Computer Engineering Department
Faculty of Engineering
Deanery of Higher Studies
Islamic University – Gaza
Palestine

Author Attribution from Arabic Texts

Mohammed F. Eltibi

Supervisor
Prof. Ibrahim S. I. Abuhaiba

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

1434H (2013)

DEDICATION

To My Father, and Mother,

To My Wife,

To My Family, and Friends,

ACKNOWLEDGMENT

All thanks are to Allah the almighty, who guide me to accomplish this work, so all praise is to Allah.

Also the completion of this work cannot done, without all people around me, especially my advisor Prof. Ibrahim S. I. Abuhaiba, who guided me through this research; his patience and support leaded me to success through and until the completion of this thesis.

Table of Contents

LIST OF ABBREVIATIONS

AFP	Agency France Press
API	Application Programming Interface
ATB	Arabic Treebank
CFG	Context-free grammar
IG	Information Gain
JGAP	Java Genetic Algorithms Package
KNN	K- Nearest Neighbor
MI	Mutual Information
NLP	Natural Language Processing
PCFG	Probabilistic Context Free Grammar
POST	Part of Speech Tagging
RBF	Radial Basis Function
SOM	Self-Organized Map
SVM	Support Vector Machines
TiMBL	Tilburg in Memory Based Learner
UTF	Unicode Transformation Formats
X^2	Chi Square Score
XPCFG	Enhanced Probabilistic Context Free Grammar

LIST OF SYMBOLS

$\boldsymbol{G}$	Language Grammar
$\boldsymbol{\Sigma}$	Set of terminals
$\boldsymbol{N}$	Set of non-terminals
$\boldsymbol{S}$	Start symbol
$\boldsymbol{R}$	Set of production rules
$\boldsymbol{U}$	Set of punctuation marks
$\boldsymbol{P}$	Set of rules' probabilities
$\boldsymbol{P_T}$	Set of terminals' probabilities
$\boldsymbol{P_N}$	Set of non-terminals' probabilities
$\boldsymbol{P_U}$	Set of punctuation marks' probabilities
$\boldsymbol{VP}$	Verbal phrase
$\boldsymbol{VBP}$	Imperfect verb
$\boldsymbol{NP}$	Noun phrase
$\boldsymbol{PP}$	Prepositional phrase
$\boldsymbol{IN}$	Subordinating conjunction or Preposition
$\boldsymbol{DTNN}$	Common noun
$\ddot{\boldsymbol{\mathcal{L}}}$	Final log-likelihood
$\widehat{\boldsymbol{\mathcal{L}}}_{\boldsymbol{R1}}$	Log-likelihood of terminal rules
$\widehat{\boldsymbol{\mathcal{L}}}_{\boldsymbol{R2}}$	Log-likelihood of non-terminal rules
$\widehat{\boldsymbol{\mathcal{L}}}_{\boldsymbol{T}}$	Log-likelihood of terminals
$\widehat{\boldsymbol{\mathcal{L}}}_{\boldsymbol{N}}$	Log-likelihood of non-terminals
$\widehat{\boldsymbol{\mathcal{L}}}_{\boldsymbol{U}}$	Log-likelihood of punctuation marks

LIST OF FIGURES

LIST OF TABLES

تحديد هوية المؤلف من النصوص العربية

محمد فؤاد الطيبي

ملخص

تحديد هوية المؤلف من النصوص عبارة عن إسناد مؤلف إلى أحد النصوص مجهولة الكاتب. قمنا بعرض رؤية جديدة لحل هذه المشكلة، عبر استخدام نموذج اللغة، حيث يعتمد النموذج الجديد على تحسين النموذج المسمى "Probabilistic Context Free Grammar (PCFG) " عبر تدعيمه بالمزيد من الخواص النحوية والمعجمية، عبر إضافة الاحتمالات لكل من الكلمات ووصف الكلمة وعلامات الترقيم، بجانب الاحتمالات الموجودة لكل قاعدة كتابة التي تنتج عن استخدام النموذج PCFG. أيضاً تم تدعيم نموذج اللغة المقترح بدالة إسناد النقاط، التي تقوم بإسناد قيمة النقاط لكل من قواعد الكتابة. ولأن النموذج الجديد يحتوي على عدة خواص مختلفة، تم إضافة وزن لكل خاصية والذي يتحكم بمعدل مشاركة الخاصية في عملية إسناد مؤلف لأحد النصوص، ميزة استخدام العديد من الخصائص هو قدرة نموذج اللغة المقترح على وصف وتحديد العديد من طرق الكتابة المختلفة للمؤلفين. أيضاً استخدام دالة النقاط يساعد على تحديد قواعد الكتابة الأكثر فعالية في التفريق بين طرق الكتابة، بالإضافة إلى تجاهل قواعد الكتابة التي تؤثر على الكفاءة. إن استخدام الأوزان يدعم أيضا وصف طرق الكتابة المختلفة للمستخدمين، حيث أن وضع قيم مناسبة لهذه الأوزان يمكن أن يساعد على زيادة كفاءة النظام. تم تجربة النموذج الجديد على 9 مؤلفين، بحيث يوجد 20 نص عربي لكل مؤلف ، وقد تمت عملية التدريب والتجريب باستخدام طريقة Leave-One-Out. في حين أن عملية إيجاد القيمة المناسبة للأوزان تمت باستخدام Genetic Algorithm والذي استخدم مجموعة نصوص مختلفة عن المجموعة الأولى، حيث احتوت المجموعة الجديدة على 10 نصوص لكل مؤلف. حقق النموذج الجديد دقة بلغت نسبتها 95% حيث تعتبر هذه النتيجة تحسين على النموذج الأصلي بنسبة 3.5% ، أيضا تطبيق أوزان مناسبة يستطيع زيادة الدقة إلى 96%.

Author Attribution from Arabic Texts

Mohammad F. Eltibi

ABSTRACT

Author attribution is the problem of assigning author to an unknown text. We propose a new approach to solve such a problem, by using an enhanced language model, our model is an enhanced version of the probabilistic context free language model (PCFG), by supplying it more syntactic, and lexical information. So that behind the probabilities for the production rules generated from PCFG, we add probabilities for terminals, non-terminals, and punctuation marks. Also the new language model is augmented with a scoring function, which assigns a score for each production rule. Since the new model contains different features, weights are added to the model to govern how each feature participates in classification. The advantage of using many features is to successfully capturing the different writing styles for authors, also using a scoring function can help by identifying the most discriminate rules, and ignoring the general rules that can affect the performance. Using weights supports capturing different authors' styles, and setting weights properly can increase classifier's performance. The new model is tested over 9 authors, each has 20 Arabic documents, where the training and testing is done using Leave-One-Out method. The model achieves 95% of accuracy, which is an enhancement of 3.5% over PCFG. While searching for best weights is implemented using Genetic algorithm over a new corpus of 10 documents per author, this increase the accuracy to 96%.

Keywords: Author Attribution, Author Identification, Language Model, PCFG Language Model, Chi-Square Score, Genetic Algorithm.

Chapter 1

INTRODUCTION

1.1 Authorship Attribution Definition

Author attribution is the problem of identifying the author of an anonymous text, or text whose authorship is in doubt, by studying strategies for discriminating between the styles of different authors, also it can be defined as the automatic identification of the author of a text on the basis of linguistic features of the text.

1.2 Applications of Authorship Attribution

The old applications for author attribution include the traditional plagiarism detection, as settling disputes regarding the authorship of old historically documents, also its importance appears in many fields such as civil law, which including copyrights violation [1], and literary research. The most common problem is the authorship of the Federalist Papers, which is a series of 146 political essays written by John Jay, Alexander Hamilton, and James Madison. Twelve of which claimed by both Hamilton and Madison was undoubtedly the most influential work in authorship attribution. Plagiarism detection applications are also important in commercial field, and academic field. In commercial field the copyright problem is a traditional example. While in academic, author attribution can be used to detect plagiarism in college essays.

Another field that needs author attribution is criminal law, which includes determination of documents authority in courts, and forensic

linguistics. A real case [1] occurred when a government employee wrote an email to his supervisor in which he disparaged her racial heritage, after he was terminated, he sued the government, claiming that someone break his workspace and sending the email from his computer, in such a case the court needs to at least find if the employee wrote the email or not.

More recently, author attribution gained new importance in cyber crimes, including deducing the writer of inappropriate communications that were sent anonymously or under a pseudonym, and in a more general search for reliable identification techniques [1]

Another area where author identification and profiling can provide valuable information is in deriving marketing intelligence from the acquired profiles [2], and in the rapidly growing field of sentiment analysis and classification [3].

Author attribution appears in specific applications as recognizing the author of a program to help detect copyright violation of source code as well as plagiarism [4]. Also it helps the developing of the applications by identifying the author of non-commented source code that we are trying to maintain. Finally it is useful to detect the programmer of a malicious codes, and viruses [5]

Due to the growing increase in the number of documents (especially in the web), an automated text categorization is a useful way to organize a large documents collection, and a one useful categorization of documents is classifying documents by their authors. Author attribution is becoming an important application in web information

management, and beginning to play a role in areas such as information retrieval, information extraction and question answering.

The variety in the applications of author attribution, is returned to the categories of the author attribution problem. Our definition (identifying the author of an unknown text) is one of many categories of the problem, called author classification. Next section overviews some categories of author attribution problem.

In this thesis the term author attribution and author classification will be the same.

1.3 Author Attribution Categories

The scope of author attribution problem do not include only identifying the author of an unknown text, there are several author analysis tasks for authorship problem, including the followings:

1. Author Classification and Verification: in author classification we will decide the author for an unknown text, or in verification decide whether a given text was written by a certain author, this will be our study scope in this thesis.

2. Plagiarism Detection: we discussed previously that plagiarism detection is one from many applications for author attribution problem, in plagiarism detection we interested to find the similarity between two texts [6].

3. Author Profiling: this scope includes finding information about the author from his written texts; the information may include his age, education, sex, etc.

4. Detection of Inconsistent Styles: in this analysis the goal is to analyze a written text to detect if there exist parts of text, which are inconsistent with the others, this can help in case of collaborative writing, where many of authors write in same text[6].

1.4 Approach

The general approach that was used to solve such a problem starts from a set of training documents, which are documents whose authors are known, then a set of features that considered to be most informative in identifying the author are extracted, then a machine learning algorithm is implemented and learned using these features, to be able to classify a document with unknown author.

Researches were done to find the most informative features to be used for the author attribution problem, and best machine learning algorithm to be used to classify unknown text accurately. Next sections illustrate these features, and different attribution methods.

1.4.1 Features

Researchers tried to taxonomy the features that can be used in author attribution in order to quantify the writing style. The basic categorization is lexical, character, syntactic, and semantic features. Following we will describe each set of features.

(a) Lexical Features:

Using this set of features the text is viewed as a sequence of tokens that grouped into sentences, where a token is a word or punctuation mark. From this representation some features can be used, as length

of sentences, and length of words. Although these features are basic but its advantage is that they can be applied to any language with no additional requirements, but still we need a tokenizer tool to detect tokens and sentences boundaries. While these features are available for any text, these features may not capture the style of a written text, especially for texts contain lot of abbreviation.

Another features can be extracted from tokens is the vocabulary richness features which measure the diversity of the vocabulary of a text. A traditional example that used in author attribution problem is the type-token ratio described by *V/N,* where *V* is the size of the vocabulary which is number of unique words, and *N* is the total number of tokens. Another vocabulary richness features are the hapax legomenon, and hapax dislegomenon, which is words occurring once, and words occurring twice respectively. The vocabulary richness features are biased toward text length as they increased when the text length increased, so they are considered unreliable to be used alone.

A more efficient approach is to measure the frequency of each word, where the text is viewed as a set of words each having a frequency of occurrence disregarding contextual information, one can argue that words frequencies cannot capture authors style since they are topic dependent, actually this is true but the big advantage of using words frequencies is to specifying function words, which are words that have little lexical meaning but serve to express grammatical relationships with other words, and these words are proved to capture the style of the authors across different topics, although using function words can capture writing style of the

authors, but the selection of the specific function words require language dependent expertise. There are various researches were done to find best function words for author attribution problem [7].

While words frequencies feature computes the frequency of each word without regarding the contextual information, the *n*-grams take advantages of contextual information. An n-gram is a contiguous sequence of *n* items from a given sequence of text or speech, where an item is usually a word, and *n* is the number of grams that controls the level of context. *N*-grams were used as textual features in author attribution problem [8] and can achieve good results but not always, because they may capture content specific information rather than stylistic information.

Uncommon lexical features [9] measure various writing errors to capture authors writing styles. These features are captured using spell checker tools, however the accuracy of spell checker is problematic for many languages, and the available text always error-free since it is available in electronic form.

(b) Character Features:

According to these features a text is viewed as a sequence of characters, so that simple character level measures can be defined, as alphabetic characters count, digit characters count, letter frequencies, and punctuation marks. These features are available for any language, and can easily be found without needing any extra tools.

Another effective approach is to extract *n*-grams on the character level [10], character based *n*-grams are also computationally

simple, the approach is to extract the frequencies of each character based *n*-gram, this approach is able to capture nuances of style including lexical information, contextual information, and using of punctuation marks, the other advantage of this model is its ability to be tolerant to noise. In cases that the texts are noisy containing grammatical errors or making strange use of punctuation, the character based *n*-gram model is not affected dramatically. This model shows an acceptable result in author attribution problem, but such a method requires more experiments to find the best value for *n*. Also the dimensionality of this representation is considerably increased in comparison to the word-based approach, since many *n*-grams are needed to represent a single word, so that it may capture redundant information.

(c) Syntactic Features:

A more efficient feature, that can extracted from a text is the syntactic information, where the idea is that authors tend to use similar syntactic patterns, which are out of their consciously. In comparison to lexical and character level features, the syntactic features are considered more valuable to detect the writing styles of authors.

The first attempt to use syntactic features [11] was done by producing parse tree for each sentence in a document, and then extracting writing rules frequencies. The results of using these rules in author attribution problem is acceptable, but syntactic features alone performed worse than lexical features, also the syntactic features require robust and accurate Natural Language Processing (NLP) tools to perform analysis of text, thus the extraction of such

features is language-dependent procedure, depends on the efficiency of NLP tools.

The simple approach of syntactic features is to use Part of Speech Tags (POST), so that each word will be assigned a tag based on contextual information, then frequencies' for each tag are computed as features. This type of syntactic features provides only a hint of the structural analysis of sentences, since it is not clear how the words are combined to form phrases, or how the phrases are combined into higher-level structures.

(d) Semantic Features:

NLP tools can be applied successfully to low-level tasks, such as sentence splitting, POS tagging, text chunking, and partial parsing, so relevant features would be measured accurately and the noise in the corresponding data sets remains low. On the other hand, more complicated tasks such as semantic analysis can not yet be handled adequately by current NLP technology for unrestricted text. As a result very few attempts have been made to exploit high level features for stylometric purposes.

An important method [12] used semantic features, by estimating information about synonymous and hypernyms of the words, and identification of casual verbs, in order to detect semantic similarities between words. Also a more advanced approach tried to assign words, or phrases semantic information based on their meaning and indication.

It is clear that the semantic features require a more advanced NLP tools, where such tools are not available, and if some tools are developed, they are still not very efficient.

1.4.2 Attribution

Author attribution problem is a classification problem, since the goal is to assign author for an unknown document, so that there is a set of candidate authors, a set of text samples of known authorship covering all the candidate authors (training corpus), and a set of text samples of unknown authorship (test corpus), where each one of texts should be attributed to a candidate author.

The authorship attribution approaches can be classified according to whether they treat each training text individually or cumulatively (per author). The methods which treat documents cumulatively per author will produce a cumulative representation of that author's style based on his training documents. This is usually implemented using machine learning algorithms, which are trained using texts samples for each author to produce a classifier (maybe separate classifier for each author). Then an unknown text will be assigned to such a classifier to obtain the author of the test document. With the evolving of NLP tools, some few methods use the language models in author attribution problem, to produce a representation of author's style using author's training documents; such methods build an individual language model for each author from his training documents. In attribution process the test document will be assigned the author of the model that has the best ability to produce the test document.

On the other hand the methods that treat each training document individually, considers each author's document represents his style. So that a test document is compared with all training documents, to find the best matching one, then assigning to test document the author of training document, which most matches it. This approach is implemented based on Information retrieving method, by computing a score between the test document, and each training documents. Thus the best matching document is the one that has the highest score.

This classification of attribution process is not formal; it just illustrates the various attribution processes, since there are some author attribution approaches combine the two previous methods.

1.5 Research Overview

1.5.1 Objective

In our method we focus on author attribution problem, and the main goal is to classify the author of an unknown text accurately.

Due to the importance of author attribution problem, we need to find a new method that can capture the style of authors, to classify an unknown text. For critical applications, this method should gain a high accuracy. There are many solutions to author attribution problem, most of them follow the process of identifying set of features that considered most informative for authors styles, then a classifier is implemented based on those features in order to assign an author for a test document. Our method will use the language models [13] in order to assign author to a test document (of unknown author), this will done by improving one of the language models, so that it efficiently could reflect the most

informative features for author attribution problem, so that increasing the accuracy of the classification process.

1.5.2 Methodology

Researchers assume that all authors have specific style characteristics that are outside their conscious control; hence on the basis of those linguistic patterns and markers, the author of a document can be identified.

Our method starts by forming a language model for each author from his own training documents, this standard language model can effectively describe the language syntax for each author, such a syntax is considered as one of syntactic features, which is one type of features that can be used in author attribution problem, and proved to be informative about authors styles, however this type of features alone is not efficient to discriminate authors.

The proposed method tries to enhance the language model by involving more syntactic features than the language syntax, as Part of Speech Tagging (POST) feature, where each word will be assigned a tag, reflects its corresponding part of speech, such as noun, verb, etc. Plus this more lexical features will be added. Thus by adding this we will have an enhanced language model that contains rich features of different types, thus can be effectively used in author attribution problem. This enhanced language model is produced for each author using his own training documents.

When classifying an unknown document, the method starts to form an uncompleted language model to represent the test document, then matches this language model with each author's language model, so

that the test document will be assigned the author whose language model produces the best match with test document language model.

As we said the language model captures the language syntax of a document, this is done by producing a set of writing rules form that document, and since an author will have many documents, it is often to have a lot of such rules. The proposed method will automatically find the best of the writing rules, by assigning each rule a score based on its efficiency in discriminating authors.

1.5.3 Contribution

Our first contribution is to use language models to produce the features from author's documents, this is because the linguistic features are considered effective features in author attribution problem [14], also we will enhance the language model to capture more syntactic features, and lexical features. Incorporating more features produces a rich language model that can be used effectively in author attribution problem, while this process may produce unneeded or redundant information, so that another enhancement is to use scoring function which assigns a score for each rule produced by the language model for an author, the advantage of such function is to automatically finding the most informative rules that can be used to classify a test document, another advantage of the scoring function is that it will find the best rules without needing a prior knowledge of the written language, also no information about the structure of the language is needed to discover syntactic features as function words, they will be discovered automatically for any language.

So that the proposed method can be viewed as a language-free method that can be used with any language (the method works in Arabic language).

Because of the importance of lexical features [15], the method will involve some lexical features to enhance the language model, in order to increase the accuracy of the classification process, where an optimization method will be used to assign a best weight for each feature type to get a high accuracy. The advantage of such weights is to govern how each feature in the language model participates in classification process. Many researches were done in order to select best features for author attribution problem. Optimizing weights for each feature in classification can select the most efficient features for each author. Another advantage is that each author will be assigned his own weights for different features, so that there is a flexibility in selecting most efficient features, for example the method may optimize a high weight for lexical feature for a specific author, while for another author it optimize a high weight for syntactic feature, so that the method will not select a lexical feature, over syntactic feature or the opposite. It just finds the optimized features for each author, the advantage of this optimization process it the ability to cover more styles of authors.

1.5.4 Organization

The next chapter (Chapter 2) will view a related works for the author attribution problem, we will describe each work that was done in the field, focusing on the features that used in author attribution problem, the classification (attribution) method that used to assign author for a test document, the advantages and drawbacks of each work, and we

will also review the result for each work in order to compare them with our proposed method result.

Then we will overview some background theory in Chapter 3, where firstly we will explain the theory around the language models, especially the one that is used as basic in our method, then we will illustrate the scoring function that will be used to effectively find the best effective features in the language model. Also this chapter overviews some methods used in training and testing process.

Chapter 4 describes our proposed method in details, it starts with an overview of the whole method, supported by illustrative figures, and algorithms, then each step of producing the enhanced language model will be described in details, starting from parsing to produce the set of rules, and the method used to compute the probabilities of these rules, then describing the added syntactic, and lexical features, and how they are involved in the language model to produce a new language model for each author from his training document. The chapter also describes in details how a score is computed for each rule to indicate its efficiency. Also the classification process is included in this chapter, and it describes the process of assigning author for a test document. Finally the chapter illustrates assigning each type of rules a weight which reflects how each type of features participates in classification process.

Chapter 5 views the data set that used to test the proposed method in Chapter 4, and the experimentation that was done on this data, also it describes the metrics that are used to estimate the efficiency of the proposed method, ending with the results obtained with our method,

these results are described using several tables, and graphs, and compared with other methods which worked in author attribution field.

Finally the conclusion of the research was included in Chapter 6, which summarizes the research, remarks, and some notes around the work. Also a future work section was included to suggest some recommendations to be handled in future.

Chapter 2

RELATED WORKS

There are many researches were done in author attribution field, we are going to focus on them starting from traditional methods, which based on a machine learning algorithm with some selected features to classify a document. Because the set of features that can be used in author attribution is very large, feature selection methods will be proposed, and then in other section we will discuss some advanced methods used to solve the problem.

2.1 Machine Learning-Based Methods

Most of works, which have been done in this field, used a machine learning algorithm to discriminate the author of a given text using some set of features. There are many types of features -as we described before- that can be used in author attribution problem, we will overview some of researches that worked in author attribution field, focusing on the used features, classification method, and the results.

Carole [1] used a set of lexical features, as words frequency, text length, punctuation count, and average word length. Also he augmented the features with part of speech tagging (POST), which is a syntactic feature. This set of features was used to generate a linear discriminate function, which maximizes the difference between authors documents groups, so the coefficients of this function can be used to predict the group membership for a given test document. He considered 10 documents for each author, where each document is related to a predefined topic. The advantage of his method is that it achieved

accuracy of about 92%. However using lexical features can not efficiently describe author's style, even if it was augmented with a syntactic feature (POST), since POST is a simple syntactic feature that describes the type (syntax) of a single word, and can not reflect the syntax of a phrase.

Another traditional method was proposed by Nikos et al [16], they gathered a set of 85 features. The features are classified as follows: lemma-related features that capture the occurrence of specific word lemmas, these lemmas are selected as their low "order of occurrence" for at least one author, and high "order of occurrence" for at least one other author, also a new type of features is verbal features which captures how an author uses verb forms. They used POST feature which capture the frequency of occurrence of grammatical category of a word. They also implemented many lexical features to capture word length, sentence length, punctuation marks frequency, and the frequency of occurrence of the most common words expressing negation. These 85 features are supplied to three classifiers: the first classifier is a multi-layer perceptron network, the second is Radial Basis Function (RBF), which is a special type of neural network that uses radial basis function in hidden layer, and the last one is Self-organized map (SOM). They suggested that the accuracy depends on model deployment, i.e. the parameters that used to configure the classifiers, but in all classifiers accuracy did not exceed 85%. Using too many features can affect the performance of the classifiers, since they may contain unneeded information that can decreases classifiers performance, also the model depends on optimization the parameters to obtain a good result, usually estimating such parameters is complex, and needs more computation and optimization techniques.

Another method [15] applied the neural networks, and Tilburg in Memory Based Learner (TiMBL), which is a more advanced version of K Nearest Neighbor (KNN) algorithm, over a different set of features, some of the features are lexical features such as word length, *n*-grams, type-token ratio, hapax legomenon, and common word frequencies. The syntactic features are POST extracted for each token in the text, and the rewrite rules which detect some structure of a sentence such as subject, and objects. They used a shallow text analysis to extract the syntactic features. The best achieved accuracy from the two classifiers was about 72%. Even that the method combined lexical and syntactic features, it did not achieve good performance, the reason of that returned to the output of the shallow text analyzer since it detect some special type of words (as subject, and object), or because there are no optimization between lexical, and syntactic feature, so that poor features dominate the descriptive features.

Another machine learning method in the field of author attribution problem was proposed by Jochim et al [17]. They used a Support Vector Machine (SVM) classifier over a set of features extracted from various documents to identify the author of a given document. The point in their research is that SVM classifier can handle a very large set of features in a better way compared with other classifiers, but also the precision of their method ranged from 60-80%. The disadvantage of this method is the using of too many features with SVM classifier, since features have not same efficiency for author attribution problem.

Luyckx [18] also used the SVM classifier in author attribution, but instead of building a classifier for each author, he used a multi-class SVM, that can classify an author simultaneously. He used three types

of features, characters feature represented by character-level *n*-grams, lexical features represented by word *n*-grams, and functional words, and the syntactic feature represented by using POST. He suggested that the precision of such a classifier depends on its configuration; this is a disadvantage when using such a method, since adjusting parameter for a classifier is not a trivial problem, and requires complex estimations, but on the other hand the multi-class SVM can deal with small and large datasets very well.

Filiz et al [8] built and tested four different machine learning algorithms, each was supplied by a feature vector which is combined from *n*-grams and additional features, they used bi-gram (2-gram), and tri-gram (3-gram), counting the occurrence of each gram to be included in a feature vector. The additional features include statistical features as sentence length, word length, also they included vocabulary richness features, as type-token ratio, words occur once (hapax legomenon), and words occur twice (hapax dislegomenon), the feature vector also includes POST for each word in the text, and function words. We can note that this research (as many others) combines different types of features to help classifying a test document correctly, because of the high number of features they categorize features to a four sets, and test each set of features independently by applying the SVM, KNN, Random Forest, and multi layer perceptron classifiers. Each set of features obtains a different results on different classifier, but the overall result ranged from 60% – 84%. As we said before that using *n*-grams has two drawbacks, first there is a problem in defining the best value for *n*, and we can notice that this method tried to use multi values for *n*, in order to find the best solution, second drawback is that *n*-grams may capture content specific information, and we search for stylistic

information for author attribution problem. Also there is no difference between the used features, they equally participate in classification process.

The basic unit in traditional n-gram models is a word, Fuchon et al [10] proposed a new method based on character level *n*-gram model, in which the character is the basic unit, the details will be the same as "word based" *n*-gram model, they suggested that using a character level *n*-gram will discover useful inter-word, and inter-phrase features. The advantage of this method that it avoids the need of explicit word segmentation, so there is no need to parse sentences, and the method can be used to detect any language. The approach is to learn a separate language model (character level *n*-gram) for each author, which is trained from author's documents. In classification an unknown document will be supplied to each language model, to evaluate the likelihood, and pick the winning author. They evaluated the accuracy for three different languages data sets, and achieved a result between 70% and 90%. Character based *n*-gram model still inherits the problem of identifying the best value for *n*, also the representation of this model leads to high dimensionality space, which requires complex computations, and with the probability of capturing redundant information.

Another variation in using *n*-gram model was applied in [19], in which byte *n*-grams are used to build a language model for each author. Clearly to extract such grams the text is viewed as a sequence of bytes, then using this to build a profile for each author to be the set of most frequent *n*-grams, with their normalized frequencies generated from training documents. The classification will be the same as previous by

using the likelihood classification. Viewing text as a sequence of bytes ignores neither content specific information, nor stylistic based information, and can be considered less effective choice for the author attribution problem.

Ouamour et al [20] also built a classifier based on SVM algorithm. They used a Sequential Minimal Optimization method to speed up the training of the SVM. The algorithm was trained using several features; characters, character *n*-grams, words, word *n*-grams and rare words. The system was trained using only two Arabic documents for each author, and the testing was made using only one document. Using different combinations of features, the best achieved accuracy was 80%. This method used only lexical features in order to classify a document; this may describe the obtained result. Also the used data set is very small, which affected the result badly.

Another method that investigated the author attribution problem over the Arabic texts was proposed in [21]. They introduced a set of Arabic function words to be used as features in author attribution. This set of words was used by a hybrid classifier, which used an Evolutionary Algorithm, and a Linear Discriminant Analysis classifier, where the role of the Evolutionary Algorithm is to find a subset of the function words that are used to train the Linear Discriminant Analysis. The system did not exceed a 93% of accuracy. The drawback of the method is that it depended only on function words to discover authors, and these function words were identified to reflect the semantic of English function words from previous researches.

To achieve a good results, some methods tried to use different types of features to capture authors' style. Abbasi et al [22] used a set of 300

features of types; lexical, syntactic, structural, and content-specific features. The structural features measure the format of online texts written by authors, as font color, font size, embedded images, and hyperlinks. They tested these large set of features using SVM classifier, over online texts written both in Arabic, and English. The classifier reached 97% of accuracy when it was tested over English texts, while for Arabic texts it reached 94% of accuracy. Merging different types of features can effectively capture authors' styles, but the method did not perform well for Arabic texts, this may returned to the huge number of features that were used in classification, could not be discriminated in Arabic language.

Machine learning-based methods achieve acceptable results in author attribution problem, but we can notice that almost all methods did not benefit of efficient syntactic features as sentence structure, although this type of features considered to describe author's style, this maybe because of the hard implementation of such features in machine learning algorithms, since the set of such features is large. Even methods that combined syntactic features, with other features, assumed that all features have the same importance for author attribution problem.

2.2 Features Selection Methods

As you can see in the previous researches, there are many features that can be used in author attribution problem, this can be helpful, but in many cases the huge amount of features may decrease the performance of a classifier, in case of computing unimportant features. Because of this many researches were performed in order to study the best features that can be used in author attribution problem, one of these researches

was proposed by Jiexun et al [23], in which a Genetic algorithm [24] was used to identify the best features, where each gene represents a single feature with value 0 or 1 to indicates wither a feature is selected or not, the fitness function is defined as the accuracy of the corresponding classifier, they implemented an SVM algorithm to classify an unknown text, the algorithm shows that chosen 130 features from 270 can increase the results. The problem of this method is that one can not capture all stylometric features, they may be very large and require complex estimation to detect best features, also the system depends on a single classifier (SVM) to judge the importance of a feature, and last the syntactic features did not involved in the method, because it is hard to represent such features using Genetic Algorithms.

Another approach to select best features was proposed in [25], in which features will be selected according to their predictive values automatically, this value is calculated using chi square metric (X^2) which estimates the expected and observed frequency for every feature to identify features that are able to discriminate between authors, the algorithm uses combination of lexical features, plus syntactic features extracted by a parser to produce POST. In classification two different machine learning algorithms were used TiMBL, and SOM. The research did not compare the result of the classifier before and after using chi square metric over features.

2.3 Other Methods

Koppel et al [26] tried to work with a new approach depending on similarity measurements, rather than machine learning approach, they investigated the author attribution problem for large candidates (10,000 authors) using a similarity-based classification derived from

information retrieving theory. They represented the text as a vector that includes the frequencies of each 4-gram characters, including punctuation, numerals, and sundry, to find an author from large set of authors, they used a similarity based method; especially they use a cosine similarity [27], which is a common metric used in information retrieving methods. This similarity-based classification achieved a precision of about 46%, so they improved the procedure by repeatedly selecting top *k* documents, then computing the score for each author depending on the top *k* set, after some time the algorithm returns the author that has maximum score. Hence the idea is to check if a given author proves to be most similar to the test document for many different randomly selected feature sets of fixed size. The drawback of this method is to restrict the features on *n*-grams only, and as we said before *n*-grams can not capture the writing style. The vector that represents the text document must contain more descriptive features that can efficiently capture authors' styles.

In author attribution problem, the dataset is a set of text documents, this encourages the researchers to extract more than lexical features from these texts, so another approach was raised in author attribution problem by trying to incorporate language models in order to classify an unknown text; this approach assumes that each author has writing characteristics that can be captured using a language model. Sindh et al [14] tried to use a more advanced language model in author attribution problem, they applied the Probabilistic Context Free Grammar (PCFG) language model, by training a language model for each author from his known text documents, then for a test document they computed the likelihood for each language model related to each author, so that the test document will be assigned to the author whose language model

gives the highest likelihood score. The method achieved a good result in the range of 87-95%. PCFG language model, describes the structure of the sentences that are used in text, such a description is considered as a syntactic feature. Syntactic features obtain better result in author attribution field if it combined with another features, the research did not use any other features just depends on the syntactic features expressed by PCFG model, even with not using any other features the method obtained a good result, this is returned to the strength of PCFG model in expressing the structure of the sentences in the text, it seems that PCFG language model is a good descriptor of syntactic features.

Stochastic language models (as PCFG), contain a syntactic features, which can be used in author attribution, and even more, such models contain rich implicit lexical features, which can with the syntactic features efficiently capture authors styles, so that incorporating lexical features on the language models leads to a rich language model that can efficiently be used in author attribution problem.

Many of the previous works were tested over documents written in English language [8, 1]; some used Greek language [17,10], Belgian language [15], Germany language [17], and Arabic language [20-22]. In this thesis we will focus on applying an enhanced language model over Arabic texts, the language model will express the syntactic features in a more efficient way, plus providing more other features, to help solving the problem of detecting the author of an unknown document.

Chapter 3

THEORETICAL BACKGROUND

In this chapter we will overview the theory beyond some techniques that are used in the research. We will start describing the PCFG model, since that our enhanced method depends on it. Then an overview about the chi square metric will be proposed, since that the proposed method use the chi square function to discover the most efficient rules in PCFG language model. The chapter describes the Genetic Algorithm, and how it can be used to find a best solution for a specific problem. The Genetic algorithm is used to find the best weights between features in the enhanced language model. Finally we will overview the Leave-One-Out method, which is used to train and test the proposed language model.

3.1 Probabilistic Context Free Grammar

Context-free grammar (CFG) [13] is considered as the most effective grammar formalization for describing language syntax; it is adopted for language description. CFG is defined as a tuple $G= \{\Sigma, N, S, R\}$, where Σ is a set of terminal symbols which are symbols (words) actually seen in the sentences, N is a set of non-terminal symbols each of which points to further production rules, these two sets are disjoint, $S \in N$ is the start symbol, and R is a finite set of production rules that define how a string of terminal and non-terminal symbols can be immediately produced from a non-terminal symbol, it has the form $A \rightarrow \alpha$, where A is a non-terminal $\in N$, α is a sequence of terminal and non-terminal

symbols. So in CFG grammar a phrase can be viewed as a sequence of terminals.

All CFG rules contain only one symbol on the left hand side, each of which states that a given symbol can be replaced by a given sequence of symbols (on the right side), thus the “context” in which a symbol on the left hand side of a rule occurs is unimportant.

CFG provides a simple and mathematically precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the "block structure" of sentences in a natural way. CFG can exactly describe the basic recursive structure of sentences, the way in which clauses nest inside other clauses, and the way in which lists of adjectives and adverbs are swallowed by nouns and verbs. For example the sentence (تطلع الشمس من الشرق) is built from three smaller phrases blocks ((تطلع) (الشمس) (من الشرق)). This phrase structure is represented by production rules, where each production rule can be viewed in the form $non\text{-}terminal_0 \rightarrow non\text{-}terminal_1 \ldots non\text{-}terminal_n$, or $non\text{-}terminal_0 \rightarrow terminal_1$, Figure 3.1 shows the set of rules that represent the previous sentence, it is clear that the first rule represents that this sentence is a verbal phrase (*VP)*, where this verbal phrase consists of other three blocks, the first is a verb (*VBP)*, the second is a noun phrase (*NP)*, and the last is prepositional phrase (*PP)*, and so on every rule describes a smaller block, until describing the basic blocks in the sentence, which are the actual words.

$S \rightarrow VP$

$VP \rightarrow VBP\ NP\ PP$

$VBP \rightarrow$ تطلع

$NP \rightarrow DTNN$

$DTNN \rightarrow$ الشمس

$PP \rightarrow IN\ NP$

$IN \rightarrow$ من

$NP \rightarrow DTNN$

$DTNN \rightarrow$ الشرق

Figure 3.1: The rules generated from the sentence (تطلع الشمس من الشرق), where each rule describes a block structure of the sentence.

Another way to represent a sentence in CFG is to use a tree structure. A tree consists of labeled nodes with arcs indicates a parent-child relationship. Trees start from root node, and in CFG it always the *S* symbol, every node (except the root) has only one parent node, and zero or more child nodes. A node with no child is called leaf node.

Figure 3.2 shows a tree structure represents the grammar for the sentence (تطلع الشمس من الشرق), this tree captures the same structure captured by the production rules shown in Figure 3.1. Trees are more readable by humans than the production rules, since trees visualize the structure of a sentence, while capturing the same information about the grammar. Figure 3.2 shows the following structural analysis: there is a sentence *S* that is a verbal phrase (*VP)*, the sentence consists of (*VBP)* followed by (*NP)* and (*PP)*, the (*VBP)* (present verb) is the verb (تطلع), and the noun phrase (*NP)*, is a (*DTNN)* which is a noun (الشمس), the (*PP)* (propositional phrase) consists of *IN* (من), and *NP* which is a *DTNN* noun (الشرق).

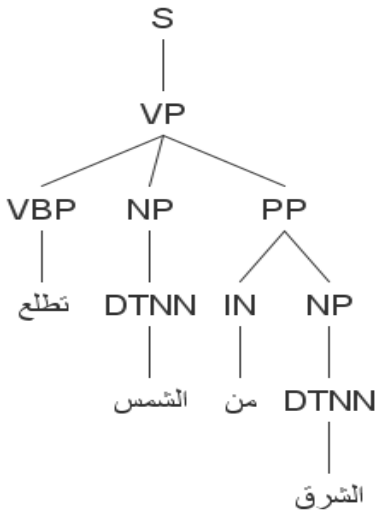

Figure 3.2: A tree representation of the sentence (تطلع الشمس من الشرق), where every node represent terminal, or non-terminal, and the arcs represent a parent-child relationship

CFG is considered a general formalization to describe language syntax, the most grammar formalization are derived from CFG, one of grammar formalization is the Probabilistic Context Free Grammar (PCFG), which is a probabilistic version of CFG in which each production rule is assigned a probability. These probabilities are required to sum up to 1.0 for each non-terminal, thus PCFG inherits all CFG's characteristics, and augments each rule with a probability, so we can view PCFG as a tuple $G = \{\Sigma, N, S, R, P\}$, where P is a list of probabilities, each probability is assigned to one of rules in R, and defines the likelihood with which this rule is used in generating a sentence. For the previous example (Figures 3.1, and 3.2), the probability for the first rule in Figure 3.1 ($S \rightarrow VP$) is one, while the probability for the rule ($DTNN \rightarrow$ الشمس) is 0.5 since there are two rules start by DTNN terminal. In practice the set of rules extracted from a

text will be large, so that the rules will have diversity in their probabilities values.

After parsing PCFG grammar *G*, and computing a probability for each rule from the training data. Probability of generating a string, given a grammar *G*, is the product of the probabilities of productions taken at each branch of its parsing tree; thus some derivations are more consistent with the grammar than others.

For example if we want to compute the probability of generating the sentence *S* = (تطلع الشمس من الشرق). Given that the production rules generated by parsing the sentence shown in Figure 3.3. Suppose then we have two PCFG models, each has its own probabilities for many rules, including the sentence rules as shown in Figure 3.3, then the probability of the model M_1 to generate the sentence *S* is given by:

$$P(S|\mathrm{M}_1) = 0.6 \times 0.25 \times 0.05 \times 0.3 \times 0.01 \times 0.7 \times 0.12 \times 0.4 \times 0.02 = 0.0000001512$$

And so the probability of the model M_2 to generate the sentence *S* is given by:

$$P(S|\mathrm{M}_2) = 0.45 \times 0.32 \times 0.07 \times 0.25 \times 0.04 \times 0.55 \times 0.21 \times 0.65 \times 0.01 = \ 0.0000000756756$$

It is clear that the language model M_1 has a high probability to produce the sentence *S*, than the second language model M_2, in other point of view the derivation of sentence *S* is more consistent with language model M_1, than M_2.

Table 3.1: The rules generated from the sentence S = (تطلع الشمس من الشرق), with probabilities for each rule in two different PCFG models ($M_{1, and}\ M_2$)

Production Rules of the Sentence S = (تطلع الشمس من الشرق)	PCFG Model M_1	PCFG Model M_2
$S \rightarrow VP$	0.6	0.45
$VP \rightarrow VBP\ NP\ PP$	0.25	0.32
$VBP \rightarrow$ تطلع	0.05	0.07
$NP \rightarrow DTNN$	0.3	0.25
$DTNN \rightarrow$ الشمس	0.01	0.04
$PP \rightarrow IN\ NP$	0.7	0.55
$IN \rightarrow$ من	0.12	0.21
$NP \rightarrow DTNN$	0.4	0.65
$DTNN \rightarrow$ الشرق	0.02	0.01

3.2 Chi-Square Feature Selection

A main problem in author attribution is the multiplicity of features, there are many features that can be used to identify the author, and because of this we need a powerful technique to select the best features, which are the most informative.

In our algorithm we will have many rules produced from authors' documents, we need the rules that are most efficient to discriminate authors, a popular feature selection method is chi-square (X^2). In statistics, the X^2 test is applied to test the independency of two events, where two events A and B are defined to be independent if $P(AB) = P(A)P(B)$ or, equivalently, $P(A|B) = P(A)$ *and* $P(B|A) = P(B)$ [27]. In classification problems we can view the two events as a feature t and a class c, so that the X^2 score measures the lack of independency between feature t and class c.

In our algorithm we will have a rule *r* and a class *c*, and we want to know the dependency of each rule and the class (author).

One way to compute X^2 is by using the two-way contingency table [28] of a rule *r* and a author *c*, the X^2 score between rule *r*, and author *c*, is defined to be:

$$X^2(r,c) = \frac{N\ (AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \qquad \ldots\ldots\ldots\ldots\ (3.1)$$

where *A* is the number of times *r* and *c* co-occurs, *B* is the number of times the rule *r* occurs without *c*, *C* is the number of times *c* occurs without *r*, *D* is the number of times neither *c* nor *r* occurs, and *N* is the total number of documents

If rule *r* is independent of author *c*, then the X^2 score will be zero. The computation of X^2 scores has a quadratic complexity, similar to mutual information (MI), and information gain (IG). A major difference between X^2 and MI is that X^2 is a normalized value; hence X^2 values are comparable across terms for same category.

By using X^2 we will have a score for every author's rule, the score will denote the relation of the rule and the corresponding author, so that a rule with small value score denotes that the rule is general and not a discriminate rule for that author, while a high value score denotes that the rule is specific rule that captures the authors style.

3.3 Genetic Algorithm

Genetic algorithm [24] is a machine learning algorithm that mimics the process of natural evolution. It is a search algorithm that routinely used to generate useful solutions to optimization a search problems.

It searches for best solution (hypothesis) from a set of candidate solutions (hypotheses); the best solution is the one that optimizes a predefined numerical function called fitness function.

A hypothesis is encoded as a chromosome (called individual), which reflects a candidate solution to an optimization problem, the chromosomes evolved toward better hypotheses. The evolution occurred in generations starting from a population of randomly generated chromosomes, where in each generation the fitness of every chromosome in the population is evaluated, so that some chromosomes are selected, and modified to form a new population, which will be used in the next iteration of the algorithm. The algorithm will continue following this process, until a termination criterion has occurred. Usually the termination condition is reaching a satisfactory fitness value, in other problems the algorithm may terminates due to reaching a maximum number of generations.

A simple Genetic algorithm is viewed by Algorithm 3.1, where in first step the algorithm choose an initial population of individuals, these individuals are usually randomly generated, the population size (number of chromosomes in population) is an application dependent value, but typically a population contains several hundreds or thousands of individuals. Then fitness value is evaluated for each chromosome in the population.

The next step tries to choose some individual from the current population based on their fitness value, then based on the selected individuals the algorithm generates a new population of individuals through genetic operators. The two most common operations used by genetic algorithms, are crossover and mutation.

Purpose: Finding best solution for a specific problem based on fitness function Input: Fitness function that judge the efficiency of each individual Output: An individual that represent a solution to the problem. Procedure:	
1	Begin
2	Choose the initial population of individuals
3	Evaluate the fitness of each individual in that population
4	Repeat on this generation until termination
5	Select the best-fit individuals for reproduction
6	Create new individuals using Genetic operators
7	Evaluate the individual fitness of new individuals
8	Replace least-fit population with new individuals
9	Return the best-fit individual as a solution.
10	End

Figure 3.3: Simple Genetic Algorithm procedure

Crossover operation takes two solutions (parents), and produces two new individuals from it, by copying characteristics from the parent individuals, while the mutation operation produces small random changes to individual by choosing a single characteristic at random and change its value. Mutation is often performed after crossover. Following this process a new solution that shares many characteristic of its parents is created. For each new individual, new parents are selected. The process continues until a new population of solutions of appropriate size is generated. This generational process is repeated until a termination condition has been reached.

3.4 Leave-One-Out Method

A machine learning system depends on data samples, and usually these samples are categorized in two sets, the first is the training samples set,

which is used by the system in learning phase, while the other is testing samples set, which will be used to validate the system.

It is known that training set must be of a reasonable size, so that it can produce a representative sample of the true target function [24]. But in some systems the training samples may be not available in the required size.

The problem of lacking in data samples is addressed by cross-validation method, which starts by partitioning data into complementary subsets, performing the analysis (learning) on the training set, and validating the analysis on the testing set. Then multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

The purpose of such process is to overcome the problem of overfitting. A learning algorithm is said to be overfited if it is more accurate in fitting known data, but less accurate in predicting new data. The purpose of such process is to predict the fit of a model to validation set when an explicit validation set is not available. Overfitting is particularly likely to happen when the size of the training data set is small.

There are many types of cross validation method, one is the Leave-One-Out method [29], where for a data set (training and testing) of size N, the training is performed using N-1 samples, and the test is performed using the excluded sample. If this is misclassified, an error is counted.

This process is repeated N times, each time excluding a different sample. The total number of errors leads to the estimation of the classification error probability.

So that training is achieved using, basically, all samples, and at the same time independence between training and test sets is maintained (because data used in training are not used in testing). But the disadvantage of the Leave-One-Out method is it needs many computations, which leads to high computational complexity.

Chapter 4

PROPOSED METHOD

4.1 Overview

The PCFG grammar describes the language syntax, by capturing the structure of sentences, but this alone can not be used to distinguish the author of an unknown text [14], since it focuses on grammar rules and their probabilities only, so our contribution is to extend the PCFG language model in order to capture additional features that can increase the efficiency of language models in author attribution problem, as we saw in all previous works, they used some basic features as lexical features to distinguish between authors [1,10], these features proved to be informative [15], we will try to involve these features in PCFG language model.

Recall that we already know the words in each sentence (terminals) and their types (non-terminals) from the PCFG language model, using this we will capture some lexical features, this will be handled by adding a new set P_T in the grammar G, which contains the probabilities for each terminal.

Also a second set P_N will be added to the grammar G, which indicates the probability for each non-terminal, for example the probability of verb will be $P(V)$, this set will be fixed size since that the set of non-terminals is predefined, PCFG will not consider any punctuation marks in estimation rules or probabilities, however these are considered as major feature to capture the style of a text [30], so we will add a new

third set P_U to grammar G, which indicates the probability of each punctuation mark.

The other extension to PCFG model, is to compute weights for each rule probability in set R, these weights will be computed using chi-square score (X^2). So the extended weighted PCFG model (we will call it XPCFG) tuple will be:

$$G = \{\Sigma, N, S, R, P, U, X^2, P_T, P_N, P_U\} \quad \ldots\ldots\ldots\ldots \quad (4.1)$$

where X^2 is the set of weights for each rule in R, P_T is probabilities for each terminal $\epsilon\ \Sigma$, P_N is the probabilities of each non-terminal $\epsilon\ N$, U is the set of punctuation marks, and P_U is probabilities for each punctuation mark $\epsilon\ U$.

Our algorithm will generate an XPCFG model for each author, starting from set of training documents for each author, and tries to extract the grammar from theses documents for each author; this is done using parsing application. After the grammar has been formalized by generating production rules for each author, a probability will be computed for each production rule, to build a complete PCFG grammar for each author, then a score will be computed for each generated rule to compute the dependency between this rule and its corresponding author, this will be accomplished by computing X^2 score for each rule, hence producing a full weighted PCFG grammar. Also the probabilities of terminals, non-terminals, and punctuation marks will be computed in this step to produce the new XPCFG language model for each author from his training documents. Figure 4.1 illustrates the process of generating author's XPCFG language model, where the input is a set of

training documents, which are parsed to generate set of production rules as in Figure 3.1, these rules contain the information about terminal, non-terminals, and punctuation marks, so that the next step is to compute the probabilities for the rules, and the other features (we called them lexical features), you can note that no extra tools is needed to extract lexical features, because the information is contained in production rules, so we only make use of them by computing probabilities for lexical features. As we said the chi-square score is computed for each production rule.

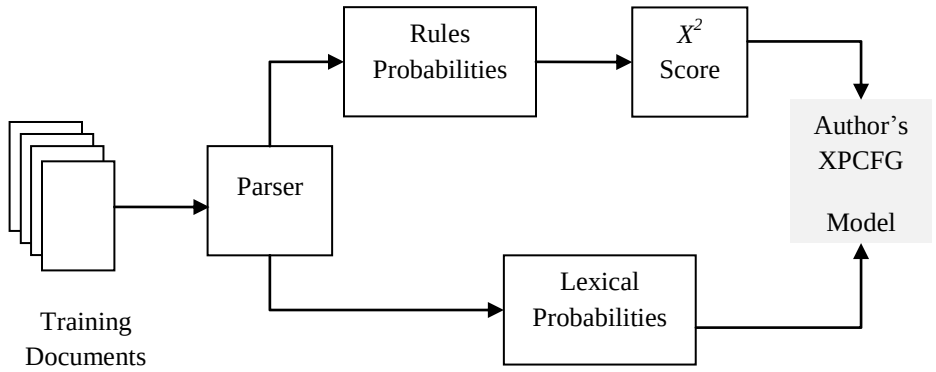

Figure 4.1: The process of generating the enhanced language model XPCFG for a specific author, using a set of training documents that are belongs to the author.

In Figure 4.2 we illustrate the process of finding best weights for the different features. Because we use different types of features, grammatical features represented by PCFG rules, and lexical features represented by probabilities of terminals, non-terminals, and punctuation marks (actually non-terminals is considered a syntactic feature), so that a Genetic algorithm will be used to find the best weights for the lexical features and grammatical features, as shown in Figure 4.2, the algorithm uses a new corpus called Held-out data set, it is used only for the purpose of finding the best weights between

different features for a specific author. The inputs for Genetic algorithm are the Held-out data set, author's XPCFG model, and a fitness function, the output is the best weights between features for that author, the values of weights depends on maximizing the classification accuracy for documents in Held-out set.

Finally in the classification process (shown in Figure 4.3), a test document is passed to classifier, with all authors' models, and optimum weights for each author. The classifier estimates a score between the test document and each author's language model, and by using weights found by Genetic algorithm, so that the test document is assigned to author who has the maximum score.

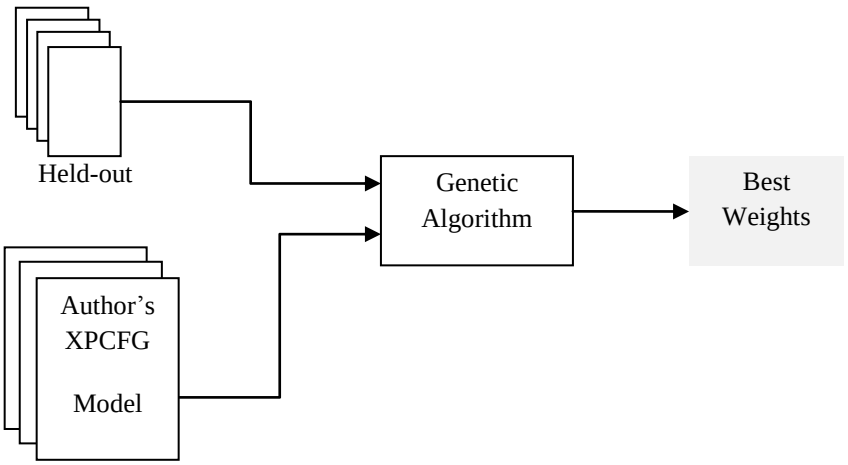

Figure 4.2: Estimation of optimum weights between different features in XPCFG model for a specific author.

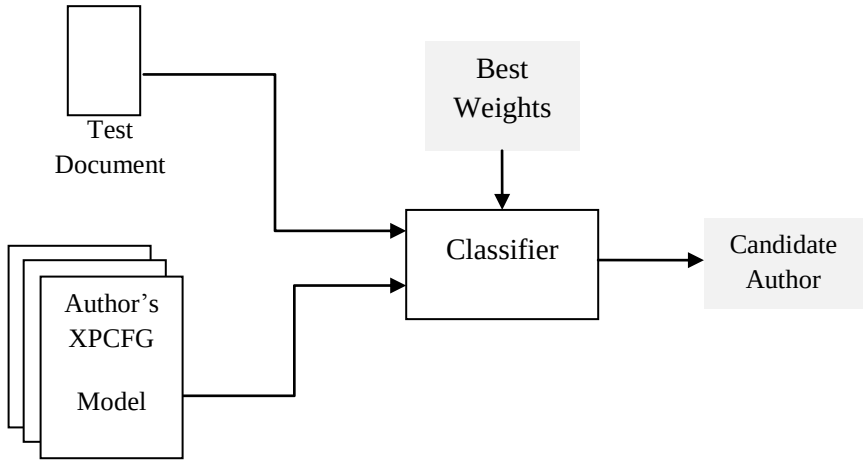

Figure 4.3: Classification a test document, the classifier inputs are a test document, authors XPCFG models, and weights for all authors.

4.2 Parsing

As shown in Figure 4.1 the first step in training an XPCFG language model for a specific author is parsing his training documents. Parsing is the process of analyzing a text, made of a sequence of tokens (words), to determine its grammatical structure with respect to a given formal grammar [31], so that any document in training, testing, or Held-out corpuses is parsed before it can be used.

We use a probabilistic parser (also called statistical parser), which is a parser that uses knowledge of language gained from previously hand-parsed sentences, so that to produce the most likely analysis of new sentences. The result of the parsing process is a set of grammatical rules, as shown previously in Figure 3.1.

4.3 Training

In training phase as shown in Figure 4.4 we attend to produce a full XPCFG language model for each author, the language model includes the PCFG rules (produced in parsing phase), with their probabilities

Purpose: Produce a complete XPCFG model for a specific author from his training documents Input: Training document for a specific author Output: Complete XPCFG language model for the author Procedure:	
1	Begin
2	For each training document *j*
3	Parse document d_j to generate the set of rules *R*
4	Compute rules probabilities *P*
5	Compute terminals probabilities P_T
6	Compute non-terminals probabilities P_N
7	Compute punctuation marks probabilities P_U
8	End loop
9	Compute the average probabilities for P, P_T, P_N, and P_U , over all training documents
10	Compute the X^2 score for each rule in *R*
11	Return XPCFG = $\{\Sigma, N, S, R, P, U, X^2, P_T, P_N, P_U\}$
12	End

Figure 4.4: Training XPCFG language model

and scores, and three lists of terminals, non-terminals, and punctuation marks that used in training documents, with their probabilities. The following sections overview the training steps in details.

4.3.1 PCFG Rules Probabilities

After parsing each document in the training data set, and producing the rules, we want to compute a probability for each rule to produce a PCFG grammar. This is shown in Algorithm 4.1 step 4. Remember that a separate language model will be generated for each author, so that for each author's document we will find the probability for each rule that appears in the document, then producing a complete grammar by averaging the probabilities for each rule. Starting from first document for a specific author, we compute the probability for each rule in each

training document which belongs to the corresponding author by counting. For example let the probability for rule $r \rightarrow a\ b$ is given by $p(r \rightarrow a\ b)$, then this probability will be computed using the following equation:

$$p(r \rightarrow a\ b) = \frac{count(\ r \rightarrow a\ b\ )}{count(\ r \rightarrow anything\ )} \quad \ldots\ldots\ldots\ldots\ (4.2)$$

After computing the probability for each rule in each training document for a specific author, we will generate a PCFG language grammar for that author. The set of rules will be gathered from each training document, and the probability for each rule will be the average probability for the rule in all training documents (Step 9 in Algorithm 4.1). Given n is the number of training documents for a specific author, and $p_i(\ r \rightarrow a\ b\ )$ is the probability of rule $r \rightarrow a\ b$ in i^{th} training document for that author, then the average probability $p\hat{}(r \rightarrow a\ b)$ is given by:

$$p\hat{}(r \rightarrow a\ b) = \frac{\sum_{i=1}^{n} p_i\ (\ r \rightarrow a\ b\ )}{n} \quad \ldots\ldots\ldots\ldots\ (4.3)$$

Following this procedure each author language model will contains non duplicated rules, i.e. a rule will be found only once in author's language model, and has a probability reflects the averaged probability of that rule in all training documents that belong to the corresponding author.

Recall that when rules are generated from training documents by the parser, the right side of a rule contains either non-terminals, or terminals, clearly the first set of rules will be non-leaf rules in parsing tree, while the other set (terminals rules) will be leaf rules in the parsing tree (see Figure 3.2), in a traditional PCFG language model

there is no difference when dealing with these sets of rules, but in our proposed model (XPCFG), we separate the PCFG rules in two different sets, the first is non-terminal rules, and the other is the terminal rules, for example in Figures 3.1 and 3.2 the rules $S \rightarrow VP$, $VP \rightarrow VBP\ NP\ PP$, $NP \rightarrow DTNN$, $PP \rightarrow IN\ NP$, and $NP \rightarrow DTNN$ are categorized as non-terminal rules, while the rules $VBP \rightarrow$ تطلع, $DTNN \rightarrow$ الشمس, $IN \rightarrow$ من, and $DTNN \rightarrow$ الشرق are categorized as terminal rules. Returning to Equation 4.1 the set R will be divided to two sets, and so the probabilities P. The goal of this categorization of the rules is to measure the efficiency of each set of rules (non-terminal and terminal rules) in the classification process.

Note that for the simplicity of notation we did not mention this categorization of rules in the previous equation, but you will see the purpose of this method in Section 4.5.

4.3.2 Non-terminals Probabilities

Non-terminals are considered as one of effective syntactic features in author attribution problem [15], so that we extend the PCFG model by incorporate non-terminals behind sentences structure, to allow the model capturing more detailed syntactic features, this is done by adding a new list to language model so that capture the probabilities of non-terminals.

Note that the set of non-terminals N is a fixed size set, since non-terminals are predefined by the parser, so that for each author's language model the set N will contain the same non-terminals, but with different probabilities. To compute probabilities of non-terminals P_N for a specific author, we use the PCFG rules which are already have

been generated, and their probabilities were computed (Section 4.3.1), we here focus on the rules that contain non-terminals, so that for a specific author a non-terminal probability will be computed for each author's training document, then a final averaged probability will be recorded in P_N list in the XPCFG language model. Finding the probability of a non-terminal in one of training documents, is done by scanning the PCFG rules produced from this training document, then counting the occurrence of this non-terminal in the rules, this process is the step 6 in Algorithm 4.1. So for a non-terminal *nt*, in one of training documents (belongs to specific author), that contains *m* non-terminals in all rules, the probability of *nt* is given by:

$$p(nt) = \frac{count(\ nt\ )}{m} \qquad \ldots\ldots\ldots\ldots\ (4.4)$$

Note that *m* is the total number of non-terminals in a specific training document, so that when we said the non-terminals are fixed, we mean they are known, but how many they appear in a document is captured by *m*, and depends on the size and the grammar used in writing this document.

Equation 4.4 is used to compute the probability for each non-terminal in a single training document, following this process for other training documents that belong to same author, will produce a set of probabilities for each single non-terminal, and for a specific author we want a final probability, which will be computed by averaging the probabilities of that non-terminal over author's training documents (Step 9 in Algorithm 4.1), as in Equation 4.3 the average of a non-terminal *nt* is the sum of this non-terminal probabilities in training documents, divided by number of documents.

4.3.3 Terminals Probabilities

In PCFG model the set of terminals Σ contains the actual words used by a specific author, it is clear that words used by authors are a topic dependent feature, and this type of features can not alone successfully classify an unknown text, but the purpose from using terminals is to find the function words, which are considered as an effective feature in author attribution problem [32], as we later defined the functional words as words that have little lexical meaning but server to express grammatical relationships with other words, there are many researches [33] tried to find these words, unfortunately all of them works in English language, and as we focus here on author attribution of Arabic texts, so that by using terminals we try to automatically find functional words. Returning to computing terminals probabilities, we follow the process used when computing the non-terminals probabilities in Section 4.3.2, as we previously started from PCFG rules to find non-terminals, we also here start from the set of PCFG rules, that are generated from a training document for a specific author, using these rules we can simply count each terminal occurrences, and dividing it by total number of terminals in this training document, note here the number of terminals is not fixed, it depends on the size of training document.

As we did before a final averaged probability of a terminal will be computed by averaging the probabilities of this terminal among the training documents, by this we will have a complete terminals probabilities list P_T produced for each author from his own training documents

4.3.4 Punctuation Marks Probabilities

Many researches were done in author attribution field proved that punctuation marks can effectively discriminate authors [30], because of this a new extension to PCFG language model is done, by adding a list of punctuation marks probabilities. Punctuation marks also generated by PCFG model rules, and considered as terminals, we decide to make a new probabilities list for punctuation marks for two reasons, first we want to find how they can affect the classifier, and the other reason because we want to find a reliable probability of each punctuation mark in a training document, i.e. if punctuation marks is treated as terminals then the probability of a punctuation mark, will be its occurrence in the document divided by the total number of terminals in this document, which will give a very small probabilities for punctuation marks since they occur not very much, and the probability of terminals will overwhelm them, so the reliable probability of a punctuation mark in a training document is the occurrence of this punctuation mark, divided by the total number of punctuation marks in the document.

After computing the probabilities of each punctuation mark in training documents (belong to specific author), a final probability is computed by averaging theses probabilities, so that the list P_U will contain the probabilities for punctuation marks used by a specific author, and extracted form his own training documents.

4.3.5 Chi-Square Score (X^2)

After computing the average probability of each rule extracted from training documents, the proposed algorithm in Algorithm 4.1, computes a score for each rule (Step 10), each score measures the dependency between a rule and its corresponding author, in other word the score

function will discover the most effective rules, because there are some rules considered general rules, as they appears in all authors texts, and can not be considered a fingerprint for author's style, in other hand there are rules that will get a high score because they are capture specific style for different authors, the computation of score is done using Equation 3.1, from the equation we can note that the score will be high if a rule occurs fewer times in training documents, thus the rule is a highly discriminate rule, and can capture specific writing style for the corresponding author, in the opposite side if a rule occurs in virtually all training documents, then it will be assigned a low value, thus it is a general writing rule and does not capture specific writing style for authors.

The benefit of finding a score for each rule in the XPCFG model, is that the model automatically finds the best rules that can discriminate authors, and without knowing the structure of the written language or the language grammar, so that the XPCFG is a language-free model, that has the ability to find the author's style without knowing the structure of the language used by authors to write documents.

So by computing chi-square score for each rule, we will produce the list X^2 in grammar *G*, which contains the score for each rule in *R*. Thus we have now formed the enhanced PCFG model described by Equation 4.1, $G = \{\Sigma, N, S, R, P, U, X^2, P_T, P_N, P_U\}$. Remember that the language model will be formed for each author, from his own training documents, doing this we can view our algorithm as a profile based attribution method, since each author has its own language model.

4.4 Classification

Since we have trained a separate language model for each author, by computing probabilities for rules, terminals, non-terminals, and punctuation marks, we will use a probabilistic classifier to assign author for an anonymous text. Such a classifier attempts to maximize the probability *P(x|a)* for a text *x* to belong to a candidate author *a*. Using Bayes rule [34] :

$$P(x|a) = \frac{P(a|x)\,P(a)}{P(x)} \qquad \ldots\ldots\ldots\ldots\ (4.5)$$

P(x) is the same for the test document, and so can be ignored. The prior probability of an author *P(a)* is often treated as uniform across all authors and so it can also be ignored, so that we can estimate the probability of a test document *x* by finding the probability $P(a|x)$

A test document *x* can be viewed as a sequence of $x_1, x_2, \ldots, x_n$of *n* independent and identically distributed observations, where the observations are the rules, terminals, non-terminals, and punctuation marks in the test document, to simplify the description here we will talk about all observations as one type, then we will describe the details. By using maximum likelihood [34], first we must specify the probability joint density for test document *x*, by

$$P(x_1, x_2, \ldots, x_n|a) = P(x_1|a)\,P(x_2|a)\ldots..P(x_n|a) \quad \ldots\ldots\ldots\ (4.6)$$

Then the likelihood function is

$$\mathcal{L}(a|x_1, x_2, \ldots, x_n) = P(x_1, x_2, \ldots, x_n|a) = \prod_{i=1}^{n} P(x_i|a) \quad \ldots\ldots\ldots\ (4.7)$$

Equation 4.7 estimates how an author *a* is likely produces a test document *x*, using the probabilities computed in author's XPCFG language model, since the probabilities are values between 0 and 1, multiplying them in this way will produce a very small numbers, which can not be handled by computers, thus it is more convenient to work with natural logarithm of the likelihood function, called the log-likelihood, because the logarithm is a monotonically increasing function achieves its maximum value at the same points as the likelihood function.

The likelihood function is factored into a product of individual probabilities. The logarithm of this product is the sum of individual logarithms, and estimating the summation of terms is easier than estimating the product of terms.

The likelihood score is a value between 0 and 1, so it is a probability, but the log-likelihood is going to be negative value. So the log-likelihood function will be:

$$\hat{\mathcal{L}}(a|x) = \sum_{i=1}^{n} log\, P(x_i|a) \qquad \ldots\ldots\ldots\ldots\ (4.8)$$

To classify a test document *x* we use Equation 4.8, to compute the log-likelihood ($\hat{\mathcal{L}}$) between the test document and every author's language model, then we assign the test document to author that has the highest log-likelihood. To do this we first will parse the test document to produce XPCFG language model, which will contain rules *R*, the set of terminals *Σ*, the set of non-terminals *N*, and the set of punctuation marks *U*, but with no probabilities for these sets.

In Equation 4.6 we view *x* as set of features $x_1, x_2, \dots, x_n$, but since we have four types of features in *x*, we can view test document as four sequences of observations, where the first sequence reflects the rules in *x* and denoted by $r_1, r_2, \dots, r_A$, and sequences $t_1, t_2, \dots, t_B$, $nt_1, nt_2, \dots, nt_C$, $u_1, u_2, \dots, u_D$, is terminals sequence of size *B*, non-terminals sequence of size *C*, and punctuations marks of size *D* respectively, so that ($n = A + B + C + D$). The classifier uses Equation 4.8 to compute $\hat{\mathcal{L}}$ for each sequence, for example the log-likelihood for terminals $\hat{\mathcal{L}}_T$ is given by:

$$\hat{\mathcal{L}}_T\,(a|x) = \sum_{i=1,\ t\epsilon x}^{B} log\, P(t_i|a) \qquad \dots\dots\dots\dots\dots\dots \quad (4.9)$$

In the same manner the previous equation is used to estimate the log-likelihood for non-terminals $\hat{\mathcal{L}}_N$, punctuation marks $\hat{\mathcal{L}}_U$, and rules $\hat{\mathcal{L}}_R$. One variation in computing $\hat{\mathcal{L}}_R$ is that the classifier incorporates the X^2 score for each rule in the log-likelihood $\hat{\mathcal{L}}_R$, this is done by simply multiplying each rule with its corresponding X^2 score (computed in training phase), so that a log-likelihood for rules is given by

$$\hat{\mathcal{L}}_R\,(a|x) = \sum_{i=1,\ r\epsilon x}^{A} X^2(r_i|a) \times log\, P(r_i|a) \qquad \dots\dots\dots\dots\dots\dots \quad (4.10)$$

Where $X^2(r_i|a)$, is the chi-square score for i^{th} rule in the XPCFG language model for author *a*. Remember that in training phase we distinguish between two types of PCFG rules, we defined two different sets of rules, the first contains the terminal rules, while the other contains the non-terminal rules, so a two different log-likelihood

estimation for rules is considered, and the sequence $r_1, r_2, \ldots, r_A$ is viewed as two sequences, the log-likelihood of rules is given by $\hat{\mathcal{L}}_{R1}$, and $\hat{\mathcal{L}}_{R2}$, where the first is the log-likelihood of terminal rules, and the second is the log-likelihood of non-terminal rules.

Putting all together, the classifier computes the final log-likelihood $\ddot{\mathcal{L}}$ between test document *x* which is represented as sequence of observations, and author *a*, using the following equation:

$$\ddot{\mathcal{L}}(a|x) = \hat{\mathcal{L}}_{R1}(a|x) + \hat{\mathcal{L}}_{R2}(a|x) + \hat{\mathcal{L}}_{T}(a|x) + \hat{\mathcal{L}}_{N}(a|x) + \hat{\mathcal{L}}_{U}(a|x) \quad \ldots\ldots (4.11)$$

Equation 4.11 suggests that each type of XPCFG model, participates in classification with equal weights, we can enhance classification process by assigning each part in Equation 4.11, a different weight which will govern how much each part of XPCFG model (terminal rules, non-terminal rules, terminals, non-terminals, and punctuation marks) participates in classification process, to implement this we can rewrite Equation 4.11 as follows:

$$\ddot{\mathcal{L}}(a|x) = w_{R1}\hat{\mathcal{L}}_{R1}(a|x) + w_{R2}\hat{\mathcal{L}}_{R2}(a|x) + w_{T}\hat{\mathcal{L}}_{T}(a|x) + w_{N}\hat{\mathcal{L}}_{N}(a|x) + w_{U}\hat{\mathcal{L}}_{U}(a|x) \quad \ldots\ldots\ldots (4.12)$$

Weights values are between 0 and 1 and all sums to one, one can test the classifier by assigning all weights 0.2 value so we will have a same result as Equation 4.11, these values can be set manually for testing, but we use a machine learning algorithm to find the best values for weights,

so that maximizing the efficiency of the classifier, this will described in next section.

The classifier computes the log-likelihood (Equation 4.12) for a test document, and each candidate author's language model, assigning the document the author whose language model generates the highest value. Formally speaking for a test document *x*, the classifier estimates $\ddot{\mathcal{L}}(a|x)$ in Equation 4.12, between the document *x*, and all available authors, using their XPCFG language models, then assigning the test document the author that has the maximum $\ddot{\mathcal{L}}(a|x)$ value, so that the candidate author $\hat{a}$ for an anonymous text *x*, is the one that has the maximum $\ddot{\mathcal{L}}(a|x)$, as shown in next equation

$$\hat{a} = arg\max_{a} \ddot{\mathcal{L}}(a|x) \quad \ldots\ldots\ldots\ldots \quad (4.13)$$

To illustrate the classification process, suppose that we have two authors, a_1 and a_2, each was trained by following the previous process, so that probabilities are computed for each rule, non-terminals, terminals, and punctuation marks for each author. Suppose that we have a test document *x*, contains only a single sentence (تطلع الشمس من الشرق), and was parsed into rules as shown previously in Figure 3.1. In classification we want to find which author (a_1 or a_2) writes the document. The probabilities for each author are shown in Table 4.1, together with the chi-square for each PCFG rule. So that to classify the test document *x*, we will compute a final log-likelihood (Equation 4.11) for the test document and the two authors, and to find the final log-likelihood, we need to compute the log-likelihood for terminal rules, non-terminal rules, terminals, non-terminals, and punctuation marks (note in this example there are no punctuation marks, so the log-

likelihood for punctuation marks is ignored), where each one is

Table 4.1: Probabilities for non-terminal rules, terminal rules, non-terminals, and terminals for two different authors, with chi-square score for PCFG rules for the two authors.

Training Document x		Author a_1		Author a_2	
		Probability	X^2	Probability	X^2
1	$S \rightarrow VP$	0.15	0.5	0.24	0.05
2	$VP \rightarrow VBP\ NP\ PP$	0.25	1	0.31	9
3	$NP \rightarrow DTNN$	0.29	10	0.52	12
4	$PP \rightarrow IN\ NP$	0.41	2.7	0.35	5.1
5	$NP \rightarrow DTNN$	0.12	0.3	0.26	2.5
6	$VBP \rightarrow$ تطلع	0.18	0.91	0.26	1.6
7	$DTNN \rightarrow$ الشمس	0.35	4	0.31	4.1
8	$IN \rightarrow$ من	0.37	0.1	0.21	0.36
9	$DTNN \rightarrow$ الشرق	0.05	3.8	0.12	5.1
10	VP	0.03		0.21	
11	VBP	0.09		0.1	
12	NP	0.02		0.05	
13	PP	0.04		0.09	
14	$DTNN$	0.02		0.06	
15	IN	0.36		0.41	
16	تطلع	0.04		0.15	
17	الشمس	0.06		0.18	
18	من	0.1		0.2	
19	الشرق	0.01		0.05	

computed using Equation 4.9, for example the log-likelihood for non-terminals and author a_1 will be:

$$\hat{\mathcal{L}}_N(a_1|x) = \log\big(P(VP|a_1)\big) + \log\big(P(VBP|a_1)\big) + \log\big(P(NP|a_1)\big) + \log\big(P(PP|a_1)\big) + \log\big(P(DTNN|a_1)\big) + \log\big(P(IN|a_1)\big)$$

$$\hat{\mathcal{L}}_N(a_1|x) = \log(0.03) + \log(0.09) + \log(0.02) + \log(0.04) + \log(0.02) + \log(0.36)$$

$$\hat{\mathcal{L}}_N(a_1|x) = -3.51 - 2.41 - 3.91 - 3.22 - 3.91 - 1.02 = -17.98$$

Also the log-likelihood for terminals and author a_1 will be:

$$\hat{\mathcal{L}}_T(a_1|x) = \log(0.04) + \log(0.06) + \log(0.1) + \log(0.01)$$
$$\hat{\mathcal{L}}_T(a_1|x) = -3.22 + -2.81 + -2.30 + -4.61 = -12.94$$

The one variation in computing the log-likelihood for PCFG rules is incorporating the X^2 score for these rules, as described by Equation 4.10, so that the log-likelihood for non-terminal rules is:

$$\hat{\mathcal{L}}_{R1}(a_1|x) = (0.5 \times \log(0.15)) + (1 \times \log(0.25)) + (10 \times \log(0.29)) + (2.7 \times \log(0.41)) + (0.3 \times \log(0.12)) = -17.76$$

And so for terminal rules, the log-likelihood is:

$$\hat{\mathcal{L}}_{R2}(a_1|x) = (0.91 \times \log(0.18)) + (4 \times \log(0.35)) + (0.1 \times \log(0.37)) + (3.8 \times \log(0.05)) = -17.24$$

So that the final log-likelihood (Equation 4.11) for author a_1, is the sum of all previous values, and will be:

$$\ddot{\mathcal{L}}(a_1|x) = -65.92$$

Following the same computations for author a_2, we will get the following result:

$$\hat{\mathcal{L}}_N(a_2|x) = -12.97$$
$$\hat{\mathcal{L}}_T(a_2|x) = -8.22$$
$$\hat{\mathcal{L}}_{R1}(a_2|x) = -27.18$$
$$\hat{\mathcal{L}}_{R2}(a_2|x) = -18.33$$

So the final log-likelihood for author a_2, will be:

$$\ddot{\mathcal{L}}(a_2|x) = -66.70$$

According to Equation 4.13 the test document *x* will be assigned the author that has the maximum value for the final log-likelihood, which in our example is author a_1.

4.5 Computing Optimum Weights

As the classifier assigns author to a test document based in Equation 4.12, we attend to find the optimum values of the weights, so that the classifier's accuracy is maximized as possible, such a problem is a traditional search problem in which a search algorithm tries to find the best solution among many candidate solutions. There are many machine learning algorithms can be used to solve such a problem, one is the Genetic Algorithm described in Section 3.4. In Genetic algorithm each hypothesis is represented by chromosome, and the algorithm tries to find the best solution, based on a statistical function, which is used to evaluate hypotheses.

In the proposed classifier we want to find the best weights that can be used so that the classifier is effective as much as possible. Using Genetic algorithm each chromosome represents a candidate solution to

the problem of finding best weights in Equation 4.12, for example a chromosome with values (0.2 , 0.3 , 0.1 , 0.4 , 0.0) reflects suggested values for the different weights, where 0.2 is the value of w_{R1}, 0.3 is the value of w_{R2}, 0.1 is the value of w_T, 0.4 is the value of w_N, and 0 is the value of w_U, the weights all sum to 1, this is a sample of a candidate solution for weights, the algorithm evaluates each candidate solution, to find the best one. The evaluation function is known as fitness function, and in our problem we want to evaluate weights values for the log-likelihood function, thus the fitness function of each chromosome is defined as the log-likelihood $\ddot{\mathcal{L}}(a|x)$, the algorithm will continue to generate candidate chromosomes according to the fitness function value, until reaches a candidate solution that maximize fitness function.

Such a function is considered a simple function to state the efficiency of the classifier, and can not guide Genetic algorithm to find optimum weights, so a more effective fitness function is a function that maximizing the log-likelihood of the correct author, while minimizing it for the other authors, this is implemented as follows:

$$Fitness(d_i , a_j) = \frac{\ddot{\mathcal{L}}(a_j|d_i)}{\sum_{k \neq j} \ddot{\mathcal{L}}(a_k|d_i)} \quad \ldots\ldots\ldots\ldots\ldots \quad (4.14)$$

Where d_i is the i^{th} document in a separate data set, used only by the Genetic algorithm, this data set called Held-out data set and only used to estimate the fitness function, and a_j is the j^{th} author that we want to find optimum weights for his corresponding log-likelihood.

Equation 4.14 find the fitness function of one sample in Held-out data set, and since we have many samples in this set, a final fitness function is defined as the average function over all samples in the Held-out data

set, so given a Held-out data that contains *m* documents for author *a*, then the final fitness function for author *a* is defined as:

$$Fitness(a) = \frac{1}{m} \sum_{i=1}^{m} Fitness(d_i, a) \quad \ldots\ldots\ldots\ldots\ldots \quad (4.15)$$

Chapter 5

EXPERIMENTATION AND RESULTS

5.1 Dataset

A known source for authors' documents can be found in newspapers' websites, they contain many articles for different authors. We use articles from *Felesteen* newspaper website [35], by choosing 9 different authors, and collecting 30 Arabic articles per author. The average size of articles per word is about 700. Table 5.1 views authors and the average size of articles for each author.

The dataset is divided in two sets, the first which consists of 20 documents for each author, will be used in the training and testing phase in order to train authors' language models, and testing the accuracy of the classifier, based on the Leave-One-Out method (Section 3.5), the second data set consists of the remaining 10 documents for each author, it is called the Held-out data set and will be used by Genetic Algorithm to find the optimum weights for the different features, to get high accuracy in the classification phase.

5.2 System Environment

We implement the system using special software and hardware, the software is implemented using Java programming language, and with the help of some packages that provide the parsing functionality, and Genetic algorithm, the following two sections describes the details of parsing package, Genetic algorithm package, and the final section describes the hardware used to implement the proposed algorithm.

Table 5.1: Authors' names, and documents averaged size, that used to form training and testing data set, plus the Held-out data set

Author No.	Author Name Arabic	Author Name English	Average Article Size (per word)
1	محمود عبد الرحيم	Mahmoud Abd Elrahim	833
2	خالد محمود	Khaled Mahmoud	641
3	علاء الريماوي	Alaa Remawi	664
4	مصطفى اللداوي	Mustafa Ledawi	860
5	جمال أبو ريدة	Jamal Abu Reda	607
6	خالد الخالدي	Khaled Khaldey	673
7	عصام عدوان	Esam Edwan	577
8	محمد المدهون	Mohammad Madhown	835
9	عدنان أبو عامر	Adnan Abu Amer	613
Average Size			**700.33**

5.2.1 Parser

The training phase starts by first parsing all authors' documents, as Figure 4.1 shows, so that we need a package that can deal with Arabic texts and can produce the grammar for an Arabic document.

The Stanford parsing package [36] is a powerful software that is built using Java language, the parser was proved to be efficient in parsing Arabic texts [37], it can be used as a standalone software by passing input to it, and capturing the output, or it can be used as a module in any Java application, because it provides a set of Application Programming Interface (API) that can be used in a custom Java application. We use this API to integrate the Stanford parser in a new application that is built for author attribution problem. The parser is populated under the GNU license [38], and available for public in the Stanford university website. Stanford parser package provides three probabilistic parsers:

1. An accurate un-lexicalized probabilistic context-free grammar (PCFG) parser.
2. Probabilistic lexical dependency parser.

3. A factored, lexicalized probabilistic context free grammar parser, which does joint inference over the product of the first two parsers

The first parser is recommended when parsing English language, because in many cases the lexical preferences are not available or inaccurate for many domains, thus the un-lexicalized parser will perform as well as lexicalized parser, also using un-lexicalized parser is faster, and requires less memory. The dependency parser can be used alone, but this is usually not useful, because its accuracy is much lower.

The factored lexicalized parser provides a greater accuracy since it combines the features of the other two parsers, this is done by combining the preferences of the two parsers using *A** algorithm [39], also it is recommended for other languages such as German, Chinese, and Arabic, so that this parser is used to parse authors' documents.

The output of the parser can be presented in various forms, such as: (1) Part of speech tags (POST): this will present only the part of speech tag for each word in a sentence. (2) Dependencies: view the grammatical relations between parts of a sentence; it is only available for English language. (3) Phrase structure trees: this present the structure of the parsed sentence using a tree structure, so that we can see the part of speech tag of each structural unit of the sentence, as in Figure 3.2.

As we said the Stanford parser is a probabilistic parser, which is trained over hand-parsed sentences, to parse new sentences. Stanford Arabic parser is trained over Penn Arabic Treebank [40], which is corpus of parsed sentences, provided by Penn University, it is a famous corpus used by many researchers for training parsing algorithms and other

NLP applications, they provide many corpuses each for a specific language, the parser uses the Arabic corpus. The corpus aims to provide a large Arabic machine-readable text corpus, that annotated by humans and computer. It provides a presentation of Arabic language structure at different levels: starting from word level, phrase level, and sentence level.

The process to make such a corpus consists of two phases. The first is Part-of-Speech tagging: by tokenizing the text into lexical tokens and assigning each token a lexical category. The second step is tree-banking, which identifies the structures of word sequences, then assigning categories for each non-terminal node. The first step is done using Tim Buckwalter's lexicon and morphological analyzer [41], which generates a candidate list of POST for each word, then human just select the correct POS tag, the analyzer also helps by automatically assigns some tags such as tagging numerical data, and punctuation marks, at the end of this process XML files are produced. In next step the data went through tree-bank annotation, to produce a representation of language structure as we described before, a final bit process was done manually by annotators (humans), or automatically to check for inconsistencies between the tree-bank ant POS tagging.

The data which is used during these processes is used from the *Agency France Press* (AFP) newswire [42], which is a standard Arabic corpus includes 734 stories, that have 140,265 words, and about 168,123 tokens after segmenting clitics. The project uses human's annotators that are native speakers of Arabic language, to understand enough linguistics to check morphological syntactic analysis and build syntactic structures.

Before using the parser we must view some limitation of it, to recognize its features and capabilities.

(a) Tokenization:

The parser assumes that the supplied text is tokenized as in Penn Arabic Treebank ATB. In general this set assumes a whitespace to tokenize words, and does not split off clitics (A clitic is a linguistic unit that is pronounced and written like an affix but it is grammatically independent, for example "وقال").

Also the parser considers only one character as end of sentence, the end of sentence may be full stop or comma, and it does not support the two for a single text, but in real documents authors use the two marks to separate sentences, so we define the end of sentence to be full stop, and replacing all commas to full stop in all articles before passing them to parser.

(b) Normalization:

The parser trained on a normalized form of Arabic, so that we also normalized our Arabic documents before parsing them, the normalization process includes the following steps:

- Delete *tatweel* characters, such as (الشمـــــس) will be (الشمس).
- Delete diacritics, for example (تَطْلُع) will be (تطلع).
- Replace some characters: as vowels like *Alef* with *hamza* (أ) or *madda* (آ) becomes simply *Alef* (ا), and *Alef maksura* (ى) becomes *Yaa* (ي).

(c) POST:

The parser uses Bies tag set [40], which maps morphological analysis from Buckwalter analyzer to subset of POS tags used in Penn English Treebank (some with different meanings) as shown in

Table 5.2. Also the parser augmented the set to represent words that have the determiner *AL* (ال) cliticized to them, these extra tags start with "DT", and appear for all parts of speech that can be preceded by "Al", so we have DTNN, DTCD, etc.

5.2.2 Genetic Algorithm

To find the optimum weights between different parts of our enhanced PCFG language model (XPCFG), we use Genetic algorithm package which is a Java based package named JGAP [43] of version 1.5.0, the package provides an API, so that we can integrate its functionalities in our author attribution application. It is an open source package, released under GNU license.

5.2.3 Hardware

As known parsing is an expensive task, since it needs high computer resources, to produce the best production rule for a specific sentence, also the main disadvantage of using Leave-One-Out method as we said before, is the computational expensive since we will train the system 20 time, for each author.

For these tasks we use a specific machine contains 16 processing units, each running of speed 2.6 GHz, the machine is 64bit architecture, which makes it a powerful in computations, the total memory is 8GB, and running 64bit-Linux operating system, the machine is able to work 24/7 of the time.

Table 5.2: English POST which is used as mapped tags for Arabic morphological analysis

No.	POST	Description
1	JJ	Adjective
2	RB	Adverb
3	CC	Coordinating Conjunction
4	DT	Determiner/Demonstrative pronoun
5	FW	Foreign Word
6	NN	Common noun, Singular
7	NNS	Common noun, Plural or Dual
8	NNP	Proper noun, Singular
9	NNPS	Proper noun, Plural or Dual
10	RP	Particle
11	VBP	Imperfect Verb
12	VBN	Passive Verb
13	VBD	Perfect Verb
14	UH	Interjection
15	PRP	Personal Pronoun
16	PRP$	Possessive Personal Pronoun
17	CD	Cardinal Number
18	IN	Subordinating Conjunction (FUNC_WORD) or Preposition (PREP)
19	WP	Relative Pronoun
20	WRB	Wh-Adverb
21	,	Punctuation, token is , (PUNC)
22	.	Punctuation, token is . (PUNC)
23	:	Punctuation, token is : or other (PUNC)

5.3 Experiments

5.3.1 Parsing

The first step to proceed in our proposed algorithm is parsing documents, the documents are texts written in Arabic language, the parser only recognized Arabic texts with UTF-8 encoding , so we first convert all texts to this encoding, then applying all normalization steps described in Section 5.2.1, then the application sends the parser a sentence per time to get the result, a special case here any sentence with size of 250 characters or more, will be ignored since the parser failed in parsing such long sentences.

All documents in both the training and testing data set are parsed, which contains 20 documents for each author, and documents in the Held-out data set which contains 10 different documents for each author, the parser's result for each document is stored in a separate binary file, so that it can be used in different processes without requiring to re-parse it, this minimizes the computations, especially because we use the Leave-One-Out method.

5.3.2 Training

We use the Leave-One-Out method to train and test the system as described in Section 3.5. Using Leave-One-Out in classification starts from first document in the data set and considers this document as test sample, and the other document as training documents, so that for example we start from author number 1, and document number 1, an XPCFG model is trained using 19 documents for author number 1, and all 20 documents for other authors, this model is stored in a binary file with the format *Author1_1.pcfg,* also an XPCFG language model is

trained using the whole documents (*Author1_full.pcfg*) this full-trained language model will be used in classification, so that for each author we produce 21 XPCFG language models, where the first one is trained using 20 documents for a specific author, while the others trained using 19 documents for the corresponding author, by excluding a new document each time. Each one of these language models will be stored in a separate binary file to be used in classification easily.

In classification, the test document is passed to all authors XPCFG models to compute the likelihood score, the document will be assigned to author whose XPCFG model generate the highest score among the other models, for example to test document number 5 for author number 1, the system will pass the document to XPCFG of author 1 that excludes the 5^{th} documents (*Author1_5.pcfg*), and to the full-trained XPCFG for other authors, by this the system implements the Leave-One-Out method.

5.3.3 Computing Optimum Weights

Using the JGAP package we configure the chromosomes to contain 5 genes, each reflects a different weight in Equation 4.12, a gene will carry fractional values, with a constraint that all genes values sum to one. The algorithm starts with random values for genes. Also we implement the fitness function in Equation 4.15, the algorithm configured to start from a population of size 20 samples, the size of the population is not fixed, so along algorithm processing it may be increased by adding more samples, or decreased by removing some.

To estimate a fitness function of an author *a*, we average the fitness function over his Held-out data set which contains 10 documents for

each author, the documents in Held-out data set are not used in training. To compute such a function we need to calculate the log-likelihood between the corresponding author's language model, and each document in Held-out data set, that are belongs to this author (Equation 4.14), then a final estimation is averaged over these documents for the corresponding author as shown in Equation 4.15.

5.4 Performance Measurement

As known there are many measurements that can judge the efficiency of a classifier, we use three different measurements, which are error rate, accuracy, and precision [27]. First because we use Leave-One-Out method in training and testing, it is important to compute the error of the classifier when setting one document as a test document and the others as training documents, the error rate is simply the proportion of misclassified documents of the total documents for a corresponding author, so the error rate will be always calculated by counting the number of misclassified document for a specific author divided by the number of author's documents, which equals 20 documents. For example suppose that we want to classify the 20 documents for a specific author using Leave–One-Out method, and the system truly classifies 17 documents, then the error rate will be 3/20 = 15%. After error rate is computed for each author an average will be computed for all authors, to reflect the error rate for the system, and clearly we seek to minimize this value.

Another quantity that is used to measure the performance of a classifier is the overall accuracy which is the degree of closeness of measurements of a quantity to that quantity's actual value (true value) [44], in classification problems it is the number of correctly classified

documents (true positive and true negative) divided by the number of documents classified overall, it is clear that the correctly classified documents, are the documents that belonged to an author, and classified to that author, or the documents that belongs to other authors, and each classified to its corresponding author. To compute the overall accuracy we must compute the accuracy for each author, and then find the average accuracy over all authors.

On the other hand precision is the degree to which repeated measurements under unchanged conditions show the same results. In classification problems it can be viewed as the number of correctly classified documents belongs to specific author, divided by the number of all documents classified to be belongs to that author. We will also compute the precision for each author, and then we can obtain the average precision for the system.

Clearly we need a maximum value for the accuracy, and precision, in the opposite of the error rate which better to be as minimum as possible.

5.5 Results

We have two data sets; the first of size 180 documents, and the second (Held-out) is of size 90 documents, Figures 5.1, and 5.2 show the relation between document size, and needed time to parse the document, and because the parser handles a document one sentence each time, the size of document is calculated by number of sentences. The two figures suggest a direct proportion between the document size, and the needed time to parse it. Although parsing is a computational

expensive process, the needed time did not exceeds 200 seconds, this may returned to the powerful hardware and software used in parsing.

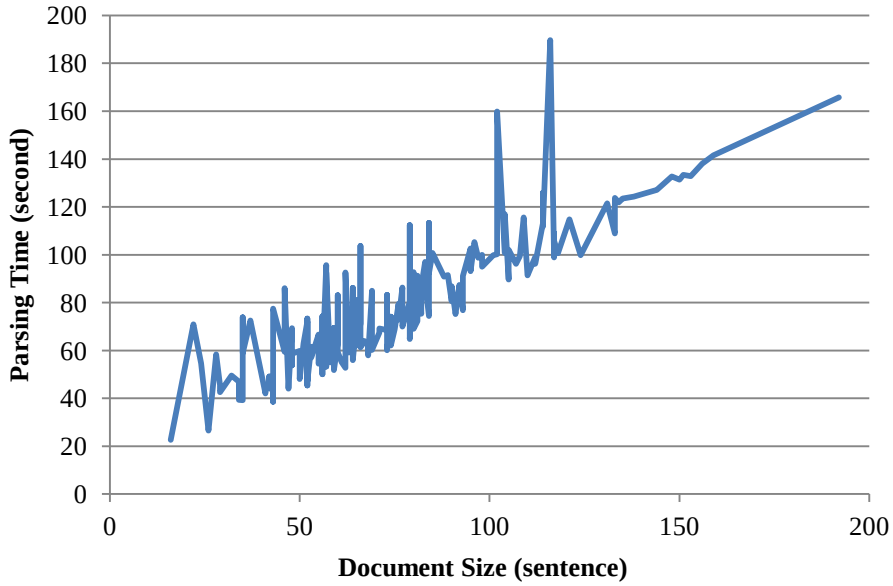

Figure 5.1: Parsing time and document size in training and testing data set.

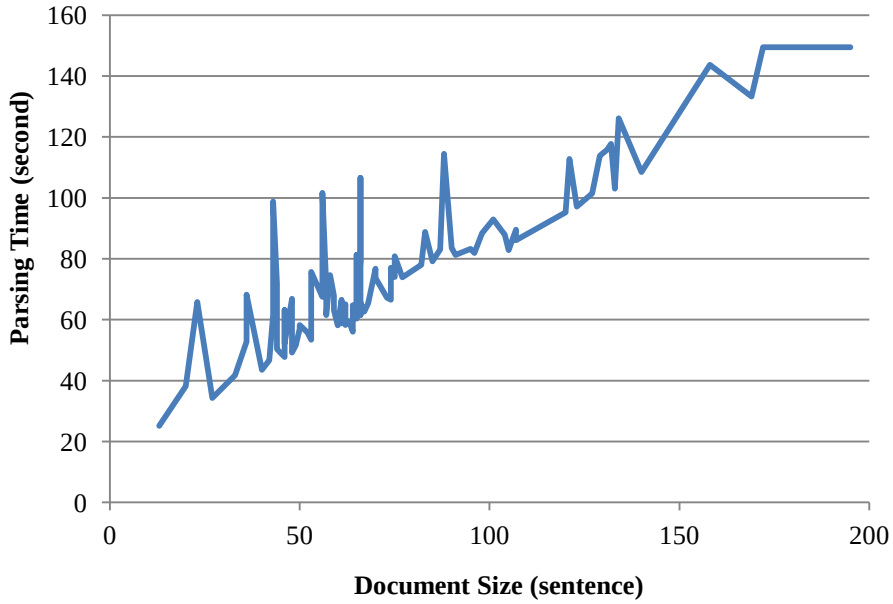

Figure 5.2: Parsing time and document size in Held-out data set.

The training phase is a very computational expensive task; because it is done by using Leave-One-Out method, so that the system is trained 180 different times, every time with different documents, also in each time the system computes the X^2 score for each rule, which requires accessing all documents and counting as Equation 3.1 described, the training phase requires too much of time (about hours) for every step of training. Figure 5.3 tries to describe the relation of training time with the size of training documents, the document size in the Figure is the size of the leaved document, remember in training, the system leaves one document in each step, and trained using the remaining documents. The most time consuming process in training is the estimation of X^2 since the system will access all documents. Note that the training time in Figure 5.3 do not include parsing time, since documents already be parsed, and the result is stored in binary files.

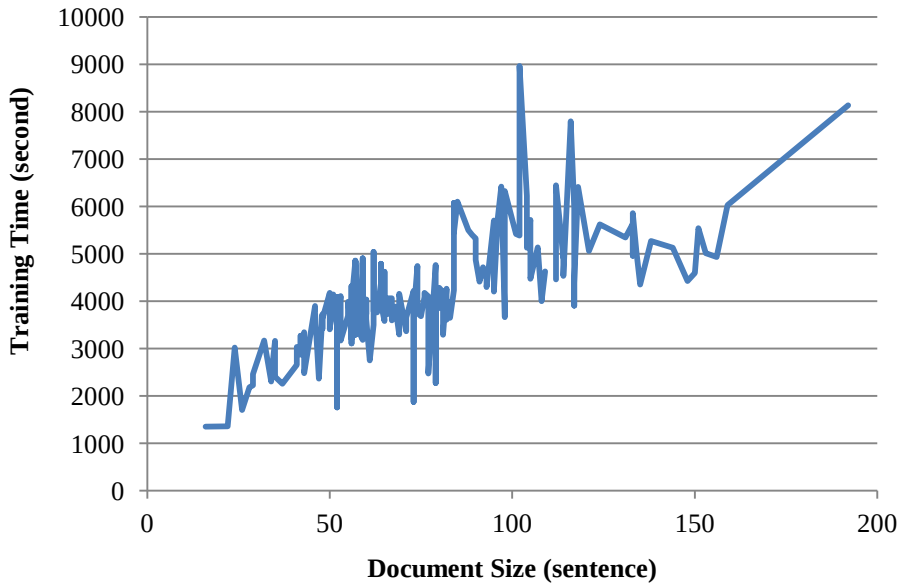

Figure 5.3: Training time and document size, where document size is the size of the leaved document in training

Using Leave-One-Out method we will have 180 documents (9×20) to be classified, and we compute three different measurements to describe system's efficiency. Since PCFG was tested in English language [14] we retest PCFG performance over our data set which is Arabic documents, to compare the results between the PCFG and the enhanced XPCFG language model. Table 5.3 shows error rate, precisian, and accuracy for each author using the Leave-One-Out method over the PCFG language model proposed in [14], it shows that the system achieves best accuracy in author #9 (97.22%), best precision in author #3,5,6 (100%), and best minimum error in author #1 (00.0%), it seems as a good result for a classifier but the final accuracy, precision, error rate for the system is 92.22%, 79.68%, 34.33% respectively, which means that PCFG model contains features that can be used in author attribution problem, but it cannot success for all authors, so that it cannot capture different writing styles for authors, this result supports our hypothesis that the PCFG model contains richness of features. In the other hand we can see in Table 5.4, which describes the result for XPCFG model, that the system achieves best accuracy for author #6,7 (98.33%), best precision 100%, and best error rate 00.0%, the result is almost similar to PCFG model, but when looking to the overall performance of the system we can see that it achieves a 95.74% accuracy, 86.88% precision, and 20.5% error rate, which is better than the previous model, this can proves that adding more lexical, and syntactic information to the traditional PCFG language model can increase its efficiency in author attribution problem. Comparing with PCFG's result, our new model achieves a better result in all measurements (error rate, precision, and accuracy), the enhancement is about 3.5%, 7.19%, and 13.88 in accuracy, precision, error rate

respectively. The highest enhancement is done in error rate which is the percent of misclassified samples, so the number of truly classified samples is increased, which satisfies the users of author attribution system.

Table 5.3: PCFG Model Results

Author	Accuracy	Precision	Error Rate
1	0.911111	0.555556	0.0
2	0.922222	0.666667	0.4
3	0.911111	1.0	0.8
4	0.966667	0.818182	0.1
5	0.944444	1.0	0.5
6	0.966667	1.0	0.3
7	0.916667	0.857143	0.7
8	0.788889	0.333333	0.1
9	0.972222	0.941176	0.2
Average	**0.922222**	**0.796895**	**0.344444**

Table 5.4: XPCFG Model Results

Author	Accuracy	Precision	Error Rate
1	0.916667	0.571429	0.0
2	0.916667	1.0	0.85
3	0.983333	1.0	0.15
4	0.983333	0.869565	0.0
5	0.944444	0.666667	0.0
6	0.983333	0.869565	0.0
7	0.938889	1.0	0.55
8	0.966667	0.9375	0.25
9	0.983333	0.904762	0.05
Average	**0.957407**	**0.868832**	**0.205556**

Remember that the XPCFG contains weights for each feature, as we described in Equation 4.12, the previous results (Table 5.4), is done using equal weights. To find the best weights between the different parts of XPCFG model we use the Genetic algorithm, which uses the

Held-out data set to find these weights, since the fitness function described in Equation 4.15 is estimated using the error rate, the Genetic algorithm will not run for authors that achieve 00.0% error rate in Table 5.4, since the system already achieves the minimum error for those authors.

The Genetic algorithm computes the optimum weights as shown in Table 5.5. After weights are calculated, we can use these weights to re-classify authors' documents (also by using Leave-One-Out method), and re-estimate the accuracy, precision, and error rate for each author, this is shown in Table 5.6 where there are no changes in result for authors that already reaches minimum error rate, the result shows that there is an enhancement when using weights generated from Genetic algorithm, for example author #2 achieves accuracy of 95.5% which is an enhanced value of the XPCFG result for that author (Table 5.4).

Looking to the overall results, the accuracy of the classifier reaches 96.5% this is an enhancement of the XPCFG's result shown in Table 5.4, the percentage of the enhancement in accuracy is 3.389%, at the level of the precision the system achieves 86.88% of overall precision, which also an enhancement by 7.78% of the previous model, also the averaged error rate achieves a good result of 12.77%.

A comparison between authors' precision, accuracy, and error rate is shown in Figure 5.4, 5.5, and 5.6 respectively. These figures show that the PCFG model achieves an acceptable result for some authors, however it can not gain good overall results, so the model can capture writing styles for some of the authors, while the XPCFG model gains an acceptable overall performance, since it is augmented with more features that successfully capture authors' styles, and since the writing

style is different between authors, setting an efficient weights for features in XPCFG model, can achieves a more enhanced result.

Comparing the results obtained by the XPCFG language model, and the results of machine learning based methods, we can note that the XPCFG language model achieves a good result over the methods based on machine learning approach, the XPCFG achieves high result over methods proposed in [1,10,15,16,17], this also support our second hypothesis which suggests that language models contain richness of features that can be used in author attribution problem, and can determine authors styles.

Table 5.5: XPCFG weights for authors, calculated by Genetic Algorithm

Author	w_{R1}	w_{R2}	w_T	w_N	w_U
2	0.00	0.00	0.02	0.96	0.02
3	0.00	0.91	0.04	0.01	0.04
7	0.02	0.87	0.03	0.06	0.02
9	0.00	1.00	0.00	0.00	0.00

Table 5.6: XPCFG Model Result using different weights

Author	Accuracy	Precision	Error Rate
1	0.916667	0.571429	0.0
2	0.955556	1.0	0.4
3	0.988889	1.0	0.1
4	0.983333	0.869565	0.0
5	0.944444	0.666667	0.0
6	0.983333	0.869565	0.0
7	0.961111	1.0	0.35
8	0.966667	0.9375	0.25
9	0.983333	0.904762	0.05
Average	**0.964815**	**0.868832**	**0.127778**

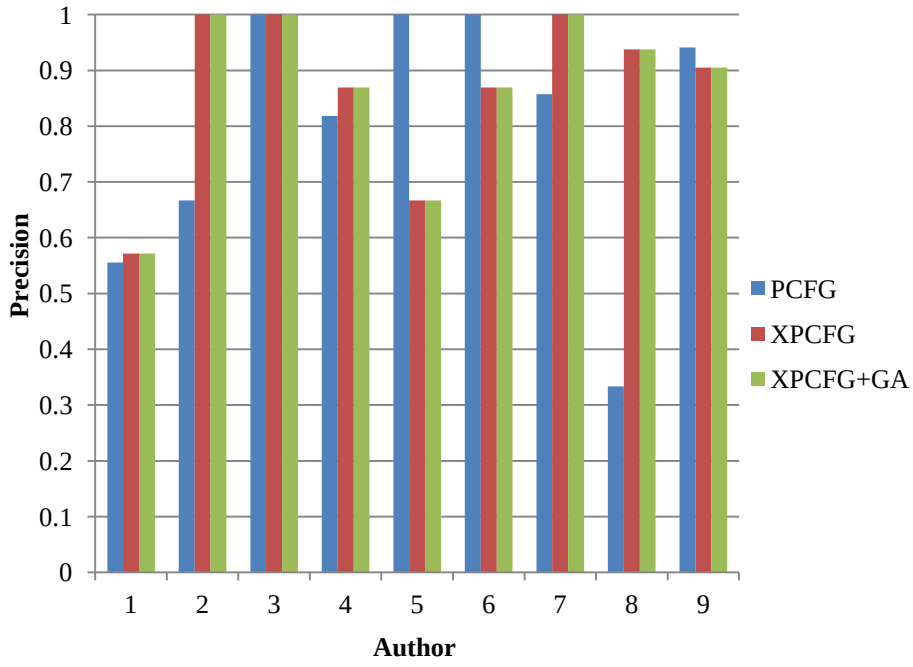

Figure 5.4: Precision for each author, using the three models; PCFG, XPCFG, and XPCFG with weights.

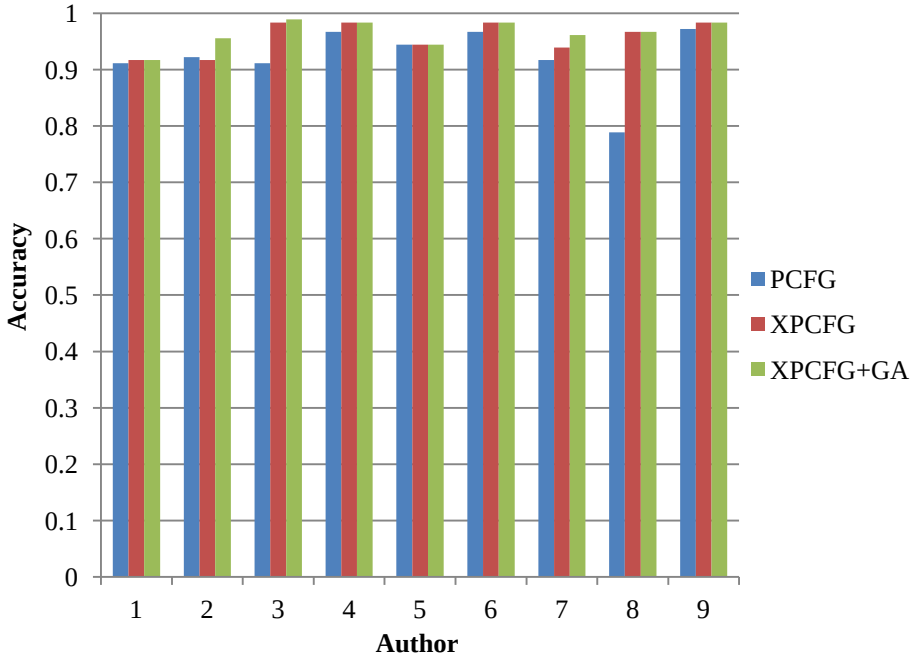

Figure 5.5: Accuracy for each author, using the three models; PCFG, XPCFG, and XPCFG with weights.

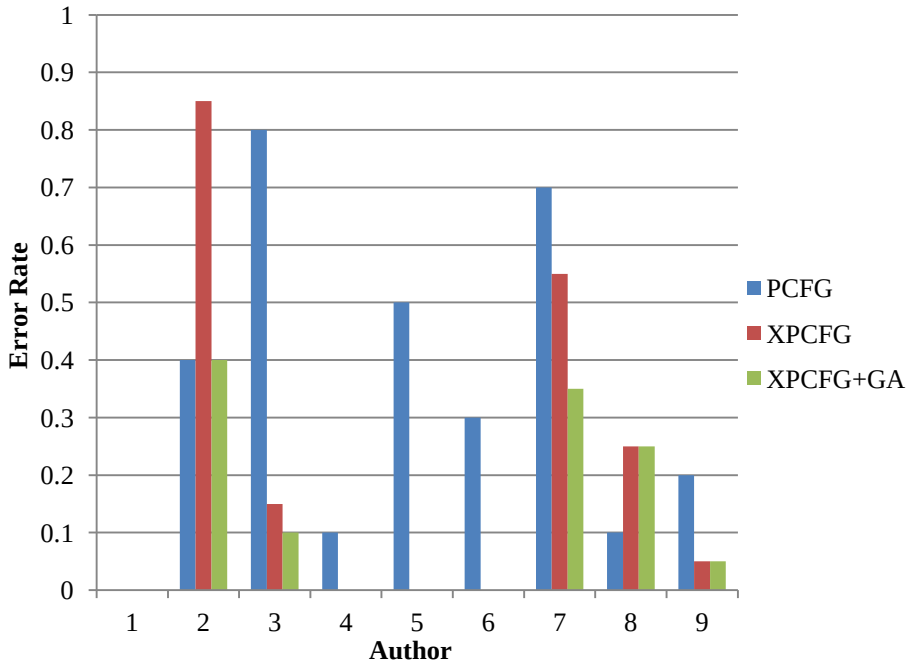

Figure 5.6: Error rate for each author, using the three models; PCFG, XPCFG, and XPCFG with weights, some authors do not have error rate for some of their models

Returning to Table 5.5 if we ignore the very small values, you can note that for any of the authors the whole weight goes to a specific feature. For example for author #2 all the weight going to non-terminals log-likelihood, this means that for author #2 the non-terminals can sufficiently used to classify documents. From that we can state that PCFG language model alone can not capture the style for all authors, since some authors may be discriminated by measuring their usage of non-terminals (author #2), or by measuring only the PCFG non-terminal rules (author #3, 7, 9).

Also we can note that terminals, terminals PCFG rules, and punctuation marks are less descriptive features in author attribution problem, even this they participated in classification for the authors (Table 5.4). They

are the less discriminate features, because terminals are a topic dependent features, depend on the topic of the written text, this applies also to terminal PCFG rules. The punctuation marks participate in classification of some authors, but for the authors in Table 5.5, the punctuation marks failed to gain high weight, this may returned to two reasons, the first that authors use punctuation marks with almost same richness in text, or punctuation marks in Arabic texts can not sufficiently discriminating the author of the text.

Chapter 6

CONCLUSIONS

6.1 Summary and Concluding Remarks

We proposed a new method to solve author attribution problem, this method depends on the language model theory, since the traditional methods used in author attribution problem use a set of features and a machine learning classifier to solve the problem of identifying the author of an unknown text.

The proposed system enhances the PCFG language model, by first adding some lexical, and syntactic features to the model, all these features are extended from the PCFG model, which contains rich information that can be used in author attribution problem, the proposed language model (XPCFG), first separates the production rules of PCFG, in two sets the first is the set contains the non-terminal rules, while the other contains terminal rules, also the XPCFG model adds lexical, and syntactic information by capturing the non-terminals, terminals, and punctuation marks. This information is described by their probabilities in the corpus that are used by the language model in training phase, so that XPCFG contains probabilities for non-terminals, terminals, and punctuation marks, plus the probabilities of the production rules inherited from the PCFG model, adding such information can reflect the writing style of authors, the rules can describe the structure of sentences used by an author, while non-terminals can capture the POS tags that are used by authors, for example a specific author has high probability for using verbs, while

the other has low probability for nouns, the terminals describe the richness of words used by the author, and punctuation marks capture the format style of the author.

Another enhancement done by XPCFG model is that it assigns weights for PCFG rules, these weights describe the importance of each individual rule; it is calculated using chi-square score, so that a rule that appears in few authors will gain high score, while other rule that appears in all authors will gain a very small weight. The purpose of adding rules weights to XPCFG is to enable the language model to automatically find the most discriminative rules among different author.

The system is trained using a set of documents for each author, and produces an XPCFG language model for each author. In the classification phase an unknown document is assigned to each author's language model, to find the best model that most likely can produce the unknown document, this is done by estimating the log-likelihood between the test document and each author's XPCFG. Using log-likelihood because that the XPCFG cantinas only probabilities that describe its content.

The log-likelihood is computed for each part of XPCFG (rules, non-terminals, terminals, and punctuation marks), and a final log-likelihood between test document and XPCFG is estimated by summing the log-likelihood for its parts. The last enhancement done by XPCFG is that the summing of its parts is governed by weights, which describe the importance of each part in classification of an unknown text.

The proposed system is tested over Arabic texts, and we use three different metrics to measure the efficiency of the system, the accuracy, precision, and error rate. The system achieves 95.7%, 86.8%, and 20% of accuracy, precision, and error rate respectively; with this result the XPCFG model exceeds the traditional PCFG language model in author attribution problem.

The weights of the XPCFG is estimated using Genetic algorithm, which searches for the best solution for the weights, so that optimizing a predefined function, and using these weights calculated by Genetic algorithm, the system reaches a more satisfaction results, by achieving accuracy of 96.4%, error rate of 12.7%, which is also an enhancement in author attribution problem.

The estimation of weights done by Genetic algorithm exploits important facts, for some authors the Genetic algorithm finds that the best classification efficiency achieved when using only non-terminals probabilities, or non-terminal PCFG rules, from this we can conclude that PCFG rules (which capture the structure of sentences) can not successfully capture the writing style for all authors, there are some authors where their writing style can not be captured by the structure of sentences that used in their written texts.

So that the extension of the PCFG model to produce XPCFG model can sufficiently capture different writing styles for different authors, especially if the weights between the parts of XPCFG model was set properly.

6.2 Recommendations and Future Work

The proposed system depends on the rules generated by the parser, so that the more efficient parser leads to more efficient system, as we described later the parser has some limitation, such that the parser can not split clitics, so that parsing of some sentences is not accurate, so if we can split clitics before parsing texts, we can obtain a more accurate rules, and we can discover the efficiency of clitics in author attribution problem in Arabic texts.

Also the parser for each author is trained over the same data set as described in Section 5.2.1, it is more efficient to train a separate parser for each author, so that each parser is trained using documents for its corresponding author, this may increase the accuracy of the parsing process so that the parser generate more reliable rules related to a specific author, not to general corpus.

Remember that we have calculated the chi-square score for each rule generated from PCFG model, to capture the importance for such rules. We can apply this approach to terminals, non-terminals, and punctuation marks, so the system can automatically find the most important set of terminals, non-terminals, and punctuation marks, especially after that the terminals, and punctuation marks failed to obtain high weights for some authors, so it is good to automatically find the terminals (words) that can capture author' style, since as we said before that terminals is a topic dependent feature.

The system achieves an acceptable result over a small set of candidate authors, we may increase the number of authors in the system, with their documents, to measure the stability of the classifier using more samples, and classes (authors).

When estimating the weights using Genetic algorithm, the results showed that the punctuation marks do not strongly participate in classification, so we may retest the proposed system by setting the weight for punctuation marks to zero to test their importance in author attribution problem, since they participate in the same weights for authors not in Table 5.5.

The XPCFG language model is language-free model, generating such a model does not depend on a specific language (expect the parser), so we may test it over other languages (especially English language), to measure its ability to classify documents written using different languages.

REFERENCES

[1] C. Chaski, "Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations", *International Journal of Digital Evidence*, vol. 4, no. 1, 2005.

[2] N. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo, "Deriving marketing intelligence from online discussion", in *ACM SIGKDD international conference on knowledge discovery in data mining*, Chicago, USA, August 2005, pp. 419–428.

[3] J. Oberlander, and S. Nowson, "Whose thumb is it anyway? Classifying author personality from weblog text", in *COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, 2006, pp. 627–634.

[4] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas, "Effective identification of source code authors using byte-level information", in *International Conference on Software Engineering*, New York, USA, 2006, pp. 893-896.

[5] S. Burrows, A. Uitdenbogerd, and A. Turpin, "Application of Information Retrieval Techniques for Source Code Authorship Attribution", in *International Conference on Database Systems for Advanced Applications*, Berlin, 2009, pp. 699 – 713.

[6] E. Stamatatos, "A Survey of Modern Authorship Attribution Methods", *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, 2009.

[7] S. Argamon, C. Whitelaw, P. Chase, S. Hota, N. Garg, and S. Levitan, "Stylistic text classification using functional lexical features", *Journal of the American Society for Information Science and Technology*, vol. 58, pp. 802-822, 2007.

[8] F. Türkoğlu, B. Diri, and M. Amasyal, "Author Attribution of Turkish Texts by Feature Mining", in *Intelligent computing international conference on Advanced intelligent computing theories and applications*, Heidelberg, Berlin, 2007, pp. 1086-1093.

[9] M. Koppel, and J. Schler, "Exploiting stylistic idiosyncrasies for authorship attribution", In *IJCAI Workshop on Computational Approaches to Style Analysis and Synthesis*, Acapulco, Mexico , 2003, pp. 69-72.

[10] F. Peng, D. Schuurmans, V. Keselj, and S. Wang, "Language Independent Authorship Attribution using Character Level Language Models", in *Tenth conference on European chapter of the Association for Computational Linguistics*, USA, 2003, pp. 267-274.

[11] R. Baayen, H. Halteren, and F. Tweedie, "Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution", Literary and Linguistic Computing, vol. 11, pp. 121–131, 1996.

[12] P. McCarthy, G. Lewis, D. Dufty, and D. McNamara, "Analyzing writing styles with coh-metrix", in *Florida Artificial Intelligence Research Society International Conference*, 2006, pp. 764-769.

[13] N. Indurkhya, and F. Damerau, "Syntactic Parsing," in *Handbook of Natural Language Processing*, 2nd ed., USA, 2010.

[14] S. Raghavan, A. Kovashka, and R. Mooney, "Authorship Attribution Using Probabilistic Context-Free Grammars", in *ACL Conference Short Papers*, Sweden, 2010, pp. 38-42.

[15] K. Luyckx, and W. Daelemans, "Shallow Text Analysis and Machine Learning for Authorship Attribution", in *Computational Linguistics*, Netherlands, 2005, pp. 149-160.

[16] N. Tsimboukakis, and G. Tambouratzis, "Neural Networks for Author Attribution", in *Fuzzy Systems Conference*, London, 2007.

[17] J. Diederich, J. Kindermann, E. Leopold, and G. Paass, "Authorship Attribution with Support Vector Machines", *Applied Intelligence Journal*, vol. 19, no. 1-2, 2003.

[18] K. Luyckx, "Authorship Attribution of E-mail as a Multi-Class Task", in *CLEF 2011 Labs and Workshop*, Netherlands, 2011.

[19] V. Keselj, F. Peng, N. Cercone, and C. Thomay, "N-Gram-Based Author Profiles for Authorship Attribution", in *Pacific Association for Computational Linguistics*, Canada, August 2003, pp. 255-264.

[20] S. Ouamour, and H. Sayoud, "Authorship Attribution of Ancient Texts Written by Ten Arabic Travelers Using a SMO-SVM Classifier", in *International Conference on Communications and Information Technology*, Hammamet, June 2012, pp. 44-47.

[21] K. Shaker, and D. Corne, "Authorship Attribution in Arabic using a Hybrid of Evolutionary Search and Linear Discriminant Analysis", in *Computational Intelligence*, Colchester, September 2010, pp. 1-6.

[22] A. Abbasi, and H. Chen, "Applying Authorship Analysis to Extremist-Group Web Forum Messages", *Intelligent Systems IEEE*, vol. 20, no. 5, September, 2005.

[23] J. Li, R. Zheng, and H. Chen, "From Fingerprint to Writeprint", *Communications Of The Acm*, vol. 49, no. 4, April, 2006.

[24] T. Mitchell, "Genetic Algorithms", in *Machine Learning*, 1st ed., USA, 1997.

[25] K. Luyckx, and W. Daelemans, "Authorship Attribution and Verification with Many Authors and Limited Data", in *International Conference on Computational Linguistics*, Manchester, August 2008, pp. 1086–1093.

[26] M. Koppel, J. Schler, and S. Argamon, "Authorship attribution in the wild", *Language Resources and Evaluation*, vol. 45, no. 1, March, 2011.

[27] C. Manning, P. Raghavan, and H. Schütze, "Scoring, term weighting and the vector space model," in *An Introduction to Information Retrieval*, 1st ed., England, 2009.

[28] Y. Yang, J. Pedersen, "A comparative study on feature selection in text categorization", in *Machine Learning-International Workshop*, USA, 1997, pp. 412-420.

[29] S. Theodoridis, and K. Koutroumbas, "Supervised Learning: The Epilogue," in *Pattern Recognition*, 4th ed., Academic Press, 2009.

[30] C. Chaski, "Empirical Evaluations of Language-Based Author Identification Techniques", *International Journal of Speech Language and the Law*, vol**.** 8, no**.** 1, 2001.

[31] Parsing [online], Available: http://en.wikipedia.org/wiki/Parsing

[32] S. Argamon, S. Levitan, "Measuring the usefulness of function words for authorship Attribution", in *Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*, 2005.

[33] Y. Zhao, and J. Zobel, "Effective and Scalable Authorship Attribution Using Function Words", in *Second Asia conference on Asia Information Retrieval Technology*, Heidelberg, Berlin, 2005, pp. 174-189.

[34] R. Duda, P. Hart, and D. Strok, "Maximum likelihood and Bayesian estimation", *Pattern Classification*, 2nd ed., Wiley Publication, 2001.

[35] Felesteen newspaper [online], Available: http://www.felesteen.ps.

[36] The Stanford Parser. (2012, November). [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml.

[37] S. Green, and C. Manning, "Better Arabic parsing: baselines, evaluations, and analysis" in *International Conference on Computational Linguistics*, USA, 2010.

[38] GNUGP License. (2007, June 29) [Online]. Available: http://www.gnu.org/licenses/gpl.

[39] S. Theodridis, and K. Koutroumbas, "Template Matching", in *Pattern Recognition*, 4th ed., USA, 2009.

[40] Penn Treebank Project. (1999, February 2). [Online]. Available: www.cis.upenn.edu/~treebank/

[41] T. Buckwalter, "Buckwalter Arabic Morphological Analyzer Version 1.0", *Linguistic Data Consortium*, catalog number LDC2002L49, ISBN 1-58563-257-0, 2002.

[42] Agency France Press. (2012, November). [Online]. Available: http://www.afp.com

[43] Java Genetic Algorithms Package. (2012, November). [Online]. Available: http://jgap.sourceforge.net

[44] Accuracy and Precision. (2012, September 3). [Online]. Available: http://en.wikipedia.org/wiki/Accuracy_and_precision