I.J. Intelligent Systems and Applications, **2016, 6, 27-39**
Published Online June 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2016.06.04

Author Attribution of Arabic Texts Using Extended Probabilistic Context Free Grammar Language Model

Ibrahim S. I. Abuhaiba
Computer Engineering Department, Islamic University, P. O. Box 108, Gaza, Palestine
E-mail: isiabuhaiba@gmail.com

Mohammad F. Eltibi
Computer Engineering Department, Islamic University, P. O. Box 108, Gaza, Palestine

Abstract—Author attribution is the problem of assigning an author to an unknown text. We propose a new approach to solve such a problem using an extended version of the probabilistic context free grammar language model, supplied by more informative lexical and syntactic features. In addition to the probabilities of the production rules in the generated model, we add probabilities to terminals, non-terminals, and punctuation marks. Also, the new model is augmented with a scoring function which assigns a score for each production rule. Since the new model contains different features, optimum weights, found using a genetic algorithm, are added to the model to govern how each feature participates in the classification. The advantage of using many features is to successfully capture the different writing styles of authors. Also, using a scoring function identifies the most discriminative rules. Using optimum weights supports capturing different authors' styles, which increases the classifier's performance. The new model is tested over nine authors, 20 Arabic documents per author, where the training and testing are done using the leave-one-out method. The initial error rate of the system is 20.6%. Using the optimum weights for features reduces the error rate to 12.8%.

Index Terms—Author attribution, author identification, language model, PCFG language model, Chi-square score, genetic algorithm.

I. INTRODUCTION

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.

Old applications of author attribution include the traditional plagiarism detection as settling disputes regarding the authorship of old historical documents. It is also applied in criminal law which includes the determination of documents authority in courts and forensic linguistics. More recently, author attribution gained new importance in cybercrimes which include 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 applications by identifying the author of non-commented source code that we are trying to maintain. It is useful to detect the programmer of malicious codes and viruses [5]. Due to the growing increase in the number of documents, especially in the web, automated text categorization by their authors is a useful way to organize a large documents collection. 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 general approach that is used to solve the author attribution problem starts from a set of training documents, which are documents of known authors, a set of features, that are 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 of an unknown author.

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 will use language models to assign an author to a test document of an unknown author [6]. It starts by forming a language model for each author from his own training documents. This model can effectively capture the language syntax of the author, which is considered as one of the syntactic features that proved to be informative for capturing the author's style and can be used in the author attribution problem. However, this type

of features alone is not efficient to discriminate authors.

Our proposed method tries to enhance the language model by involving more syntactic features other than the language syntax, such as Part of Speech Tagging (POST) feature, where each word will be assigned a tag reflecting its corresponding part of speech, such as noun, verb, etc. Also, more lexical features will be added. At the end, we will have an extended language model that contains a rich set of features.

When classifying an unknown document, the method starts to form an uncompleted language model to represent the test document. Then, it 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.

The rest of the paper is organized as follows. Section II describes some of the most related works that have been done in the author attribution problem. Section III describes accurately the proposed extended language model to solve the problem. In Section IV, the experimental environment and results are described. Finally, concluding remarks and future work are presented in Section V.

II. Related Works

Researchers tried to categorize the features that can be used in author attribution. The basic categorization is lexical, character, syntactic, and semantic features:

A. Lexical features

Using this set of features the text is viewed as a sequence of tokens, where a token is a word or a punctuation mark, which are grouped into sentences. From this representation, some features can be computed such as the length of sentences and length of words. Although these features are basic, they can be applied to any language with no additional requirements, but still we need a tokenizer tool to detect tokens and sentence boundaries. However, these features may not capture the style of a written text, especially for texts containing a lot of abbreviations.

Other features that can be extracted from tokens are vocabulary richness features which measure the diversity of the vocabulary of a text. A traditional example 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. Additional vocabulary richness features are the hapax legomenon, and hapax dislegomenon, which are words occurring once, and words occurring twice, respectively. The vocabulary richness features are biased toward text length as they increase when the text length increases; so they are considered unreliable if 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 the contextual information. One can argue that word frequencies cannot capture authors' style since they are topic dependent. Actually, this is true but the big advantage of using word frequencies is to specify function words, which are words that have little lexical meaning but serve to express grammatical relationships with other words. Function words proved to capture the style of the authors across different topics. However, the selection of specific function words require language dependent expertise. There are various researches to find the best function words for the author attribution problem [7].

While word frequencies feature computes the frequency of each word irrespective of the contextual information, the n-grams take advantage 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 the 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 measure various writing errors to capture authors writing styles [9]. These features are captured using spell checker tools, however, the accuracy of spell checkers is problematic for many languages, and the available text is almost error-free since it is available in electronic form.

B. Character features

In 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 be easily 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 on n-grams. 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 tolerance to noise. In cases that the texts are noisy containing grammatical errors or making strange use of punctuations, the character based n-gram model is not affected dramatically. This model shows acceptable results in author attribution problem, but it 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, which may capture redundant information.

C. Syntactic features

The authors tend to use similar syntactic patterns which are out of their consciousness. 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 was done by producing a parse tree for each sentence in a document, and then extracting writing rules frequencies

[11]. The results of using these rules in author attribution problem are 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 and 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 is assigned a tag based on contextual information. Then, frequencies of tags 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 that relevant features can be measured accurately such that the noise in the corresponding data sets remains low. On the other hand, more complicated tasks such as semantic analysis cannot 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 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 [12]. Also, a more advanced approach tried to assign words or phrases semantic information based on their meaning and indication. In general, semantic features require more advanced NLP tools which are not available.

Most of works, which have been done in the author attribution problem, use machine learning algorithms with some set of features. In [1], a set of lexical features is used, as word frequency, text length, punctuation count, and average word length. The features are augmented with part of speech tagging (POST), which is a syntactic feature. The features are then used to generate a linear discriminate function that maximizes the difference between authors' documents groups. This function is used to predict the group membership for a given test document. Ten documents per author are considered, where each document is related to a predefined topic. The achieved accuracy is about 92%. However, using lexical features cannot efficiently describe the author's style, even if they were augmented with a syntactic feature (POST), since POST is a simple syntactic feature that describes the type (syntax) of a single word, and cannot reflect the syntax of a phrase.

Another traditional method is proposed in [13], where 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 capture how an author uses verb forms. They used a POST feature which captures 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), and the last is a Self-Organized Map (SOM). They suggested that the accuracy depends on model deployment, i.e., the parameters that are used to configure the classifiers, but in all classifiers the accuracy did not exceed 85% since using too many features may degrade the performance of the classifiers. Also, the model depends on optimization for parameters estimation, which is a complex and expensive process.

Another method [14] applied 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 POSTs extracted for each token in the text, and the rewrite rules which detect some structure of a sentence such as subjects and objects. They used shallow text analysis to extract the syntactic features. The best achieved accuracy from the two classifiers was about 72%. Although the method combined lexical and syntactic features, it did not achieve good performance, which may be returned to the performance of the shallow text analyzer and the absence of optimization to select lexical and syntactic features.

Another machine learning method in the field of author attribution problem was proposed in [15]. 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 the 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 using of too many features without selection.

In [16], an SVM classifier is also used for author attribution, but instead of building a classifier for each author, a multi-class SVM is used. Three types of features are used: character features represented by character-level n-grams, lexical features represented by word n-grams, functional words, and the syntactic feature represented by POSTs. The author suggested that the precision of such a classifier depends on its configuration, which is a disadvantage for this method, since adjusting parameters for a classifier is not a trivial problem, and requires complex estimations. On the other hand, the multi-class SVM can deal very well with small and large datasets.

In [8], the authors built and tested four different machine learning algorithms, each supplied with a feature vector combined of 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 such as sentence length and word length. Also, they included vocabulary richness features such as type-token ratio, words occurring once (hapax legomenon), and words occurring twice (hapax dislegomenon), POST for each word in the text, and function words. 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. The overall results ranged from 60% to 84%. Using n-grams has two drawbacks, first there is a problem in defining the best value for n, as this method uses many values for n, in order to find the best solution. Second, n-grams may capture content specific information, while we search for stylistic information for the author attribution problem. Also, the used features are treated equally in the classification process, which is a problem.

The basic unit in traditional n-gram models is a word. In [10], a method is proposed based on a character level n-gram model, in which the character is the basic unit, where the details are the same as word-based n-gram models. They suggested that using a character level n-gram will discover useful inter-word and inter-phrase features. The advantage of this method is 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 on 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%. The 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 the n-gram model was applied in [17], 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. A profile is built for each author from the set of most frequent n-grams, with their normalized frequencies generated from training documents. Likelihood classification is used. However, viewing text as a sequence of bytes is not effective for the author attribution problem.

In [18], a classifier based on SVM algorithm is built. 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 and the data set is very small, which may explain the reason behind the low accuracy.

In [19], the authors introduced a set of Arabic function words as features for author attribution. This set of words was used by a hybrid classifier, which used an evolutionary algorithm and a linear discriminant analysis classifier. The role of the evolutionary algorithm is to find a suitable subset of the function words to be used in training the linear discriminant analysis. The system accuracy did not exceed 93%. The drawback of the method is that it depends only on function words to discover authors.

Some methods tried to use different types of features to capture authors' style. In [20], the authors 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 accuracy reached 97% and 94% for English texts and Arabic texts, respectively. Merging different types of features can effectively capture authors' styles, but the method did not perform very well for Arabic texts.

Machine learning methods may achieve acceptable results in author attribution problem, but we notice that almost all methods did not benefit from efficient syntactic features as sentence structure, although this type of features can describe author's style. This is may be due to the difficulty of implementing such features in machine learning algorithms. Even methods that combined syntactic features with other features assumed that all features have the same importance for the author attribution problem.

In the previous researches, there are many features that can be used in the author attribution problem, which can be helpful, but in many cases the huge amount of features may decrease the performance of a classifier. Because of this, many researches were performed in order to select the best features that can be used. One of these researches was proposed by in [21], where a genetic algorithm [22] was used to identify the best features. Here, each gene represents a single feature with value 0 or 1 to indicate whether a feature is selected or not. The fitness function is defined as the accuracy of the corresponding classifier, where an SVM algorithm is used to classify an unknown text. The method shows that choosing 130 features from 270 features can increase the accuracy. The problem of this method is that one cannot capture all stylometric features since 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 the syntactic features were 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 [23], in which features are selected according to their predictive values. These values are calculated using the chi square metric (X2) which estimates the expected and observed frequency for every feature to identify features that are able to discriminate between authors. The

algorithm uses a combination of lexical features, plus syntactic features extracted by a parser to produce POSTs. In classification, two different machine learning algorithms were used: TiMBL and SOM.

In [24], the authors tried to depend on similarity measurements rather than a machine learning approach. They investigated the author attribution problem for large candidates (10,000 authors) using 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 punctuations, numerals, and sundry, to find an author from large set of authors. They used cosine similarity [25], which is a common metric used in information retrieval. The achieved precision is about 46%, so they improved the procedure by repeatedly selecting the top k documents, then computing the score for each author depending on this set. The algorithm returns the author who has the maximum score. 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 restricting the features on n-grams only, which cannot capture the writing style.

Another approach is used in the author attribution problem incorporating language models. This approach assumes that each author has writing characteristics that can be captured using a language model. In [26], the authors used a more advanced language model. They applied the Probabilistic Context Free Grammar (PCFG) language model, by training a language model for each author from his known text documents. A test document is assigned to the author whose language model gives the highest likelihood score. The method achieved a good result in the range of 87 to 95%. A PCFG language model describes the structure of the sentences that are used in text, which is considered as a syntactic feature. The research did not use features other than the syntactic ones expressed by the PCFG model, and achieved good results. Syntactic features give better results in author attribution field if they are combined with other features.

Many of the previous works were tested over documents written in English language [1, 8]. Some used Greek language [10, 15], Belgian language [14], Germany language [15], and Arabic language [18-20]. In this paper, we develop an extended the language model for Arabic texts to solve the author attribution problem.

III. Proposed Method

The proposed method is described in detail in this section.

A. Background

Context-free grammar (CFG) is considered as the most effective grammar formalization for describing language syntax [6]. CFG is defined as a tuple $G = \{\Sigma, N, S, R\}$, where Σ is a set of terminal symbols which are symbols or 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. A production rule 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 a CFG grammar, a phrase can be viewed as a sequence of terminals.

CFG provides a simple and mathematically precise mechanism for describing the methods by which phrases in some natural languages are built from smaller blocks. 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.

Most grammar formalizations are derived from CFG, one of which is the Probabilistic Context Free Grammar (PCFG), in which each production rule is assigned a probability. These probabilities are required to sum up to 1.0 for each non-terminal. We can view a PCFG as a tuple $G = \{\Sigma, N, S, R, P\}$, where P is a list of probabilities, each probability is assigned to one of the rules in R, and defines the likelihood with which this rule is used in generating a sentence. After generating a PCFG grammar, G, and computing a probability for each rule from the training data, the probability of generating a string is the product of the probabilities of productions taken at each branch of its parsing tree.

In our approach, 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 classification problems, the X^2 score measures the lack of independency between a feature, t, and class, c. 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 of a rule r and an author c [27]:

$$X^2(r,c) = \frac{N(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)} \quad (1)$$

where A is the number of times r and c co-occur, 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 of the same category. A rule with small score denotes that the rule is not discriminative for that author, while a high score denotes that it is discriminative and captures the author's style.

B. Details of proposed method

The PCFG grammar describes the language syntax, but this alone cannot be efficiently used to distinguish the author of an unknown text [26], 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. We will incorporate some basic features as lexical features to the PCFG language model to distinguish between authors [1, 10]. These features proved to be informative [14].

Recall that we already know the words in each sentence (terminals) and their types (non-terminals) from the PCFG language model. We will use this to capture some lexical features, which will be handled by adding a new set, P_T, which contains the probabilities of terminals, to the grammar, G. Also, a second set, P_N, will be added to the grammar, G, which includes the probabilities of non-terminals. This set is predefined and will be of fixed size. PCFG do not consider punctuation marks in generating the rules and their probabilities. However, punctuation marks are considered major features to capture the style of a text [28], so we will add a new third set, P_U, to grammar, G, which includes the probabilities of punctuation marks.

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

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

where X^2 is the set of weights for each rule in R, P_T is the probabilities of terminals, P_N is the probabilities of non-terminals, U is the set of punctuation marks, and P_U is the probabilities of punctuation marks.

Our algorithm will generate an XPCFG model for each author from his set of training documents using a parsing tool. After generating the production rules, a probability is computed for each production rule. Then a score is computed for each rule to compute the dependency between this rule and its corresponding author, which is accomplished by computing the X^2 score for each rule. Also, the probabilities of terminals, non-terminals, and punctuation marks are computed. Fig. 1 illustrates the process of generating author's XPCFG language model.

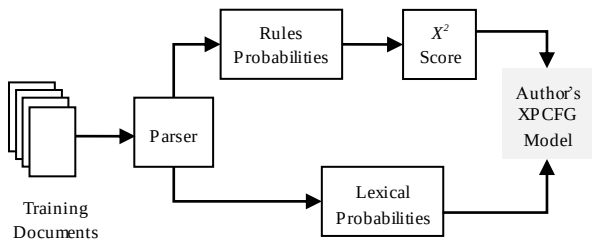


Fig.1. The process of generating the extended language model, XPCFG, for a specific author using a set of training documents belonging to that author.

We use different types of features: grammatical features represented by rules, lexical features represented by probabilities of terminals, non-terminals, and punctuation marks (actually, non-terminals are considered syntactic features). A genetic algorithm is used to find the best weights for these features, as shown in Fig. 2. The algorithm uses a new corpus called genetic data set, which is used only for the purpose of finding the best weights of different features for a specific author. The weights depend on maximizing the classification accuracy for documents in the genetic set.

Finally, in the classification process, Fig. 3, a test document is passed to the classifier, with all authors' models and optimum weights for each author as inputs. The classifier estimates a score between the test document and each model so that the test document is assigned to the author who has the maximum score.

As shown in Fig. 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 [29]. So, any document in training, testing, or genetic 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 the language gained from previously hand-parsed sentences. The result of the parsing process is a set of grammatical rules.

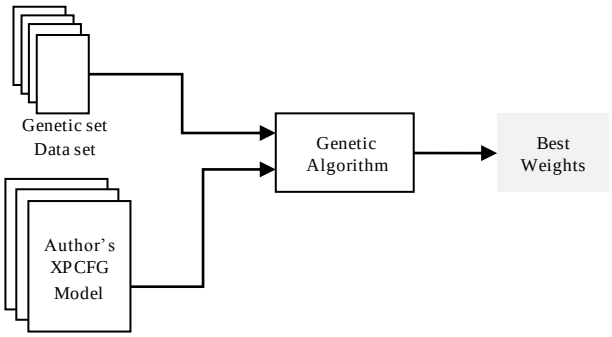


Fig.2. Estimation of optimum weights of different features in XPCFG model for a specific author.

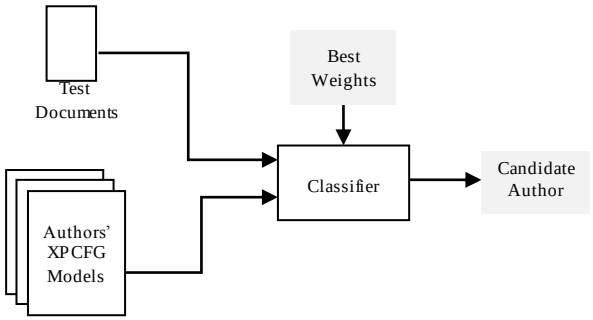


Fig.3. Classification of a test document.

In training phase, Fig. 4, we produce a full XPCFG language model for each author. The language model includes the rules produced in the parsing phase, with their probabilities and scores, and three lists of terminals, non-terminals, and punctuation marks, with their probabilities. The following subsections explain the training steps in details.

After parsing each document in the training data set, and producing the rules, the probability for each rule is computed. For example, the probability $p(r \rightarrow a\, b)$ of a

rule $r \rightarrow a\ b$ is computed by:

$$p(r \rightarrow a\ b) = \frac{count(r \rightarrow a\ b)}{count(r \rightarrow anything)} \quad (3)$$

If 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 the i^{th} training document for that author, then the average probability $\bar{p}(r \rightarrow a\ b)$ is given by:

$$\bar{p}(r \rightarrow a\ b) = \frac{\sum_{i=1}^{n} p_i(r \rightarrow a\ b)}{n} \quad (4)$$

Following this procedure, each author language model contains non-duplicated rules, and a rule has a probability that reflects the average probability of that rule in all training documents that belong to that author.

When rules are generated from training documents by the parser, the right side of a rule contains either non-terminals, or terminals. In a traditional PCFG language model, there is no difference when dealing with these sets of rules, but in our proposed model, XPCFG, we have two different sets of rules, the first is non-terminal rules, and the other is terminal rules. For example, rules such as *S → VP, VP →VBP NP PP, NP → DTNN, PP → IN NP,* and *NP → DTNN* are categorized as non-terminal rules, while the rules *VBP →* تطلع, *DTNN →* الشمس, *IN →* من, and *DTNN →* الشرق are categorized as terminal rules. The goal of this categorization of the rules is to measure the efficiency of each set of rules in the classification process. Note that for the simplicity of notation we did not mention this categorization of rules in (2).

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:
Begin
  For each training document d_j
    Parse document d_j to generate the set of rules *R*.
    Compute rules probabilities *P*.
    Compute terminals probabilities P_T.
    Compute non-terminals probabilities P_N.
    Compute punctuation marks probabilities P_U.
  End loop
  Compute the average probabilities for $P, P_T, P_N,$ and P_U, over all training documents.
  Compute the X^2 score for each rule in *R*.
  Return XPCFG = $\{\Sigma, N, S, R, P, U, X^2, P_T, P_N, P_U\}$
End

Fig.4. Generating of XPCFG language model.

The set of non-terminals, *N*, is of fixed size set, since non-terminals are predefined by the parser. So, for each author's language model, the set, *N*, will contain the same non-terminals, but with different probabilities. For a non-terminal, *nt*, in a training document that contains *m* non-terminals in all rules, the probability of *nt* is given by:

$$p(nt) = \frac{count(nt)}{m} \quad (5)$$

where *count*(*nt*) is the number of occurrences of this non-terminal in the rules of the training document. Following this process for other training documents that belong to same author will produce a set of probabilities for each non-terminal. The final non-terminal probability is the average of all probabilities of that non-terminal in all author's training documents.

To compute terminals probabilities, we follow the process used when computing the non-terminals probabilities. We start from the set of rules that are generated from a training document for a specific author. Using these rules, we count the number of occurrences of each terminal, and divide it by total number of terminals in the training document to obtain the terminal probability. Note that the number of terminals is not fixed; it depends on the size of the training document. A final averaged probability of a terminal is computed by averaging the probabilities of this terminal over the training documents.

The probability of a punctuation mark in a training document is the number of its occurrences in the document divided by the total number of punctuation marks in the document. For a specific author, the final probability of a punctuation mark is the average of its probabilities in all training documents.

As shown in Fig. 4, a score is computed, (1), for each rule to measure the dependency between the rule and its corresponding author. The score will be high if the rule occurs few times in training documents. High-score rules are retained since they capture the author's style.

C. Computing optimum weights

Using a genetic algorithm, each chromosome represents a candidate solution to the problem of finding best weights in (15). For example, a candidate solution for weights is (0.2 , 0.3 , 0.1 , 0.4 , 0.0), where the weights sum to 1 and are for $w_{R1}, w_{R2}, w_T, w_N,$ and w_U, respectively. The algorithm evaluates each candidate solution, to find the best one, using a fitness function that maximizes the log-likelihood of the correct author, while minimizing it for the other authors, which is implemented as follows:

$$Fitness(d_i, a_j) = \frac{\ddot{L}(a_j|d_i)}{\sum_{k \neq j} \ddot{L}(a_k|d_i)} \quad (6)$$

Where d_i is the i^{th} document in the genetic data set, used only by the genetic algorithm, and a_j is the j^{th} author that we want to find optimum weights for his corresponding log-likelihood function. Equation (6) finds the fitness function of one sample in the genetic data set. So given a genetic data set of *m* documents for author a_j, his final fitness function is defined as:

$$Fitness(a_j) = \frac{1}{m} \sum_{i=1}^{m} Fitness(d_i, a_j) \quad (7)$$

D. Classification

We use a probabilistic classifier to assign an author to an anonymous text. The classifier maximizes the probability $P(x|a)$ for a text x to belong to a candidate author a. Using Bayes rule [30]:

$$P(x|a) = \frac{P(a|x)\,P(a)}{P(x)} \quad (8)$$

$P(x)$ is the same for the test document, so, it can be ignored. The prior probability of an author $P(a)$ is often treated as uniform across all authors and it can also be ignored, also. So, 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 $x_1, x_2, \dots, x_n$ 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 [30], first we specify the joint probability density for test document x, by:

$$P(x_1, x_2, \dots, x_n|a) = P(x_1|a)\,P(x_2|a) \dots\dots P(x_n|a) \quad (9)$$

Then, the likelihood function is

$$\mathcal{L}(a|x_1, x_2, \dots, x_n) = P(x_1, x_2, \dots, x_n|a) = \prod_{i=1}^{n} P(x_i|a) \quad (10)$$

Equation (10) estimates how an author, a, is likely to produce a test document, x, using the probabilities computed in author's XPCFG language model. The log-likelihood function is:

$$\hat{\mathcal{L}}(a|x) = \sum_{i=1}^{n} log\, P(x_i|a) \quad (11)$$

To classify a test document x, we use (11), 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 the author who has the highest log-likelihood. To do this, we first will parse the test document to produce an 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 (9) we view x as a set of features $x_1, x_2, \dots, x_n$, but since we have four types of features in x, we can view the test document as four sequences of observations, where the first sequence, $r_1, r_2, \dots, r_A$, is the set of rules in x, the second sequence, $t_1, t_2, \dots, t_B$, is the set of terminals, the third sequence, $nt_1, nt_2, \dots, nt_C$, is the set of non-terminal, and the last sequence, $u_1, u_2, \dots, u_D$, is the set of punctuations marks, $n = A + B + C + D$. The classifier uses (11) to compute $\hat{\mathcal{L}}$ for each sequence, for example, the log-likelihood for terminals, $\hat{\mathcal{L}}_T$, is given by:

$$\hat{\mathcal{L}}_T(a|x) = \sum_{i=1,\ t \epsilon x}^{B} log\, P(t_i|a) \quad (12)$$

In the same manner, (11) 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 probability with its corresponding X^2 score (computed in training phase), so that a log-likelihood for rules is given by:

$$\hat{\mathcal{L}}_R(a|x) = \sum_{i=1,\ r \epsilon x}^{A} X^2(r_i|a) \times log\, P(r_i|a) \quad (13)$$

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 the training phase we defined two different sets of rules, the first contains the terminal rules, while the other contains the non-terminal rules. So, 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 and author a using:

$$\ddot{\mathcal{L}}(a|x) = \hat{\mathcal{L}}_{R1}(a|x) + \hat{\mathcal{L}}_{R2}(a|x) + \hat{\mathcal{L}}_T(a|x) + \hat{\mathcal{L}}_N(a|x) + \hat{\mathcal{L}}_U(a|x) \quad (14)$$

Equation (14) suggests that each part of the XPCFG model participates in the classification with equal weights. We can enhance the classification process by assigning a different weight for each part in (14) as follows:

$$\ddot{\mathcal{L}}(a|x) = w_{R1}\hat{\mathcal{L}}_{R1}(a|x) + w_{R2}\hat{\mathcal{L}}_{R2}(a|x) + w_T\hat{\mathcal{L}}_T(a|x) + w_N\hat{\mathcal{L}}_N(a|x) + w_U\hat{\mathcal{L}}_U(a|x) \quad (15)$$

Weights values are between 0 and 1 and all sum to one. For a test document x, the classifier estimates $\ddot{\mathcal{L}}(a|x)$ in (15) between the document, x, and all available authors, using their XPCFG language models, then assigning the test document the author who has the maximum $\ddot{\mathcal{L}}(a|x)$ value, so that the candidate author $\hat{a}$ for an anonymous text, x, is:

$$\hat{a} = arg\ \max_a \ddot{\mathcal{L}}(a|x) \quad (16)$$

IV. Experimentation and Results

Now, we report experimental results of our method.

A. Dataset

We use articles from *Felesteen* newspaper website [31], by choosing 9 different authors, and collecting 30 Arabic articles per author. The average size of articles is about 700 words, Table 1. The dataset is divided into two sets. The first set, which consists of 20 documents for each author, is used in training and testing the classifier using the leave-one-out method. The second dataset consists of the remaining 10 documents for each author and is used by the genetic algorithm to find the optimum weights for the different features.

Table 1. Average Size of Articles of the Datasets

Author No.	Average Article Size (words)
1	833
2	641
3	664
4	860
5	607
6	673
7	577
8	835
9	613

B. System software environment

The training phase starts by first parsing all authors' documents, Fig. 1. The Stanford parsing package [32, 33] is a powerful software that is built using Java language, and proved to be efficient in parsing Arabic texts [34]. 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, since it provides an Application Programming Interface (API) that can be used in custom Java applications. We use this API to integrate the Stanford parser in a new application that is built for the author attribution problem. 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 outputs 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 a 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 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 [35], also it is recommended for other languages such as German, Chinese, and Arabic. So, 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), which presents only the part of speech tag for each word in a sentence, (2) dependencies, to the grammatical relations between parts of a sentence; it is only available for English language, and (3) phrase structure trees for presenting the structure of the parsed sentence so that we can see the part of speech tag of each structural unit of the sentence.

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 [36], which is a corpus of parsed sentences, provided by Penn University. The corpus aims to provide a large Arabic machine-readable text corpus that is annotated by humans and computer. It provides a presentation of Arabic language structure at different levels: word level, phrase level, and sentence level.

The process to make such a corpus consists of two steps. 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 [37], which generates a candidate list of POSTs for each word, then a human just selects the correct POS tag. The analyzer also helps by automatically assigning some tags such as tagging numerical data and punctuation marks. At the end of this process, XML files are produced. In the second step, the data goes through tree-bank annotation to produce a representation of language structure. A final bit process is done manually by annotators (humans), or automatically to check for inconsistencies between the tree-bank and POS tagging.

The data which is used during these processes is used from the *Agency France Press* (AFP) newswire [38], which is a standard Arabic corpus that includes 734 stories of 140,265 words, and about 168,123 tokens after segmenting clitics. The project uses human's annotators that are native speakers of Arabic language, and have enough linguistics capabilities to check morphological syntactic analysis and build syntactic structures. Before using the parser, we present some of its capabilities and limitations.

1) 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 the end of sentence which may be a 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 a full stop, and replace all commas to full stops in all articles before passing them to the parser.

2) Normalization

The parser was trained on a normalized form of Arabic. So, we also normalized our Arabic documents before parsing them using the following steps:

- Delete *tatweel* characters, for example, (الشمـــــس) will be (الشمس).
- Delete diacritics, for example (تَطْلُع) will be (تطلع).

- Replace some characters, for example, the vowel *Alef* is replaced with *hamza* (أ), *madda* (آ) becomes *Alef* (ا), and *Alef maksura* (ى) becomes *Yaa* (ي).

3) POST

The parser uses Bies tag set [36], which maps morphological analysis from Buckwalter analyzer to the subset of POS tags used in Penn English Treebank (some with different meanings) as shown in Table 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.

To find the optimum weights between different parts of our enhanced PCFG language model (XPCFG), we use the genetic algorithm package which is a Java based package named JGAP, version 1.5.0 [39]. The package provides an API that we used in our author attribution application.

C. Experiments

The parser only recognizes Arabic texts with UTF-8 encoding. So, we first convert all texts to this encoding, then apply normalization steps described previously. The application sends to the parser one sentence at a time. Any sentence with size of 250 characters or more is ignored since the parser fails when parsing such long sentences. All documents in the training and testing data set, and the genetic data set are parsed. 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, which minimizes the computations.

We use the leave-one-out method to train and test the system. It starts from the first document in the data set and considers it as a test document, and the others as training documents. For example, we start from author 1, and document 1, an XPCFG model is trained using the remaining 19 documents for author number 1, and all the 20 documents per other author. This model is stored in a binary file with file name *Author1_1.pcfg*. Also, an XPCFG language model is trained using the whole 20 documents of author 1 and stored in the file *Author1_full.pcfg*. This full-trained language model will be used in classification. So, for each author we produce 21 XPCFG language models.

In classification, the test document is passed to all authors XPCFG models to compute the likelihood score. The document will be assigned to the author whose model generates the highest score, 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 5th documents (*Author1_5.pcfg*), and to the full-trained XPCFGs of other authors. Thus, the system implements the leave-one-out method in training and testing.

Using the JGAP package, we configure the chromosomes to contain 5 genes, each reflects a different fractional weight in (15), and with the constraint that all genes values sum to one. The algorithm starts with random values for genes. We implement the fitness function in (7). The algorithm is configured to start from a population of size 20 samples. To estimate a fitness function of an author, we average the fitness function over his genetic data set. To compute such a function we calculate the log-likelihood between the author's language model and each document in the genetic data set, which belongs to this author, (6). Then, a final estimation is averaged over these documents, (7).

Table 2. English POSTS Which Are Used As Mapped Tags for Arabic Morphological Analysis

POST	Description
JJ	Adjective
RB	Adverb
CC	Coordinating Conjunction
DT	Determiner/Demonstrative pronoun
FW	Foreign Word
NN	Common noun, Singular
NNS	Common noun, Plural or Dual
NNP	Proper noun, Singular
NNPS	Proper noun, Plural or Dual
RP	Particle
VBP	Imperfect Verb
VBN	Passive Verb
VBD	Perfect Verb
UH	Interjection
PRP	Personal Pronoun
PRP$	Possessive Personal Pronoun
CD	Cardinal Number
IN	Subordinating Conjunction (FUNC_WORD) or Preposition (PREP)
WP	Relative Pronoun
WRB	Wh-Adverb
,	Punctuation, token is , (PUNC)
.	Punctuation, token is . (PUNC)
:	Punctuation, token is : or other (PUNC)

D. Results

We use the error rate to measure the efficiency of the classifier [25]. The error rate is calculated by counting the number of misclassified documents for a specific author divided by the number of author's documents, which equals 20 documents. An average error rate is computed for all authors.

We have two data sets; the first consists of 9 × 20 = 180 documents, and the second (genetic) is of size 9 × 10 = 90 documents. Using leave-one-out method, we have 180 documents to be classified. Since PCFG was tested for English language [26], we retest PCFG performance over our dataset, which is Arabic, to compare the results between the PCFG and our XPCFG language model. Table 3 shows the error rate for each author using the PCFG language model proposed in [26]. The system achieves the best minimum error for author 1 (00.0%). The average error rate of the system is 34.4%. The results show that the PCFG model cannot capture different writing styles for all authors. The results of XPCFG

model are shown in Table 4. The system could eliminate the error for four authors: 1, 4, 5, and 6. The average error rate is decreased to 20.6%. This is a result of adding more lexical and syntactic information to the traditional PCFG language model.

Table 3. PCFG Model Results

Author	Error Rate
1	0.0
2	0.4
3	0.8
4	0.1
5	0.5
6	0.3
7	0.7
8	0.1
9	0.2
Average	0.344

Table 4. XPCFG Model Results

Author	Error Rate
1	0.0
2	0.85
3	0.15
4	0.0
5	0.0
6	0.0
7	0.55
8	0.25
9	0.05
Average	0.206

Remember that the XPCFG contains weights for each feature, see (15). The results of Table 4 are obtained using equal weights. To find the best weights, we use the genetic algorithm and the genetic data set. Since the fitness function, (7), is estimated using the error rate, the genetic algorithm will not run for authors who achieve the minimum, 00.0%, error rate in Table 4. The results using the optimum weights are shown in Table 5. The results show that there is an enhancement when using optimum weights. The averaged error rate of the XPCFG model is decreased to 12.8%.

Table 5. XPCFG Model Results Using Different Weights

Author	Error Rate
1	0.0
2	0.4
3	0.1
4	0.0
5	0.0
6	0.0
7	0.35
8	0.25
9	0.05
Average	0.128

A comparison of error rates for the three models: PCFG, XPCFG, and XPCFG with weights, per author is shown Fig. 5. The figure shows that the PCFG model achieves acceptable results for some authors, the XPCFG model achieves acceptable overall performance, and XPCFG model with optimum weights achieves the best results. Comparing the results obtained by the XPCFG language model, and the results of machine learning methods, we notice that the former achieves better results than those methods [1, 10, 13-15].

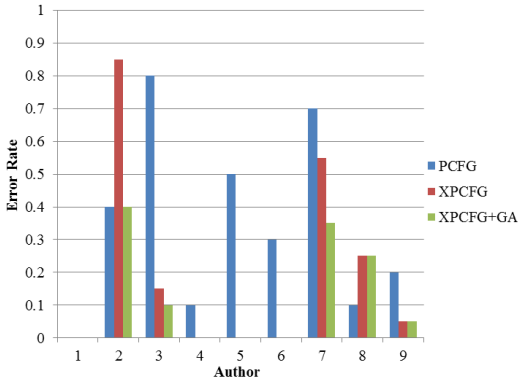


Fig.5. Comparison of error rates of the three models: PCFG, XPCFG, and XPCFG with optimum weights, per author. Some authors have zero error rate for some of their models.

V. Conclusions

We proposed a new method to solve the author attribution problem. It depends on language model theory. The proposed system is an extension of the PCFG. The proposed language model, XPCFG, separates the production rules into two sets, where the first is the set of non-terminal rules and the second is the set of terminal rules. Also, the XPCFG model adds lexical and syntactic features by capturing the non-terminals, terminals, and punctuation marks. These features are annotated with probabilities, in addition to probabilities of production rules inherited from the PCFG model. Adding such information reflects the writing style of authors since the rules describe the structure of sentences, and non-terminals capture the POS tags that are used by authors. The terminals describe the richness of words used by the author, and punctuation marks capture his format style.

Another enhancement implemented in the XPCFG model is assigning scores to rules to quantify the importance of each rule. These scores are calculated using chi-square score. This helps to find the most discriminative rules for each 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 the author whose language model yields the maximum log-likelihood. The log-likelihood is computed for each part of the XPCFG (rules, non-terminals, terminals, and punctuation marks), and a final log-likelihood is computed by summing the log-likelihood parts. Summing the log-likelihood parts is governed by weights which describe the importance of each part in the

final log-likelihood function. The best weights are computed using a genetic algorithm by optimizing a predefined function.

The proposed system is tested over Arabic texts. The error rate of the system is about 20.6%. Hence, the XPCFG outperforms the traditional PCFG language models in the author attribution problem. The system error rate is reduced to 12.8% when the optimum weights are used.

The proposed system depends on the rules generated by the parser. The used parser has some limitations as it cannot split clitics resulting in inaccuracy for some sentences. So, splitting clitics before parsing may result in more accurate rules.

We have calculated the chi-square score for each rule of the XPCFG language model. We can apply this approach to terminals, non-terminals, and punctuation marks, so that the system can automatically quantify their importance.

The system achieved acceptable results over a small set of candidate authors and small datasets. We may increase the number of authors and the size of datasets for more reliable results.

ACKNOWLEDGMENT

We thank anonymous referees for their constructive comments.

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

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.

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

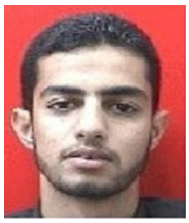
[25] C. Manning, P. Raghavan, and H. Schütze, "Scoring, term weighting and the vector space model," in *An Introduction to Information Retrieval*, 1st ed., England, 2009.

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

[27] Y. Yang, J. Pedersen, "A comparative study on feature selection in text categorization", in *Machine Learning-International Workshop*, USA, 1997, pp. 412-420.
[28] C. Chaski, "Empirical Evaluations of Language-Based Author Identification Techniques", *International Journal of Speech Language and the Law*, vol. 8, no. 1, 2001.
[29] Parsing [online], Available: http://*en.wikipedia.org/wiki/Parsing*
[30] R. Duda, P. Hart, and D. Strok, "Maximum likelihood and Bayesian estimation", *Pattern Classification*, 2nd ed., Wiley Publication, 2001.
[31] Felesteen newspaper [online], Available: http://www.felesteen.ps.
[32] The Stanford Parser. *(2012, November)*. [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml.
[33] GNUGP License. (2007, June 29) [Online]. Available: http://www.gnu.org/licenses/gpl.
[34] S. Green, and C. Manning, "Better Arabic parsing: baselines, evaluations, and analysis" in *International Conference on Computational Linguistics*, USA, 2010.
[35] S. Theodridis, and K. Koutroumbas, "Template Matching", in *Pattern Recognition*, 4th ed., USA, 2009.
[36] *Penn Treebank* Project. (1999, February 2). [Online]. Available: www.cis.upenn.edu/~treebank/
[37] T. Buckwalter, "Buckwalter Arabic Morphological Analyzer Version 1.0", *Linguistic Data Consortium*, catalog number LDC2002L49, ISBN 1-58563-257-0, 2002.
[38] *Agency France Press. (2012, November)*. [Online]. Available: http://*www.afp.com*
[39] *Java Genetic Algorithms Package. (2012, November)*. [Online]. Available: http://*jgap.sourceforge.net*

Authors' Profiles

Ibrahim S. I. Abuhaiba is a professor at the Islamic University of Gaza, Computer Engineering Department. He obtained his Master of Philosophy and Doctorate of Philosophy from Britain in the field of document understanding and pattern recognition. His research interests include artificial intelligence, computer vision, image processing, document analysis and understanding, pattern recognition, information security, and computer networks. Prof. Abuhaiba published tens of original contributions in these fields in well-reputed international journals and conferences.



Mohammad F. Eltibi received his B.Sc. degree in computer engineering, Islamic University of Gaza, and master degree in computer engineering, Islamic University of Gaza, in 2013. He research interests include artificial intelligence.

How to cite this paper: Ibrahim S. I. Abuhaiba, Mohammad F. Eltibi, "Author Attribution of Arabic Texts Using Extended Probabilistic Context Free Grammar Language Model", International Journal of Intelligent Systems and Applications (IJISA), Vol.8, No.6, pp.27-39, 2016. DOI: 10.5815/ijisa.2016.06.04

