Dynamic Ensembles in Named Entity Recognition for Historical Arabic Texts

Muhammad Majadly
Department of Computer Science
University of Haifa
Haifa, Israel
mu.ib.mj@gmail.com

Tomer Sagi
Department of Information Systems
University of Haifa
Haifa, Israel
tsagi@is.haifa.ac.il

Abstract

The use of Named Entity Recognition (NER) over archaic Arabic texts is steadily increasing. However, most tools have been either developed for modern English or trained over English language documents and are limited over historical Arabic text. Even Arabic NER tools are often trained on modern web-sourced text, making their fit for a historical task questionable. To mitigate historic Arabic NER resource scarcity, we propose a dynamic ensemble model utilizing several learners. The dynamic aspect is achieved by utilizing predictors and features over NER algorithm results that identify which have performed better on a specific task in real-time. We evaluate our approach against state-of-the-art Arabic NER and static ensemble methods over a novel historical Arabic NER task we have created. Our results show that our approach improves upon the state-of-the-art and reaches a 0.8 F-score on this challenging task.

1 Introduction

Digitized historical literature is an essential resource in facilitating historical and social research. Information extraction, the task of automatically extracting structured information from digitized documents, has emerged as a method to scale the analysis of texts and allow the integration of information from different sources (e.g., (Ehrmann et al., 2020), (Ren et al., 2017)). One of the cardinal tasks in information extraction is NER, extracting entities from text and categorizing them into predefined categories. Numerous NER algorithms have been suggested, from rule-based approaches (Mesfar, 2007) to machine learning (ML) approaches e.g., (Zhou and Su, 2002). However, an overwhelming majority of the algorithms have been developed over modern English text. Rule-based approaches designed using modern English grammatical rules are irrelevant to Arabic (Shaalan, 2014). ML approaches rely on large modern English corpora sourced from international news outlets and social media. Training these tools to extract named entities from historical texts written in ancient Arabic requires large amounts of tagged text in the same language and preferably the same dialect, which are sorely missing.

In this work, we explore two avenues to mitigate the weakness of existing Arabic NER tools over historical Arabic texts (Shaalan, 2014). First, we utilize an approach named *dynamic prediction*, that was used in adjacent fields such as schema matching (Sagi and Gal, 2013), business process matching (Weidlich et al., 2013), and pattern recognition (Ko et al., 2008). In dynamic prediction, the results of different techniques are combined according to each result's (or component of the result) predicted quality. Here, we build an ensemble learning model based on dynamic prediction for NER algorithms. To alleviate the lack of resources in Arabic historical NER, we have created a novel dataset – the Bedaya Corpus and examine how training ML-based NER tools on historic Arabic impacts their performance instead of training them over modern Arabic text.

The contribution of this paper can, therefore, be summarized as follows. 1) We introduce a dynamic ensemble model for NER over historical Arabic texts. 2) We present Bedaya corpus, an Arabic historical dataset. 3) We perform a detailed empirical evaluation of our approach over baseline and state-of-the-art methods. We now provide background and preliminary definitions of NER and ensemble-learning in Section 2 and present related work in Section 3. In Section 4 we present our predictors and ensemble learning technique. In Section 5, we present the datasets. In Section 6, we report the results of our empirical evaluation, concluding our work in Section 7.

Proceedings of the Sixth Arabic Natural Language Processing Workshop, pages 115–125
Kyiv, Ukraine (Virtual), April 19, 2021.

2 Background

2.1 NER

Named Entity Recognition (NER) seeks to locate and classify named entities mentioned in unstructured text into categories such as person names, organizations, locations, events, and others. For example, the output for the sentence *"Romeo and Juliet meet in Verona."* is as follows.

> **[Romeo]**(Person) and **[Juliet]**(Person) meet in **[Verona]**(Location).

Some NER algorithms provide a *confidence value* (usually over $[0, 1]$) associated with each mapping. Others provide a set of confidence values for each token, one for each possible class. There are two broad approaches to constructing NER algorithms. *Rule based approaches*, rely on rules and patterns manually created in order to capture terms from input documents. For example, Mikheev et al. (Mikheev et al., 1999) suggest the following rule:

> **[Xxxx]** (sequence of capitalized words) + 'is' + 'a' + **[JJ*]** (sequence of zero or more adjectives) + **[REL]** (relative). then **[Xxxx]** is (Person), e.g., "**[John White]** is a beloved brother".

Rule-based systems achieve high precision but require a significant time investment to develop. Moreover, transferring rules from one language to another or between domains is very challenging (Jiang et al., 2016).

In *machine-learning (ML)-based approaches*, an algorithm containing rules or some other internal representation is learned from training examples (Mansouri et al., 2008). This approach can be further categorized into supervised, unsupervised, semi-supervised. Supervised approaches rely on manually tagged examples. Unsupervised approaches attempt to divide the tokens into similar groups, hopefully grouping the same class's entities. Semi-supervised approaches use a small number of tagged examples to guide the otherwise unsupervised process. Several features are used in ML-based approaches, such as part of speech, capitalized words, special marks (punctuation, numbers, dates, and titles), and word length. Recent NER systems rely on (deep) neural-networks over sequences of words (Li et al., 2020). Gazetteers, or entity dictionaries, are an essential resource for NER that support entity tagging (Zamin and Oxley, 2011) by allowing to look up words and phrases that are commonly used as named entities (e.g., (Shaalan and Raza, 2009), (Sajadi and Minaei, 2017)).

NER over Arabic text has proved much more challenging than other languages due to the complexity of Arabic morphology, absence of capital letters, and the lack of resources on which to train ML-based tools (Shaalan, 2014). For example, while performance on CONLL 2000, a common NER task in English has reached an F1-score of 97.3% (Liu et al., 2019), the best performance on a modern Arabic corpus ANERcorp only achieves an F1-score of 89.9% (Balla and Delany, 2020). Arabic texts can be categorized into classical Arabic, Modern Standard Arabic (MSA), and Arabic dialects (Shaalan, 2014). Historical Islamic literature is written in classical Arabic (Habash, 2010). According to (Hetzron, 1997), MSA differs from classical Arabic in vocabulary, syntax and styles. moreover they present 9 common morphonology changes that happened upon the time that lead for MSA from classical Arabic, thus changes leads to changes in syntax and words meanings. Researchers must consider these differences when using modern datasets and NER tools to build NER tools for historical Arabic texts.

2.2 Ensemble Learning

Ensemble learning algorithms (Definition 2.1) aim to combine the outputs of several base-algorithms (e.g., classifiers) to get better predictive performance. Ensemble methods have been demonstrated to provide superior predictive performance versus single algorithms on a variety of problems. The potential for performance improvement is higher when the algorithms' performance is diverse, such that different algorithms succeed and fail on different tasks (Oza and Russell, 2001).

Definition 2.1 (NER ensemble learning algorithm)**.** Let $D \subseteq \mathcal{D}$ be a set of training documents and let $f_m : T_D \rightarrow C$ be the expected mapping between each token in this set and a class (supervised labels). Let $\mathcal{F}$ be the set of all possible NER algorithms and let $F \subset \mathcal{F}$ be the set of base NER algorithms. Let M_D be the set of all such possible mappings for D and let $m_F \subset M_D$ be the set of NER algorithm results returned by F over D. A *NER ensemble learning algorithm* is a function $f : 2^{\mathcal{D}} \times M_D \times 2^{M_D} \rightarrow \mathcal{F}$. Given a specific training set D, an expected mapping f_m, a set of NER algorithms F,

and a set of NER algorithm results m_F, f outputs a NER algorithm.

Different ensemble learning approaches have been attempted. In voting based approaches, each classifier votes for a class, and the class with the most votes is chosen (Dietterich, 2000). Linear approaches assign a weight to each classifier, and the class with the highest combined score is then chosen. Several ensemble methods are based on heuristics. *Bagging* (Breiman, 1996) algorithms train each model in the ensemble using a randomly drawn subset of the training set. In contrast, *Boosting* (Schapire, 1990) algorithms gradually build the model by training each new model instance to highlight the cases that previous models misclassified.

However, the above mentioned approaches are trained once, and their combined algorithm remains static, regardless of base classifiers' results. This static approach can lead to unexpected results. For example, a reliable classifier utilizes a word's capitalization as a major signal. In a long sentence that is, for some reason, capitalized throughout (perhaps for emphasis), this classifier may decide that all tokens belong to one long, named entity. The result would appear wrong to human eyes but will be taken heavily into account due to the classifier's static high importance in the ensemble. In section 4.2, we present our dynamic approach.

3 Related work

Speck and Ngomo 2014 explore the use of NER ensemble learning in English, combining four NER algorithms using 15 different ensemble learning algorithms. They evaluate these methods over five different English language datasets and show that ensembles can reduce the error rate by an average of 40%. However, all of the ensemble learning methods evaluated were static methods. In this work, we employ predictors to assess the quality of NER results dynamically.

For Arabic NER, Abdallah et al. 2012 integrate an ML-based approach with a rule-based one. They train a decision tree model to combine the results, achieving an F1-score of 87.8%. (Sajadi and Minaei, 2017) learn a static ensemble for named entity recognition over classical Arabic text and rely on the Adaboost algorithm, an implementation of the multi-class boosting method. Using their novel NOORcorp (5) historic Arabic corpus, the system was evaluated over its ability to recognize three classes: location, person, organization. Ekbal and Bandyopadhyay 2010 propose a NER for Arabic based on Support Vector Machines (SVM) (Hearst et al., 1998) used to classify feature vectors for every word.

Recent NER systems rely on (deep) neural-networks over sequences of words. These systems infer features from raw sentences by using several layers of components allowing to represent higher levels of abstraction over the word sequence (see survey (Li et al., 2020)). The current state of the art are NER systems based on the contextual word embedding approach (Peters et al., 2018) and especially systems which employ BERT (Devlin et al., 2018), a novel pre-trained deeply bidirectional, unsupervised language representation, that takes into account the context for each occurrence of a given word. Al-Smadi et al., 2020 propose an Arabic NER system using a six-layer deep neural network model based on the transfer learning architectures among deep neural networks (Yosinski et al., 2014), (Devlin et al., 2018). The system achieves an overall F1-score of 90% Compared with their baseline BI-LSTM-CRF model which reached an overall F1-score of 73%. Their work is trained and tested over MSA, they used WikiFANE-Gold (Alotaibi and Lee, 2014) as a dataset that builds over the Arabic version of Wikipedia. While our work focus on historical text, and emphasizes (see 6.4) the non improving NER over historical Arabic text using MSA.

Dynamic prediction has been previously proposed for schema matching (Sagi and Gal, 2013) and pattern matching (Ko et al., 2008). To the best of our knowledge, it has not been used to combine NER algorithms, as proposed here.

4 Employing Predictors for Dynamic Ensemble Building

Static ensemble methods rely on the base NER algorithms' results to generate a NER algorithm that combines their results such that on every document set, the results of the base NER algorithms are combined in the same way. Predictors have been explored for schema matching (Sagi and Gal, 2013) and pattern recognition (Ko et al., 2008) as a method for fine-tuning the ensemble creation method to the task at hand. Predictors assess the result of each base algorithm on the task and provide a score. The ensemble model can use this score to determine whether or not the algorithm has succeeded in this specific occasion to identify

the correct class. Formally:

Definition 4.1 (Predictor)**.** Let D be a set of documents and let $\{T_D|\forall t \in T_D : t \in d \wedge d \in D\}$ be the set of tokens contained in these documents. Let C be a set of classes, and let $f : T_D \rightarrow C \times \mathbb{R}$ be a NER algorithm result assigning a real number and a class to each token. Let $\mathcal{N}$ be the set of all possible such NER algorithm results and let $n \in \mathbb{D}$ be some natural non-zero number, then a *predictor* is a function $f : \mathcal{N} \rightarrow \mathbb{R}^n$.

Thus, a predictor assesses the quality of the algorithm's outputs without knowing the expected results. Predictors are defined over different result cardinalities. *Dataset-level* predictors output a single number for the entire dataset. *Sentence-level* predictors output one number for each sentence, and *token-level* predictors output a number for each token. Furthermore, as mentioned in Section 2, a NER algorithm result may include a confidence score for each token for each of the possible classes. In this case the NER algorithm result (and predictor input) is a function $f : T_D \rightarrow (C \times \mathbb{R})^{|C|}$. Here, we propose the following predictors.

4.1 Predictors

Token-level Predictors Our token-level predictors are calculated over the NER algorithm's confidence rates for each token and class. Thus, an algorithm tasked with classifying tokens into four classes will report four confidence values for each token, representing the confidence in its prediction for every class. The first predictor is *max confidence rate token* (MCT), which takes the value of the maximum confidence rate per token. Usually, this is the reported confidence for the chosen class. This predictor indicates how confident the NER tool is in its outcome, which leads it to prefer NER tools with higher confidence rates. The second predictor is the *Difference confidence Rate Token* (DCT), which measures the difference between the token's maximal confidence and second-best confidence rate, another indication of how confident the NER tool is in its outcome.

Sentence-level Predictors We explore two approaches to constructing sentence-level predictors. The first two predictors rely on the confidence rate information as in the token-level predictors. The latter two rely on counting results. *Binary distance* (BD) measures the sum of distances between the reported confidence value and the closest binary value. BD indicates the number of confident tokens in a sentence. A higher value for confident tokens leads to a lower value of BD. For each token's sentence, the confidence value is rounded towards the closest binary value, and the difference is taken and summed. For example, in Figure 1, given the confidence values from two NER's output, the lower BD indicates a confident NER. This predictor's rationale is that good NER results are those in which the NER algorithm is confident and either marks a token as belonging to the class or not. Half-hearted values are penalized. Following a similar rationale, *difference confidence rate sentence* DCS is the sum of differences between the highest and second-highest confidence rate for every token in the sentence. Like DCT, DCS indicates the NER tool's confidence in its outcome. The second group of *sentence-level* predictors utilizes a counting representation of the sentence over the number of named entities in general or the number of named entities from a specific class. With these predictors, the ensemble model can prefer a NER Tool with good performance for recognizing a type of Named Entity or can ignore others that are bad at recognizing this type. *Sentence number of named entities* (SNN) counts the number of tokens marked as belonging to named entities in the sentence normalized over the sentence's length. We similarly define three additional predictors: *sentence number of persons* (SNP), *sentence number of organizations* (SNO), and *sentence Number of Locations* (SNL). Each predictor counts the number of tokens from their respective class. Finally, SLD measures the proportion of tokens associated with the most prevalent class in the sentence from all tokens in the sentence. Table 1 contains examples for predictor calculation over the sentence *"Sherlock Holmes lives on Baker Street"*.

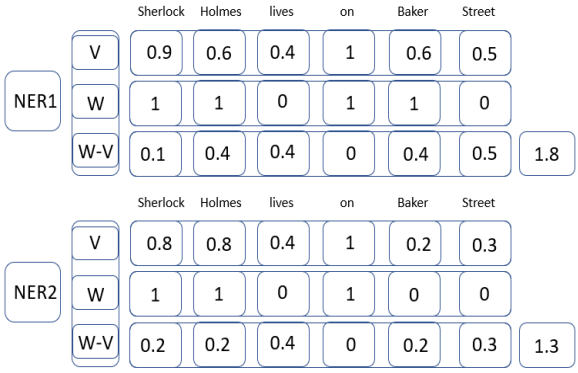

Figure 1: BD-sentence example.

Predictor Name	Example
MCT	If the confidence rate for *'Sherlock'* is 0.9 for *Person* class and 0.7, 0.8, and 0.3 for *Location*, *Organization*, and *Other* classes respectively, its MCT will be 0.9.
DCT	If the confidence rate for *'Sherlock'* is 0.9 for *Person* class and 0.7, 0.8, and 0.3 for *Location*, *Organization* and *Other* classes respectively, its DCT will be 0.9 - 0.8 = 0.1.
BD	In figure 1 we describe two NER tools NER_1 and NER_2. NER_2 is judged superior by BD since it is very confident for three tokens and only uncertain for two versus NER_1 that is only confident for two tokens.
DCS	If the confidence rate for every token in the sentence (six in our example) is 0.9 for the *Person* class and 0.7, 0.5, and 0.3 for the *Location*, *Organization*, and *Other* classes respectively. DSC for this sentence will be $\sum_{n=1}^{6}(0.9 - 0.7)$.
SNN	If NER_1 recognizes just *Sherlock* and *Holmes* as *Person* (or another NE) the predictor score will be 2 for this NER tool.
SNP, SNO, SNL	If NER_1 recognizes just *Sherlock* and *Holmes* as *Person* (or another NE) the predictor score will be 2 for *Person*, and zero for *Organization* and *Location*.
SLD	If NER_1 recognizes *Sherlock*, *Holmes* as *Person*, and *Baker* as *Location* (or another named entity) the predictor score will be 2/3.

Table 1: Predictor examples.

4.2 A Dynamic Ensemble Model (DEM) for NER Algorithms

Figure 2 shows the workflow of our technique. After training NER base algorithms over the training data, predictors are calculated over the results. Both the predictor value and the raw NER algorithm results serve as inputs for the ensemble-learning model, aiming to learn an ensemble method. When the learned method is employed at test/usage, the dataset is first fed into the NER tools to get the NER results, then the predictors are calculated, and the learned ensemble method combines both to give a class label for each token.

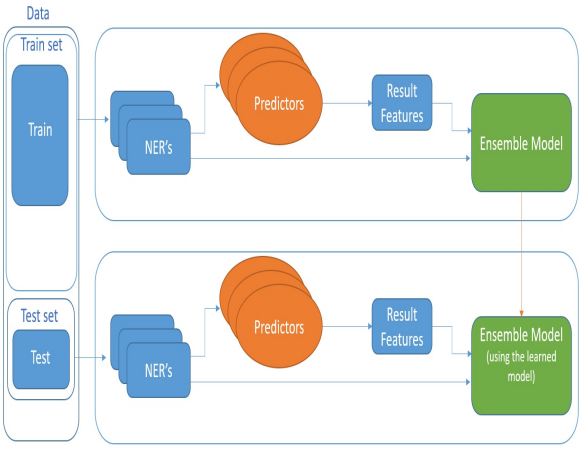

Figure 2: Workflow of the proposed approach.

5 Datasets

There is an apparent lack of Arabic corpora (Abouenour et al., 2010). Moreover, few of these corpora have been made freely and publicly available for research purposes (Bies et al., 2012). while others are available but under licenses (Mostefa et al., 2009) Thus, researchers often rely on private corpora, which precludes reproducing previous work and comparing its performance over a consistent benchmark.

In this work we use three corpora, NoorCorp (Sajadi and Minaei, 2017) based on a different historical book circa 800 AD, ANERcorp (Benajiba et al., 2007), a modern corpus which has become a standard in Arabic NER works. ANERcorp is based upon web-documents written in modern Arabic collected from 316 articles from newspapers such as bbc[1] and aljazeera[2]. Table 2 compares the corpora's token counts and class distributions. The datasets are tagged into four classes, *Person*, *Organization*, *Location*, and *Other*. The latter is assigned to tokens that do not belong to the first three classes. The third corpus is a historic one contributed in this work.

[1] https://www.bbc.com/arabic
[2] https://www.aljazeera.net/

Bedaya corpus

The scarcity of public Historic NER datasets has led us to create a new corpus. The *Bedaya* corpus[3], is based on Ibn Kathir's historical Arabic opus Al-Bidāya wa-al-Nihāya (The Beginning and The End). Kathir's ten volumes contain an extensive description of the world's history from its creation and are widely used in Islamic studies. The dataset is taken continuously from the 7th part of the book. It contains 20,500 tokens with 5,664 different tokens. Named entities were annotated manually by one of the authors and verified by an Arabic language expert that reviewed 50% of the dataset with a 99.998% inter-rater agreement.

Over a 80-20% training-test partition (Sajadi and Minaei, 2017) over NOORCorp dataset reached approximately 97% F1-score. In our evaluation of the NOORCorp dataset, we challenge our tools with a harder task, which is to train over one dataset and to be tested on a different one. [4] Thus, we ensure the generality of our algorithm and obtain enough space for ensemble algorithms improving.

Class	Noor	Bedaya	ANER
Loc.	1,320 (1.2%)	328 (1.6%)	6,426 (4.8%)
Pers.	15,873 (14%)	2,840 (14%)	5,036 (3.4%)
Org.	1,932 (1.7%)	336 (1.6%)	3,402 (2.3%)
O	93,055 (83%)	17,016 (83%)	133,768 (90%)
All	112,180	20,520	148,632
Voc size	19,319	5,664	33,208

Table 2: Dataset Token Counts and Class Distribution.

6 Experiments

We now present a series of empirical evaluations examining the following questions. Which predictors are the most correlated with our desired quality measures (§ 6.2)? Does our dynamic ensemble approach outperform the baseline approaches of using a single NER algorithm and using static ensemble learning (§ 6.3)? What is the performance impact of training NER algorithms over modern Arabic text versus historic Arabic for a historic Arabic NER task (§ 6.4)? We begin by introducing the experimental setup.

6.1 Setup

6.1.1 Ensemble Algorithm

We employ C4.5 (Quinlan, 2014) as our ensemble learning algorithm. C4.5 generates a decision tree, a learned classifier that uses a tree-like graph or model of decisions and possible outcomes. Each internal node represents an "examination" on an attribute, each branch represents the outcome of the *examination*, and each leaf node represents a class label. The paths from the root to the leaves represent classification rules. Our DEM approach utilizes the J48 implementation of C4.5 in Weka (Hall et al., 2009).

6.1.2 NER Tools

The following are the NER algorithms used in this work. **Pattern** is a rule-based tool. The tool uses pattern comparison with a small lexicon containing the 50 most prevalent words for every named entity-tag in the historical Arabic dataset NoorCourp. Additionally, the tool employs six rules related to the Arabic language and historic Arabic specifications taken from (Sajadi and Minaei, 2017) and (Shaalan and Raza, 2009) with some minor changes. Table 3 presents the rules we use.

The rest of the tools utilize an ML approach. **CRF** uses the conditional random field technique (Lafferty et al., 2001), a discriminative model for sequence labeling. It models the dependency between each sample and the entire input sequence. CRF uses the following extracted features: Part of speech (Stanford POS tagger (Manning et al., 2014)), special flag marks (is punctuation?, is a number?), words, word parts (last three, two, one letter), word length, and nearby words features. **LSTM-CRF** (Huang et al., 2015) combines a bidirectional long short-term memory neural network (Hochreiter and Schmidhuber, 1997) with CRF. Long short-term memory (LSTM) is an artificial recurrent neural network (Rumelhart et al., 1986). Unlike standard feedforward neural networks, LSTM has feedback connections and can process entire sequences of data. **Polyglot** (Al-Rfou et al., 2015) is a multilingual, semi-supervised ML-based NER that uses word embedding. Word embeddings are representations of words acquired by using vast amounts of raw text. These representations capture information about words' syntactic functionality and semantics. Polyglot embeddings

[3] available at https://github.com/muhammad-majadly/Bedaya-dataset

[4] It should be noted, that in preliminary work we were able to reproduce these same results over an 80-20 split of NOORCorp using both static and dynamic ensembles.

Rule Name	Rule Format
Person Rule 1	word1[noun] + (son of) + word2[noun] then word1 and word2 is a person.
Person Rule 2	(equivalent words to Mr./MRs) + word[noun] word is a person.
King Rule	king [or equal words like 'Khalifa'] + word[noun] word is a person.
Location Rule	word1[pers] + (from/born) + word2[noun] then word2 is a Location.
Organization Rule	word1[first word of tribes' names] + word2[noun] then word1 and word2 is an organization.
Same NER Rule	word1[NE=x] + (and/like) + word2[noun] then word2 is x.

Table 3: Pattern-NER rules.

are trained on Wikipedia, the text consists of the most frequent 100K words for each language, and the word representation consists of 64 dimensions. The polyglot model learns a simple three-layer neural network using the word embedding representation for classification, taking as features the embeddings for the words in a nearby interval of text around each word.

The last three algorithms (**SVM**, **LR**, and **DT**) use three different machine learning classifiers over the same feature space: part of speech tags, and word length. SVM (Chang and Lin, 2011) works by finding a hyperplane in N-dimensional space (N number of features) that distinctly classifies the data points. Logistic Regression (LR) (Hosmer Jr et al., 2013) uses a logistic function to model a binary dependent variable. Decision Tree (DT) (Quinlan, 1986) is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes. To ensure diversity in our ensemble approach, we chose these non-sequence ML-based algorithms that consider token-level information together with the sequence-based algorithms CRF, LSTM-CRF, and Polyglot. The classifiers use the following extracted features: Part Of Speech tags, the word itself, words, and word length.

6.1.3 NER Diversity

Ensemble methods have been demonstrated to provide superior predictive performance versus single algorithms on a variety of problems. The potential for performance improvement is higher when the algorithms' performance is diverse, such that different algorithms succeed and fail on different tasks (Oza and Russell, 2001). To ensure diversity, we choose both rule-based and machine learning algorithms, and within the latter group, a diverse set of principles and models. Empirical diversity was measured over this task by using the Pearson product-moment correlation coefficient (PPMCC) (Steel et al., 1960) between each pair of NER algorithms results over the Bedaya dataset. An absolute value of 1.0 indicates a perfect correlation, and 0.0 indicates no correlation at all. CRF is positively correlated with LSTM-CRF at 0.7 and less correlated with POLYGLOT at 0.38. Polyglot is highly correlated with LR with 0.7. DT with LR and SVM with LSTM-CRF are very correlated at 0.84. Pattern and LR have a 0.78 correlation between them. The most correlated tools are SVM NER and DT with 0.96.

6.1.4 Measures

We measure NER performance using the commonly employed precision, recall, and F1-score. *Precision* is the portion of correctly classified tokens (true positives) among the tokens classified as belonging to a named entity by the NER system (predicted positive). In contrast, *recall* is the fraction of true positives predicted by the NER system among the total number of tokens belonging to named entities in the dataset (actual positive). *F1-score* is the harmonic mean of precision and recall.

Oracle We use the concept of an oracle (Ko et al., 2008) to measure the upper limit for the ensemble of classifiers' performance. An oracle is the union of true positive results of all classifiers. If one classifier can correctly classify a given input, then an ensemble of classifiers that can classify this input will not be better than the oracle.

Definition 6.1 (Oracle)**.** Let T be a token and let C be the set of classes where $c \in C$ is the expected class of token T and let $\mathcal{O} \in C$ be the class for tokens labeled as a non named entity. The *oracle*

over NER tools $[N_1, ..., N_n]$ is defined as follows.

$$Oracle(T) = \begin{cases} c & \exists i \in [1, .., n] | N_i(T) = c \\ \mathcal{O} & \text{else} \end{cases}$$

The oracle's performance over Bedaya dataset is **94%** Precision, **75%** Recall, and F1-score **83%**.

6.2 Evaluating Predictors

To assess the quality of a predictor, we use the correlation between the predictor value and the result's eventual quality. In our case, we measure it using PPMCC and Point Biserial Correlation (PBC) (Tate, 1954). PPMCC can be used between two continuous variables to assess their linear relationship. PBC is used to measure the correlation between a binary (is entity?) and continuous variable (prediction). We split our dataset (Bedaya) into sentences for sentence-level predictors, run all base NER algorithms over each sentence, and compare the predictor value over each sentence to precision and recall calculated on it using PPMCC. For token-level predictors (DCT and MCT), we use PBC with true/false token predictions. Correlation measurement is done on two levels, for all the tools together and for each tool individually. Predictors correlated with all the tools are more general and make the system more effective. On the other hand, predictors correlated with one tool can give helpful information to the ensemble model about a specific tool. Table 4 summarizes correlation results for predictors with all the tools reporting their correlation over 10,000 tokens and 500 sentences using seven different NER tools with quality measures precision and recall. The best-correlated sentence-level predictor is DCS, and the best token-level predictor is MCT. A cross-correlational analysis revealed that BD is correlated strongly with DCS, SNN is strongly correlated with SNP, and DCT is correlated with MCT. Other predictor pairs are weakly correlated.

6.3 Evaluating DEM

When applying NER algorithms over Bedaya (Table 5) with NoorCorp (Table 5) as a train dataset, somewhat surprisingly, classic CRF achieves the best score while the state-of-the-art ML-model LSTM-CRF NER is only the fourth-best tool. Using DEM results in an ensemble that outperforms CRF by three percentage points on recall and 2.4 percentage points on F1-score. In table 6 DEM is compared with the following static ensemble methods: C4.5(Quinlan, 2014), Adaboost (M1)(Freund and Schapire, 1996) and bagging (BG) (Breiman, 1996) with RandomForest (Breiman, 2001), REPTree, and C4.5 as the base classifiers. The rest of the static ensemble methods are Logistic Model Trees (LMT) (Landwehr et al., 2005), Sequential Minimal Optimization (SMO) (Hastie and Tibshirani, 1998), and Naive Bayes (NB) (John and Langley, 1995). As we chose the C4.5 algorithm as our dynamic ensemble method, we can see the impact of utilizing C4.5 with predictors versus using it without predictors by comparing the other eight static models' results. It seems that in this setup and over this task, DEM can improve both the recall and the resulting F1-score by approximately three percentage points compared with the best static method's performance. When comparing to the Oracle, one can see that DEM manages to close

	Pearson		**Biserial**	
Predictor	Precision	Recall	R_{pb}	p_{value}
BD	-0.181	-0.233		
DCS	**0.182**	**0.255**		
SNN	0.121	0.147		
SNP	0.096	0.120		
SNO	0.110	0.120		
SNL	0.040	0.070		
SLD	0.020	-0.014		
DCT			0.42	< 0.001
MCT			0.40	< 0.001

Table 4: Correlation results for our proposed predictors. Pearson Correlation with Precision and Recall, and Biserial Correlation with true/false token prediction.

Type	**Precision**	**Recall**	**F1**
CRF	90.0%	68.0%	77.4%
LSTMCRF	81.3%	64.6%	73.0%
SVM	86.3%	66.0%	74.8%
LR	88.3%	50.0%	63.8%
DT	85.0%	65.3%	75.2%
Pattern	84.0%	30.0%	42.0%
PolyGlot	59.0%	39.0%	47.0%
DEM	**90.8%**	**71.0%**	**79.8%**
Oracle	**94.0%**	**75.0%**	**83.0%**

Table 5: Performance of DEM vs. single NER Tools.

half of the distance between the best performing NER and the Oracle in terms of F1-score. In contrast, the next best ensemble method only manages to close a quarter of this distance.

Type	Precision	Recall	F1
NB	86.50%	67.90%	76.10%
SMO	91.00%	68.20%	78.00%
LMT	91.00%	68.30%	78.00%
BG (REPTree)	91.00%	68.20%	78.00%
BG (randomForest)	91.10%	68.40%	78.14%
BG (C4.5)	91.08%	68.34%	78.09%
C4.5	91.10%	68.30%	78.07%
M1(c4.5))	91.10%	68.30%	78.07%
DEM	**90.80%**	**71.00%**	**79.80%**
Oracle	**94.00%**	**75.00%**	**83.00%**

Table 6: Performance of DEM vs. Static Models.

6.4 Modern Text Versus Historical Text

Of the seven NER algorithms we evaluate, five are ML-based. Here, we evaluate their performance when trained on modern and historical datasets. Figure 3 shows the algorithms' performance when tested over the Bedaya dataset and trained over two different datasets: ANER (Benajiba et al., 2007) a modern corpus, NoorCorp (Sajadi and Minaei, 2017) a historical corpus, and over both. The results demonstrate the importance of historic Arabic datasets as algorithms trained on historic data achieve better performance than when trained upon modern data. When trained over a mix of historical and modern data, didn't lead to improvement, thus relatively poor performance highlights the need to find better ways to transfer learning from modern to historical texts. Table 7 shows some outputs of tokens from Bedaya dataset by CRF-NER tool (the best tool in our ensemble model) when it trained over modern and historic text.

7 Conclusion and Future Work

We have proposed a dynamic ensemble model for NER and demonstrated its efficacy over a historical Arabic task. We have shown that training ML-based NER algorithms over modern Arabic text negatively impacts their performance over historical text, even when the training is combined with historical sources. This result highlights the need for more tagged historical datasets, such as the Bedaya corpus contributed by this work. In future

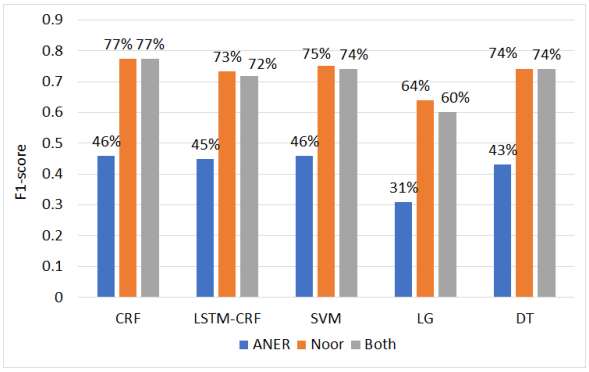

Figure 3: Comparative results of NER trained over modern (ANER) and historic (NoorCorp) data.

Token	CRF (MSA)	CRF (Historic)	Class
'talha'	O	PERS	PERS
'mnaf'	O	PERS	PERS
'almgazi'	O	LOC	LOC
'alhodybya'	O	LOC	LOC
'alshabha'	O	ORG	ORG
'bniAMR'	O	ORG	ORG

Table 7: Output Examples for CRF-NER when trained over modern and historical text.

work, we intend to enhance the dynamic ensemble model by exploring additional predictors and alternative ensemble learning methods. We further intend to explore different ways to utilize modern Arabic corpora in historical text analysis tasks.

References

Sherief Abdallah, Khaled Shaalan, and Muhammad Shoaib. 2012. Integrating rule-based system with classification for Arabic named entity recognition. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '12)*, pages 311–322. Springer.

Lahsen Abouenour, Karim Bouzoubaa, and Paolo Rosso. 2010. Using the yago ontology as a resource for the enrichment of Named Entities in Arabic WordNet. In *Proceedings of The Seventh International Conference on Language Resources and Evaluation (LREC 2010) Workshop on Language Resources and Human Language Technology for Semitic Languages*, pages 27–31.

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of the society of industrial applied mathematics (SIAM '15) International Conference on Data Mining*, pages 586–594. SIAM.

Mohammad Al-Smadi, Saad Al-Zboon, Yaser Jararweh, and Patrick Juola. 2020. Transfer learning for arabic named entity recognition with deep neural networks. *IEEE Access*, 8:37736–37745.

Fahd Alotaibi and Mark Lee. 2014. A hybrid approach to features representation for fine-grained arabic named entity recognition. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 984–995.

Husamelddin AMN Balla and Sarah Jane Delany. 2020. Exploration of approaches to arabic named entity recognition.

Yassine Benajiba, Paolo Rosso, and José Miguel Benedíruiz. 2007. Anersys: An Arabic named entity recognition system based on maximum entropy. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '07)*, pages 143–153. Springer.

Ann Bies, Denise DiPersio, and Mohamed Maamouri. 2012. *Linguistic resources for Arabic machine translation*. John Benjamins, Amsterdam, The Netherlands.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST '11)*, 2(3):1–27.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, (MCS '00), page 1–15, Berlin, Heidelberg. Springer-Verlag.

Maud Ehrmann, Matteo Romanello, Alex Flückiger, and Simon Clematide. 2020. Extended overview of clef hipe 2020: named entity processing on historical newspapers. In *CLEF 2020 Working Notes. Conference and Labs of the Evaluation Forum*, volume 2696. CEUR.

Asif Ekbal and Sivaji Bandyopadhyay. 2010. Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical, Computer, and Systems Engineering*, 4(2):155–170.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nizar Y Habash. 2010. Introduction to Arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Trevor Hastie and Robert Tibshirani. 1998. Classification by pairwise coupling. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, (NIPS '97), page 507–513, Cambridge, MA, USA. MIT Press.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Robert Hetzron. 1997. *The Semitic Languages*. Taylor & Francis.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*, volume 398. John Wiley & Sons.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ridong Jiang, Rafael E Banchs, and Haizhou Li. 2016. Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*, pages 21–27. Association for Computational Linguistics (ACL '16).

George H. John and Pat Langley. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, (UAI '95), page 338–345, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Albert HR Ko, Robert Sabourin, and Alceu Souza Britto Jr. 2008. From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*, 41(5):1718–1731.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Niels Landwehr, Mark Hall, and Eibe Frank. 2005. Logistic model trees. *Machine learning*, 59(1-2):161–205.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. Gcdt: A global context enhanced deep transition architecture for sequence labeling. *arXiv preprint arXiv:1906.02437*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security IJCSNS*, 8(2):339–344.

Slim Mesfar. 2007. Named entity recognition for Arabic using syntactic grammars. In *International Conference on Application of Natural Language to Information Systems (NLDB '07)*, pages 305–316. Springer.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL '99)*.

Djamel Mostefa, Mariama Laïb, Stéphane Chaudiron, Khalid Choukri, and G Chalendar. 2009. A multilingual named entity corpus for Arabic, English and French. *MEDAR*, 2009:2nd.

Nikunj Chandrakant Oza and Stuart Russell. 2001. *Online ensemble learning*. University of California, Berkeley.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

J Ross Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.

Xiang Ren, Jiaming Shen, Meng Qu, Xuan Wang, Zeqiu Wu, Qi Zhu, Meng Jiang, Fangbo Tao, Saurabh Sinha, David Liem, et al. 2017. Life-inet: A structured network-based knowledge exploration and analytics system for life sciences. In *Proceedings of ACL 2017, System Demonstrations*, pages 55–60.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Tomer Sagi and Avigdor Gal. 2013. Schema matching prediction with applications to data source discovery and dynamic ensembling. *The VLDB Journal*, 22(5):689–710.

Mohamad Bagher Sajadi and Behrooz Minaei. 2017. Arabic named entity recognition using boosting method. In *2017 Artificial Intelligence and Signal Processing Conference (AISP '17)*, pages 281–288, Shiraz, Iran. IEEE.

Robert E Schapire. 1990. The strength of weak learnability. *Machine learning*, 5(2):197–227.

Khaled Shaalan. 2014. A survey of Arabic named entity recognition and classification. *Computational Linguistics*, 40(2):469–510.

Khaled Shaalan and Hafsa Raza. 2009. Nera: Named entity recognition for Arabic. *Journal of the American Society for Information Science and Technology*, 60(8):1652–1663.

René Speck and Axel-Cyrille Ngonga Ngomo. 2014. Ensemble learning for named entity recognition (iswc '13). In *International semantic web conference*, pages 519–534. Springer.

Robert George Douglas Steel, James Hiram Torrie, et al. 1960. Principles and procedures of statistics.

Robert F Tate. 1954. Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3):603–607.

Matthias Weidlich, Tomer Sagi, Henrik Leopold, Avigdor Gal, and Jan Mendling. 2013. Predicting the quality of process model matching. In *Business Process Management (BPM '13)*, pages 203–210. Springer.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Norshuhani Zamin and Alan Oxley. 2011. Building a corpus-derived gazetteer for named entity recognition. In *International Conference on Software Engineering and Computer Systems*, pages 73–80. Springer.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.