen' text data into a form of data convenient to us: N-grams. Our smallest unit of text for the purpose of this method is a *word*. In the expression (3.1:1), W_i is the i^{th} word in the text.

$$text \equiv \{W_1,\ W_2,\ W_3,\ \ldots,\ W_i,\ \ldots,\ W_{n-1},\ W_n\} \tag{3.1:1}$$

Let us call the pieces of text that we use for creation of the MLM as *training data*. The model is based on documenting the frequencies of occurrences of N-grams in the training data. For this, we scan the data and tabulate all unique N-grams and track the frequency of their occurrence in the training data. Similarly, we can construct multiple data tables for various values of N.



```
i should be 9
should be a 4
be a bad 1
a bad husband 1
bad husband under 1
husband under such 1
under such circumstances 1
such circumstances as 1
circumstances as well 1
as well as 17
well as a 3
as a bad 1
a bad clergyman 1
```

Figure 1: Example of trigram MLM data

In the MLM table entry: $\{W_i\ W_j \ldots y\}$, y is the number of times the sequence '$W_i\ W_j \ldots$' has occurred in the training data.

Shown in figure 1 is an example for $N = 3$ of some training data. Interpretation: the group of words "as well as" has occurred 17 times in the text, in that order, whereas the less common group of words "a bad husband" occurs only once. Note that we retain the basic punctuation as it carries additional semantic data. For instance, certain words are more likely to occur at the end of a sentence, denoted by a period.

In this manner we create tables for $N = 1$ (unigrams), 2 (bigrams), 3 (trigrams) and 4 (tetragrams).

3.2 Computing the score

Here, we discuss the computation of the score and define a score function (3.2.1). We will also see other variables affecting the score (3.2.2), and a method to handle unseen data in the text to be classified. We also compare this method with an existing one. (3.2.3).

3.2.1 Markov chain and the score function

The Markov-chain probability (P_M) of a piece text with respect to an MLM is given as:

$$\begin{aligned}\mathrm{P}_M(\textit{text}) \doteq{}& \mathrm{P}(W_1|\star) \times \mathrm{P}(W_2|W_1) \times \mathrm{P}(W_3|W_1, W_2) \times \\ & \cdots \times \mathrm{P}(W_n|W_1, W_2, \ldots, W_{n-1})\end{aligned} \tag{3.2.1:1}$$

where $P(a|b)$ is the conditional probability of event a given that event b has occurred before and '$\star$' is used as a wildcard. This is essentially the probability of the event that the piece of text can be found in the data which was used to create the MLM. However, this is impractical for large values of n (total number of words in *text*) as longer the sequence, the less likely it is that we have seen it in the training data.

Markov's approximation:

$$\prod_{i=1}^{n} \mathrm{P}(W_i|W_1, .., W_{i-1}) \approx \prod_{i=1}^{n} \mathrm{P}(W_i|W_{i-k}, .., W_{i-1}) \tag{3.2.1:2}$$

The probability of occurrence of a word given all the previous words is approximately equal to the probability of that word occurring in the text given the previous k words, where k is the order of the Markov model.

Given a piece of text, we associate a numerical *score* with it based on the MLMs of the authors in our database. We associate a different score relative to each author which enables us to compare the scores across all the authors to determine the best match. For instance, let the scores of a piece of text for authors a and b be x and y, respectively. Suppose $x > y$. Then we say that the text is most likely to have been produced by author a rather than author b. The score function is recursively defined in terms of the order (k) and total N-grams in text (n). For a higher order chain, we need to define the initial case using a lower order as we don't have any context available in the beginning of the text. This is taken care of in the recursive definition.

$$\begin{aligned}S(k, n)\ \doteq{}& \Big[\prod_{i=k+1}^{n} \mathrm{P}(W_i|W_{i-k}, \ldots, W_{i-1})\Big] \\ & \times S\big(k-1, k\big)\end{aligned} \tag{3.2.1:3}$$

As this score is generally numerically very low, the computation process encounters a numerical underflow. To overcome this, it is convenient to take the summation of the logarithm of the terms at each stage of computation. We will call the score evaluated in this manner 'logarithmic score' or S_{log}. We evaluate the score recursively as long as the arguments of the function are within bounds.

That author, whose MLM produces the greatest *score* among the set of all authors known, is most likely to have composed the unseen piece of text.

3.2.2 Order of the MLM

Here, $k \in \{0, 1, 2 \dots\}$ is said to be the order of the chain. A 0^{th} order chain computes the score based on the frequencies of occurrence of individual words in the training data. This means that any word can have occurred before. A k^{th} order chain will give us the conditional probability score with the context of k previous words.

It was shown that the accuracy of prediction using a Markov model is dependent on the order of the Model (k) in genomic sequence analysis (L. Narlikar et al., 2013). Here, we demonstrate how that observation holds for MLM based text classification and also determine the optimal order for 'text' data.

3.2.3 Handling unseen data using a *back-off* model

Our basic unit is a word. The "alphabet" for the Markov models we wish to use consists of these word units. Since the number of words in any natural language can be exceedingly large, it is impossible for us to have seen an occurrence of all words and their N-grams at least once. As our alphabet is huge, we have to resort to approximations to deal with the unseen data.

If an N-gram is present in the input text but not in the MLM, we assume to have seen it once and compute the score, if there is there is at least one occurrence of the $(N-1)$-gram. If not, we skip $N-1$ entirely and use the same approximation to estimate the score using the frequency of the $(N-2)$-gram, and so on.

A similar method was proposed and demonstrated for sparse data in speech recognition (S.M. Katz, 1987). Here, a method similar to that of Katz's "back-off model" with certain modifications is implemented to deal with the unseen data.

In case the frequency of occurrence for some sequence $\#(\dots)$ is zero, our entire score evaluation will return a zero value as we are taking a product. We assume to have seen all possible N-grams at least once before. Since the maximum N we deal with is 4, the following is the procedure for data smoothing using a back-off model similar to that of Katz. We will discus the differences between the two back-off models shortly.

Let's say we wish to compute the probability of seeing the tetragram '$ABCD$' in the MLM. We will take cases for the implementation of our back-off model.

1. We have seen $ABCD$ in the tetragram data. $\implies$ We have also seen ABC in the trigram data. Then $\mathrm{P}(D|ABC) \doteq \frac{\#(ABCD)+1}{\#(ABC)+V_{ABC}}$ where V_{ABC} is the *vocabulary*, i.e. the number of unique occurrences of any tetragram $ABCx$ in the list of 4-grams (x is a substitutable wildcard placeholder). This is because we are incrementing all tetragrams in the list by one, hence we must only include the unique appearances. We change our earlier definition (3.2.1:4) and apply this new definition that accommodates unseen data.

2. We haven't seen $ABCD$ in the tetragram data.
   (a) We have seen ABC in the trigram data. Then $\mathrm{P}(D|ABC) \doteq \frac{1}{\#(ABC)+V_{ABC}}$
   (b) We haven't seen ABC in the trigram data either. Then we *back-off* to a lower order, i.e. k=2. We reduce the order from 3 to 2, hence we make the transformation: $ABC \to BC$. Then $\mathrm{P}(D|ABC) \doteq \frac{1}{\#(BC)+V_{BC}}$
   (c) Similarly, if we don't find BC in the bigram data, we *back-off* another step to just C and if it comes to the last step, $\mathrm{P}(D|ABC) \doteq \frac{1}{size\ of\ MLM+V_x}$ where V_x is the total number of unique unigrams in the unigram data.

This prescription provides a reasonably good and computationally efficient way of handling the unseen N-grams.

Another advantage of using this is that at each step where a match is not found, the score gets worse than before. So this method better reflects (un)likelihood of finding an N-gram with respect to our trained MLM. This is different from the Katz's model because that model backs off to the lower order N-gram and calculates the score of that N-gram. Instead, here we assume that we haven't seen even the lower order N-gram(s).

The Katz's approach fails in a case where let's say we want to compute $\mathrm{P}(C|AB)$ and we don't have any occurrences of the trigram ABC. However, we do have a very high conditional probability of seeing C after B. This means that we have seen the bigram BC multiple times before. In such a case, backing off to the lower order N-gram BC according to the Katz's model will give us a considerably high score while in reality it may be significant that the trigram ABC can not exist in the language itself.

However, using the back-off model implemented here, the score will either worsen or stay the same (rare case), but never increase. This gives a more accurate judgement of the probability of the desired N-gram.

3.3 Closeness of a text sample to an MLM

The best possible score S_{log} that can be obtained is 0. This can happen for a special case where a particular sequence of text appears y times and the total number of unique sequences is one: that is the only sequence that occurs. Obviously, $y = 1$ for $k \geq 1$ and $y = n_{MLM}$ for $k = 0$.

The worst possible score is when none of the N-grams in the text to be classified match those in the MLM. In this case, the score is the logarithm of $(n_{MLM})^{-n_{text}}$ where n_{text} is the number of N-grams in the text to be classified and n_{MLM} is the size of the MLM with reference to which the score is being calculated.

We define the scale of closeness with these extremes and compute the score to identify how close a sample is to a text. We can also compare closeness of one sample with one MLM with that of a different sample with yet another MLM using such a scale.

4 Description of data

For the purpose of testing of this method, literary creations of four authors are used. The amount of literature used for training and testing is large enough in size to generate sufficiently diverse data in terms of the number of words in corpus, of the order of 10^5 words. This can be further scaled to as many authors as required. The list of the authors whose literature is used as data is given below.

1. Arthur Conan Doyle, 2. Jane Austen, 3. Charlotte Bronte, 4. Daniel Brown.

Multiple books of these authors have been used as data. The data is split into training and testing parts. For comparison, different analyses using each of these kinds of data are prepared.

For each type of testing, the unseen data was split into snippets. Shown below are the data combinations for the process of classifying some author's texts in all of the tests mentioned below. This process is repeated multiple times for each of the four authors and the average performance is considered.

$$\left\{\begin{array}{c}\text{Size of training data}\\ \text{in words:}\\ 25\times10^3,\ 50\times10^3,\ 75\times10^3,\\ 100\times10^3,\ 125\times10^3,\ 150\times10^3\end{array}\right\}\left\{\begin{array}{c}\text{Order of MLM:}\\ 0,1,2,3\end{array}\right\}$$

4.1 Training-on-training (Test set - α)

The very first test of the model is to be able to successfully classify the data which was used to create the model in the first place. This set of data contains text that was used to create the MLMs.

4.2 Testing from the same books (Test set - A)

This set of unseen test data contains text obtained by splitting all books into snippets and using equal number of snippets per book as training and unseen data and picking out the unseen data. i.e. for x snippets, random $x/2$ snippets are used as training data along with other books and the remaining $x/2$ are treated as unseen data to be classified. The purpose of this test is to verify the functioning of the method with known similar data. We also wish to observe the known fact through this test that the writing style of an author tends to be similar throughout the same book or novel.

4.3 Testing from different books (Test set - B)

This set of unseen test data contains text obtained by forming snippets of text from only select books. We keep some snippets as training data and keeping the remaining books as unseen data. This means, the testing data does not belong to any of the books that were used as training data. The purpose of this test is to show that the writing style of an author is carried across books and different kinds of literature.

5 Experimental setup

To test as mentioned above as per test sets α, A and B, snippets of uniform length (in words) of the data are made.

Score is calculated for each snippet using MLMs of all authors for the orders in {0,1,2,3} for different sizes of the MLM. The performance of the method with varying orders and varying sizes of MLMs is compared.

To compare accuracy, *confusion matrices* are constructed for each test. Let $M_{r,r}$ be a confusion matrix of r rows and columns. Let $a_{i,j}$ denote the element in the i^{th} row and j^{th} column. Then $a_{i,j}$ is the number of times the algorithm assigned $author_j$ to the sample when it was actually $author_i$. Here $author_i$ is the i^{th} author in the list of authors (refer to section 4). We say that a particular classification result is correct if the author determined by the algorithm and the actual author are the same. In other words, the more the values of $a_{w,w}$ $\forall$ w in $[1, r]$, the more the accuracy of the algorithm.

The cumulative error is calculated for each test set and each order.

$$error_M \doteq \sum_{i=1}^{r} \sum_{j=1}^{r} \left(a_{i,j} | i \neq j \right)$$
$$\Longrightarrow error_M \% = \frac{error_M \times 100}{(total\ no.\ of\ samples)} \tag{5:1}$$

Shown below is a confusion matrix for some arbitrary test data for the sake of example.

$$\begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 1 & 5 & 94 & 0 \\ 0 & 3 & 0 & 97 \end{bmatrix} \rightarrow error\ \% = \ 2.25\%$$

(ex. of 5:1)

For the purpose of testing, this error % is averaged over three observations, i.e. 1200 samples of text for each of the test sets for one size of the MLMs.

6 Discussion

In a gist, the writing style of an author is reflected even in small groups of N words for various values of N, and conversely, patterns of word groups can be associated with an author or a writing style.

In what follows, "size of the MLM" implies the amount of training data (in terms of the number of words) that was used for its creation.

We observe that the algorithm performs well for all data in the α test. (Figure 3). This is because all the input has already been seen. This confirms the functioning of the model and now it can be tested against unseen data. In case of the tests A and B, A gives a better result as compared to the B data. (Figure 3). This shows us that the writing style of an author tends to remain similar in the same book(s) as compared to the writing style across different books by the same author. One of the reasons is that the theme of literature tends to vary in different books while it is quite similar in a given book or a novel. Another reason is that different books by an author are usually chronologically separated. The author's style itself changes a little over time, which we can see from the data. However, with higher training data we are able to achieve high accuracy even for the writing style across different books by the same author. (Figure 2).

The performance increases going from k=0 to k=1 (k is the order of the MLM used for assigning the score). At k=1, we observe maximum performance. It dips again at k=2 and becomes worst at k=3. (Figures 2 & 3). For the 0^{th} order, our model is a bag of words. If the training data as well as test data is diverse enough, k=0 model tends to do well. However, the 1^{st} order model does *better* than this because it sees the context of the previous word and is able to predict and assign the score better. For 2^{nd} and 3^{rd} orders, we experience *over-fitting*. A large chunk of the input data presents in a way we haven't observed before. This in turn makes it worse to have a large context of two or three words. Since our alphabet (of word units, not characters) is practically endless, we observe the dip in performance as the order of the model increases.

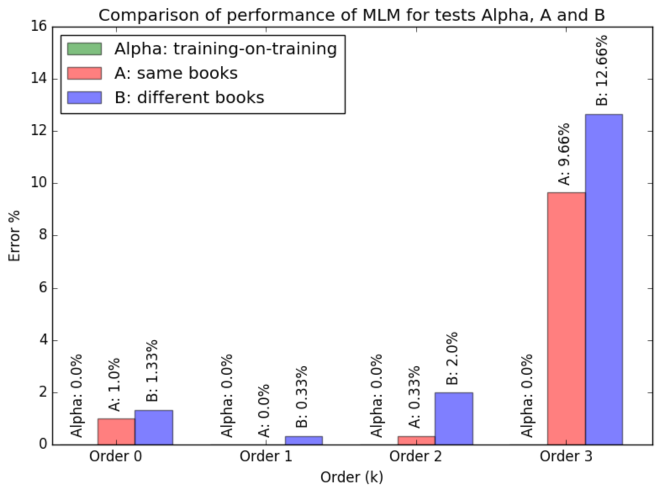


Figure 3: Comparison of performance of the model for tests α, A and B for size of MLM $(n) = 50 \times 10^3$ using an error plot.

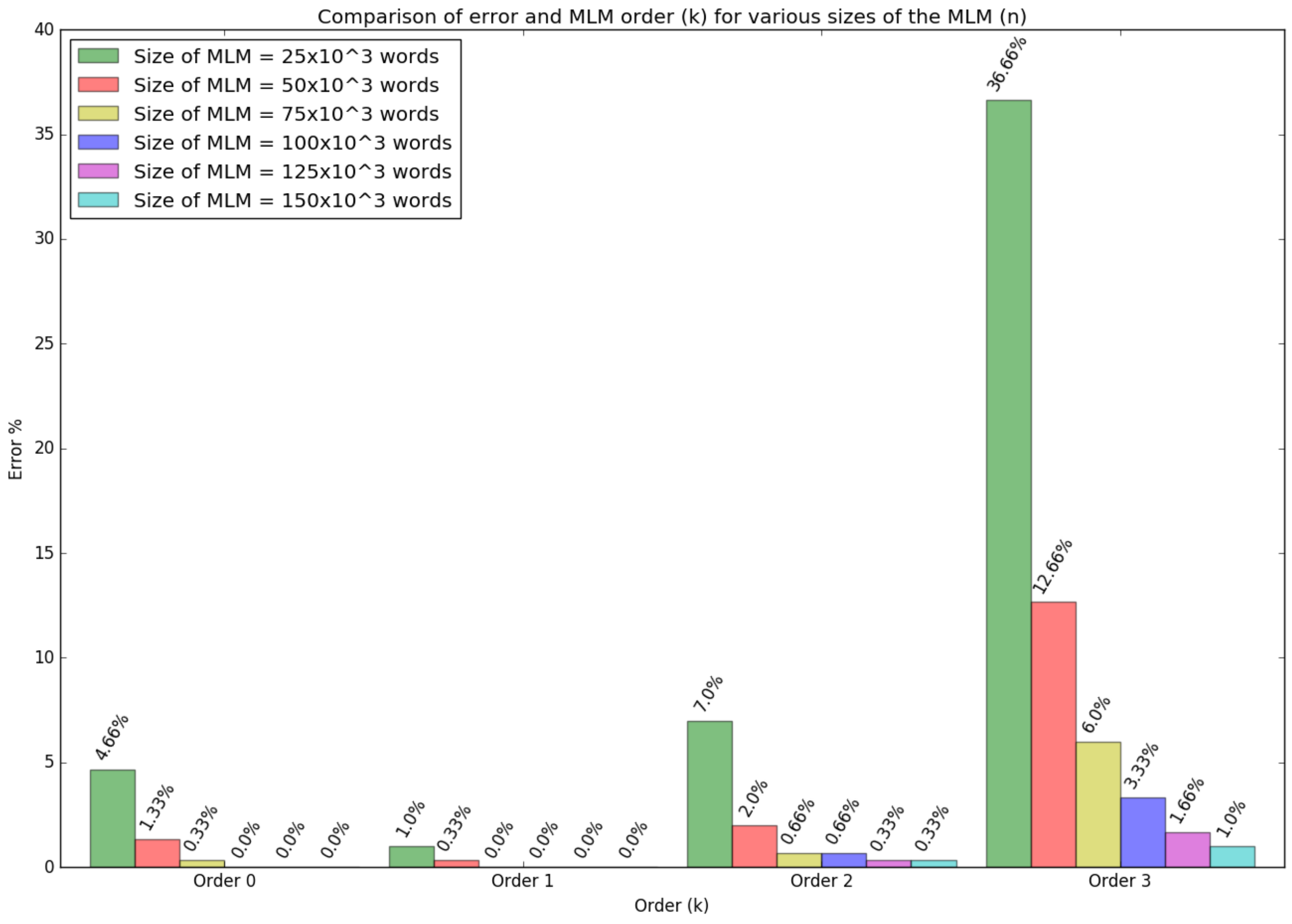


Figure 2: Comparison of performance with changing order and MLM training size using data from test B using an error plot.

We conclude that k=1 is the optimal Markov model order for this paradigm of data using word N-grams for MLM sizes $\approx 10^5$, 10^6 words. However, this best-performing Markov model order may increase to k=2 (and likely converge to that value) as the data increases (up to 10^7 or 10^8 words) which is another thing to be studied.

The accuracy also depends upon the amount of data seen as training data. Naturally, more data allows us to prepare a model which better represents the author. With more data, the over-fitting reduces. The performance increases as a logarithmic function of the amount of training data. (Figure 2).

In the future, we would like to compare the performance of a character N-gram model to that of a word N-gram model and determine which kind of a model better represents the writing style of an author, and in general, which kind of a model is better for this paradigm of data – language represented as text. For those forms of data in which the basic units that carry meaning or significance are smaller (in the sense that a character as a unit is smaller than a word), these smaller units would reflect the pattern(s) better, as opposed to writing or literary creation, where a word is the fundamental semantic unit rather than a character. A character, unlike a word, can be seen as the fundamental morphological unit.

We also wish to develop approximation methods to make the author prediction algorithm less dependent on the training data size because in real life, we use similar 'learning' methods which allow us (humans) to predict the source of the unseen text based on even small training data. The model can be improved by involving semantic and linguistic aspects such as a context-free grammar structure (CFG) to create a partially context based grammar. This can improve the accuracy above the already high accuracy level achieved and give us a head start in *natural language processing*. Practically, this can be used to detect plagiarism. We can also better understand our linguistic thought processes and how they can be modelled. This model and its implementation will find use in fields where recognition of patterns in data and learning them can be used to predict further states of the model.

7 Acknowledgements

The author thanks Leelavati Narlikar for suggesting the problem, help with the conceptualization of the idea, and her guidance throughout the project. The author also acknowledges the suggestions and comments from Kiran Barve and Mihir Arjunwadkar while writing the manuscript.

References

[1] The Gutenberg Project
The Gutenberg Project hosts electronic editions of several books in the open domain. It is also the source of data for this paper.
`https://www.gutenberg.org`

[2] A. A. Markov, 1913
Bulletin of the Imperial Academy of Sciences of St. Petersburg 7(3):153162. Andrey Andreyvich Markov *An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. (In Russian.) English translation by Alexander Y. Nitussov, Lioudmila Voropai, Gloria Custance and David Link, 2006. Science in Context 19(4):591600*

[3] S. M. Katz, 1987
IEEE Transactions on Acoustics, Speech and Signal Processing. Slava M Katz. *Estimation of probabilities from sparse data for the language model component of a speech recognizer.*

[4] Cavnar and Trenkle, 1994
http://www.psu.edu/ William B. Cavnar and John M. Trenkle. *N-gram-based text categorization.*

[5] Khmelev and Tweedie, 2001
Literary and Linguistic Computing, 2001, vol.16, no.4, pp.299-307. Dmitri Khmelev and Fiona Tweedie. *Using Markov Chains for Identification of Writers.*

[6] Peng and Schuurmans, 2003
Volume 2633 of the series Lecture Notes in Computer Science. Peng and Schuurmans. *Combining Naive Bayes and n-Gram Language Models for Text Classification.*

[7] Schler et al., 2006
Proceedings of the AAAI Spring Symposia on Computational Approaches to Analyzing Weblogs. Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. *Effects of age and gender on blogging.*

[8] T. Mathew, 2006
Thomas Mathew. *Text categorization using N-grams and Hidden Markov models.*

[9] Goswami et al., 2009
Proc. of ICWSM. Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. *Stylometric analysis of bloggers' age and gender.*

[10] Yi and Beheshti, 2009
Journal of Information Science 35(1):67-81, February 2009. Kwan Yi and Jamshid Beheshti. *A hidden Markov model-based text classification of medical documents.*

[11] Garera and Yarowsky, 2009
Proc. of ACL-IJCNLP. Nikesh Garera and David Yarowsky. *Modeling latent biographic attributes in conversational genres.*

[12] Nguyen et al., 2011
5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities. Dong Nguyen, Noah A. Smith, Carolyn P. Rose. *Author Age Prediction from Text using Linear Regression.*

[13] L. Narlikar et al., 2013
Nucleic Acids Res., 41, 1416 - 1424. Leelavati Narlikar, Nidhi Mehta, Sanjeev Galande and Mihir Arjunwadkar. *One size does not fit all: On how Markov model order dictates performance of genomic sequence analyses.*

[14] K Santosh et al., 2013
Notebook for PAN at CLEF 2013. K Santosh, Romil Bansal, Mihir Shekhar, and Vasudeva Varma. *Author Profiling: Predicting Age and Gender from Blogs.*