Morphology-based Entity and Relational Entity Extraction Framework for Arabic

Amin Jaber* — **Fadi A. Zaraket****

* *Purdue University, West Lafayette, IN*
** *American University of Beirut, Beirut 1107 2020, Lebanon*

ABSTRACT. *Rule-based techniques to extract relational entities from documents allow users to specify desired entities with natural language questions, finite state automata, regular expressions and structured query language. They require linguistic and programming expertise and lack support for Arabic morphological analysis. We present a morphology-based entity and relational entity extraction framework for Arabic (*MERF*).* MERF *requires basic knowledge of linguistic features and regular expressions, and provides the ability to interactively specify Arabic morphological and synonymity features, tag types associated with regular expressions, and relations and code actions defined over matches of subexpressions.* MERF *constructs entities and relational entities from matches of the specifications. We evaluated* MERF *with several case studies. The results show that* MERF *requires shorter development time and effort compared to existing application specific techniques and produces reasonably accurate results within a reasonable overhead in run time.*

RÉSUMÉ. *Les techniques à base de règles pour extraire des entités permettent de spécifier les entités souhaitées en utilisant des questions de langage naturel, des automates à états finis, des expressions régulières et des instructions d'extraction de données. Ils nécessitent des expertises en linguistique et en programmation, et ne soutiennent pas l'analyse morphologique de l'arabe. On présente pour l'arabe un cadre d'extraction d'entité renforcé par l'analyse morphologique (*MERF*). Il exige des connaissances de base des caractéristiques linguistiques et des expressions régulières, et fournit la possibilité de spécifier de façon interactive des fonctionnalités de morphologie et synonymie arabes, des types de tag associés avec des expressions régulières, et des relations et actions de code définies sur les correspondances de sous-expressions.* MERF *construit des entités relationnelles à partir des correspondances des spécifications. On évalue* MERF *avec des études de cas. Les résultats montrent que* MERF *nécessite un effort de développement plus court par rapport aux techniques existantes et produit des résultats raisonnablement précis avec une surcharge raisonnable en temps d'exécution.*

KEYWORDS: *Arabic, information extraction, natural language processing, tagging.*

MOTS-CLÉS : *Arabe, extraction d'information, traitement du langage naturel, marquage.*

1. Introduction

Computational Linguistics (CL) is concerned with building accurate linguistic computational models. *Natural Language Processing* (NLP) is concerned with automating the understanding of natural language. CL and NLP tasks range from simple ones such as spell checking and typing error correction to more complex tasks including *named entity recognition* (NER), *cross-document analysis*, *machine translation*, and *relational entity extraction* (Linckels and Meinel, 2011; Ferilli, 2011). Entities are elements of text that are of interest to an NLP task. Relational entities are elements that connect entities. *Annotations* relate chunks of text to *labels* denoting semantic values such as entities or relational entities. We refer to annotations and labels as *tags* and *tag types*, respectively, in the sequel.

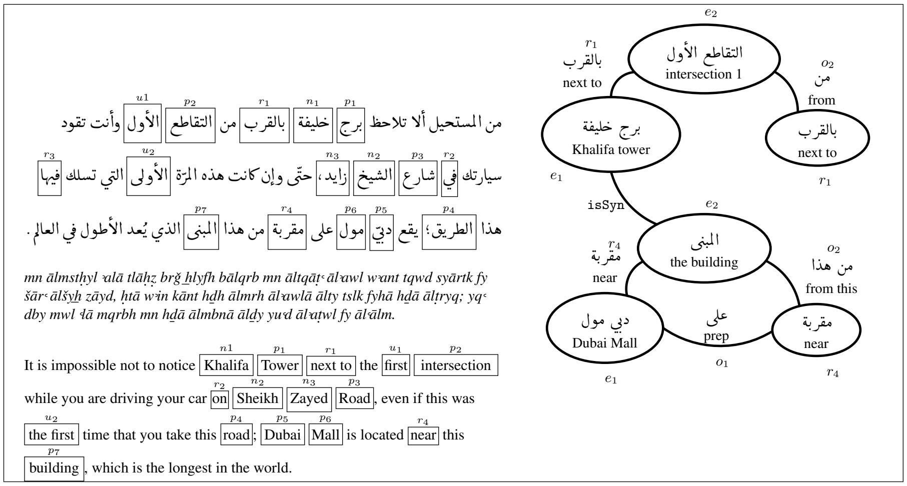
Supervised and unsupervised empirical learning techniques tackle NLP and CL tasks. They employ machine learning without the need to manually encode the requisite knowledge (Soudi *et al.*, 2007). Supervised learning techniques require training corpora annotated with *correct* tags to learn a computational model. Supervised and unsupervised techniques require annotated reference corpora to evaluate the accuracy of the technique using metrics such as precision and recall (Marcus *et al.*, 1993; Maamouri *et al.*, 2004; Xue *et al.*, 2005).

Researchers build training and reference corpora either manually, incrementally using learning techniques, or using knowledge-based annotation techniques that recognize and extract entities and relational entities from text. Knowledge-based techniques use linguistic and rhetorical domain specific knowledge encoded into sets of rules to extract entities and relational entities (Soudi *et al.*, 2007). While existing annotation, entity, and relational entity extraction tools exist (Chiticariu *et al.*, 2010; Atzmueller *et al.*, 2008; Urbain, 2012; Settles, 2011; Müller and Strube, 2006; Stenetorp *et al.*, 2012), most of them lack Arabic language support, and almost all of them lack Arabic morphological analysis support (Habash and Sadat, 2006). Fassieh (Attia *et al.*, 2009) is a *commercial* Arabic annotation tool with morphological analysis support and text factorization. However, this tool lacks support for entity and relational entity extraction.

Figure 1 illustrates the target of MERF using the directions to Dubai Mall example[1]. The figure also presents a transliteration and an English translation of the Arabic text. The framed words in the text are entities referring to names of people (n_1, n_2, n_3), names of places ($p_1, \ldots, p_7$), relative positions ($r_1, \ldots, r_4$), and numerical terms (u_1, u_2). We would like to extract those entities, and then extract the relational entities forming the graph in Figure 1 where vertices express entities, and edges represent the relational entities.

In this paper, we present MERF, a morphology-based entity and relational entity extraction framework for Arabic text. MERF provides a user-friendly interface where the user defines tag types and associates them with regular expressions over Boolean

[1] Text taken from the Dubai Mall website `http://www.thedubaimall.com/ar/`.



من المستحيل ألا تلاحظ برج خليفة بالقرب من التقاطع الأول وأنت تقود سيارتك في شارع الشيخ زايد، حتّى وإن كانت هذه المرّة الأولى التي تسلك فيها هذا الطريق؛ يقع دبيّ مول على مقربة من هذا المبنى الذي يُعد الأطول في العالم.

mn ālmstḥyl ʾalā tlāḥẓ brǧ ẖlyfh bālqrb mn āltqāṭʿ ālʾawl wʾant tqwd syārtk fy šārʿ ālšyẖ zāyd, ḥtā wʾin kānt hḏh ālmrh ālʾawlā ālty tslk fyhā hḏā ālṭryq; yqʿ dby mwl ʿlā mqrbh mn hḏā ālmbnā ālḏy yuʿd ālʾaṭwl fy ālʿālm.

It is impossible not to notice Khalifa Tower next to the first intersection while you are driving your car on Sheikh Zayed Road, even if this was the first time that you take this road; Dubai Mall is located near this building, which is the longest in the world.

Figure 1. *Direction example with Arabic text, annotated with entities, transliteration, translation, and extracted relational entities in a graph.*

formulae. A Boolean formula is defined by a term, negation of a term, or disjunction of terms. Terms are matches to Arabic morphological features including prefix, stem, suffix, part of speech (POS) tags, gloss tags, extended synonym tags, and semantic categories. For example, entity p_1 in Figure 1 has a "place" semantic category. MERF regular expressions support operators such as concatenation, zero or one, zero or more, one or more, up to M repetitions where M is a non-zero positive integer, and logical conjunction and disjunction. For example, the sequence between p_1 and p_2 matches a regular expression re that requires two semantic place categories with a place-preposition POS tag (r_1) in between.

An editor allows the user to associate an action with each subexpression. The user specifies the action with C++ code and uses an API to access information related to the matches such as text, position, length, morphological features, and numerical value. Each regular expression is associated with a named identifier to form a *local grammar* like structure (Traboulsi, 2009). A relation definition GUI allows the user to provide relational tuples where each tuple has a source, a destination and an edge label. The user uses the regular expression identifiers to define the relational tuple elements. For example, the relation between e_1, e_2 and r shown in Figure 1 is a match of a relational tuple over the components of re. We refer to regular expressions and Boolean formulae as expressions and formulae, respectively. We also refer to expressions as rules when used in a grammar context; e.g. when used with an identifier.

MERF takes an Arabic text and the local grammar defined by the Boolean formulae and the regular expressions. MERF computes the morphological solutions of the input text then computes matches to the Boolean formulae therein. MERF then generates a *non-deterministic finite state automata* (NDFSA) for each expression and simulates it with the sequence of Boolean formulae matches to compute the regular expression matches. MERF generates executable code for the actions associated with the regular expressions, compiles, links, and executes the generated code as shared object libraries. Finally, MERF constructs the semantic relations and cross-reference between entities. MERF also provides visualization tools to present the matches, and estimate their accuracy with respect to reference tags.

This work significantly extends Jaber and Zaraket (2013) that allows for manual, and morphology annotation. MERF enables a user to incrementally create complex annotations for Arabic based on automatic extraction of morphological tags through a user-friendly interactive interface. MERF has the following advantages.

– MERF provides a novel and intuitive visual interface to build formulae over morphological features, build regular expressions over the resulting formulae, and thereafter compute automatic tags.

– To our knowledge, this morphology-based framework is the first for Arabic entity and relational entity extraction.

– MERF provides the user with the ability to rapidly create annotated Arabic text corpora with sophisticated morphology-based tags.

In MERF, we make the following contributions.

– MERF enables the user to define relations in a simple manner and automatically detects relational entities matching the user defined relations.

– MERF enables the user to associate subexpressions with code actions, and executes the code action when a corresponding match is found. It also provides an API to enable access to match features such as text, position, length, numerical value, and morphological features.

– MERF enables the user to tag words based on a novel light Arabic WordNet relation that leverages the synonym Syn^k feature.

– MERF is open source and available online for the research community under `https://github.com/codelogicanalysis/atmine`.

The rest of the paper is structured as follows. Section 2 introduces Arabic morphological analysis and its important role in Arabic NLP. Section 3 explains the methodology of MERF. Section 4 presents MERF components. Section 5 presents MERF GUI. Section 6 presents and discusses related work. Section 7 presents the evaluation results. Finally, we conclude and discuss future work in Section 8.

2. Background: Morphological Analyzer

Morphological analysis is key to Arabic NLP due to the exceptional degree of ambiguity in writing, the rich morphology, and the complex word derivation system (Al-Sughaiyer and Al-Kharashi, 2003; Shahrour *et al.*, 2016; Pasha *et al.*, 2014). Short vowels, also known as diacritics, are typically omitted in Arabic text and inferred by readers (Habash and Sadat, 2006). For example, the word بن*bn* can be interpreted as بُن*bon* ("coffee") with a *damma* diacritic on the letter بـ *b* or بِن*bin* ("son of") with a *kasra* diacritic on the letter بـ *b* .

Morphological analysis is required even for tokenization of Arabic text. The position of an Arabic letter in a word (beginning, middle, end, and standalone) changes its visual form. Some letters have non-connecting end forms which allows visual word separation without the need of a white space separator. For example, the word ياسمين*yāsmyn* can be interpreted as the "Jasmine" flower, as well as يا (the calling word) followed by the word سمين (obese). Consider the sentence ذهبالمدرسة الوَلدالَى *ḏhb alwald-ilā 'lmdrsh* ("the kid went to school"). The letters د and ى have non-connecting end of word forms and the words الولد,الى, and المدرسة are visually separable, yet there is no space character in between. Newspaper articles with text justification requirements, SMS messages, and automatically digitized documents are examples where such problems occur.

MERF is integrated with *Sarf*, an in-house open source Arabic morphological analyzer based on finite state transducers (Zaraket and Makhlouta, 2012b). Given an Arabic word, Sarf returns a set of morphological solutions. A word might have more than one solution due to multiple possible segmentations and multiple tags associated with each word. A morphological solution is the internal structure of the word composed of several morphemes including *affixes* (*prefixes* and *suffixes*), and a *stem*, where each morpheme is associated with tags such as POS, gloss, and category tags (Al-Sughaiyer and Al-Kharashi, 2003; Habash, 2010).

Prefixes attach before the stem and a word can have multiple prefixes. Suffixes attach after the stem and a word can have multiple suffixes. Infixes are inserted inside the stem to form a new stem. In this work we consider a set of stems that includes infix morphological changes. The part-of-speech tag, referred to as POS, assigns a morphosyntactic tag for a morpheme. The gloss is a brief semantic notation of morpheme in English. A morpheme might have multiple glosses as it could stand for multiple meanings. The category is a custom tag that we assign to multiple morphemes. For example, we define the `Name of Person` category to include proper names.

	Prefixes			Stem	Suffix
Data	فَ*fa*	سَ*sa*	يَ*ya*	أْكُل*ʾakul*	ها*hā*
POS	CONJ+	FUT+	IV3MS+	VERB_IMPERFECT	IVSUFF_DO:3FS
Gloss	and/so	will	he/it	eat/consume	it/them/her
index	10			13	16
length	3			3	2

Table 1. *Sample solution vector for* فَسَيَأْكُلها*fasayaʾakulhā* .

We denote by $\mathcal{S}$, $\mathcal{P}$, $\mathcal{X}$, POS, $GLOSS$, and CAT, the set of all stems, prefixes, suffixes, POS, gloss, and user defined category tags, respectively. Let $T = \langle t_1, t_2, \dots, t_M \rangle$ be a set of Arabic words denoting the text documents. MERF uses Sarf to compute a set of morphological solutions $M(t) = \{m_1, m_2, \dots, m_N\}$ for each word $t \in T$. Each morphological solution $m \in M(t)$ is a tuple of the form $\langle p, s, x, P, G, C \rangle \in \mathcal{P} \times \mathcal{S} \times \mathcal{X} \times POS \times GLOSS \times CAT$ where $p = p_1 \dots p_{|p|}$, $x = x_1 \dots x_{|x|}$, $P = P_{p_1} \dots P_{p_{|p|}} P_s P_{x_1} \dots P_{x_{|x|}}$, $G = G_{p_1} \dots G_{p_{|p|}} G_s G_{x_1} \dots G_{x_{|x|}}$, and $C = C_{p_1} \dots C_{p_{|p|}} C_s C_{x_1} \dots C_{x_{|x|}}$. P_{p_i}, G_{p_i}, and $C_{p_i}, 1 \leqslant i \leqslant |p|$ are the POS, gloss and category tags of prefix p_i. P_{x_j}, G_{x_j}, and $C_{x_j}, 1 \leqslant j \leqslant |x|$ are the POS, gloss and category tags of suffix x_i. P_s, G_s, and C_s are the POS, gloss and category tags of stem s. Intuitively, p, x, P, G and C are concatenations of prefix, suffix, POS, gloss and category values, respectively.

Table 1 shows the morphological analysis of the word فَسَيَأْكُلها. The word is composed of the prefix morphemes فَ*fa* , سَ*sa* , and يَ*ya* , followed by the stem أْكُل*ʾakul* , and then followed by the suffix morpheme ها*hā* . Each morpheme is associated with a number of morphological features. The `CONJ`, `FUT`, `IV3MS` `VERB_IMPERFECT`, and `IVSUFF_DO:3FS` POS tags indicate conjunction, future, third person masculine singular subject pronoun, an imperfect verb, and a third person feminine singular object pronoun, respectively. The POS and gloss notations follow the Buckwalter notation (Buckwalter, 2002).

3. MERF Methodology

Figure 2 illustrates the four processes involved in MERF methodology. The first process takes Arabic text and provides the user with a morphology-based Boolean (MB) formulae GUI. The user interactively composes MB-formulae using the GUI and the output of the simulator and the Syn^k detector. The simulator and the detector apply the formulae over the morphological solutions of the Arabic text and produce the MB-formulae tags.

The second process takes the MB-formulae tags and the Arabic text and provides the user with a morphology-based grammar rule GUI. The user interactively composes MB-grammar rules using the GUI and the output of the MB-grammar rule simulator. The grammar rule simulator applies the rules over the MB-formulae tags and produces the MB-grammar rule tags.

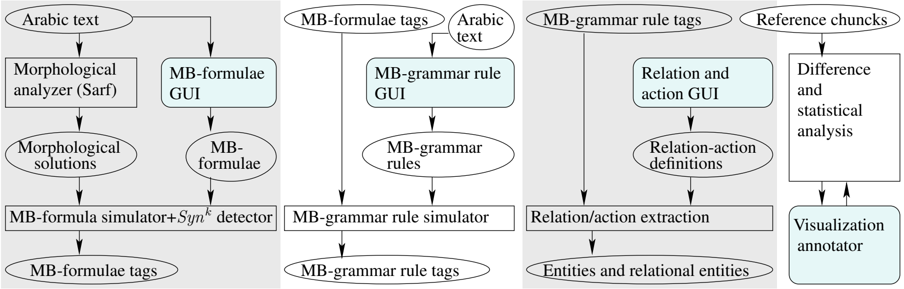


Figure 2. MERF *four process methodology with rounded corner blocks for GUI.*

The third process takes the MB-grammar rule tags and provides the user with a relation and action GUI. The user interactively provides (1) the relation definitions and (2) the actions in terms of identifiers from the MB-grammar rules. The relation extraction produces the target entities and relational entities. The action execution enriches the entities and the relational entities with powerful semantics. For example, users can utilize actions to compute statistical features, store intermediate results, or apply intelligent entity inference techniques as we show later in the numerical extraction example of Subsection 7.4. Finally, in the fourth process the user compares the results with golden reference chunks and visualizes the difference. This allows the user to refine the formulae, rules, relations and actions.

After relation extraction, we are interested to relate entities that express the same concept. MERF provides the extended synonym feature of second order as a default cross-reference relation (Syn^2). In Figure 1, triggering this feature creates the edge labeled with `isSyn` between the nodes `Khalifa Tower` and `The building`.

The user may refine the defined formulae, rules and relations and the corresponding formulae tags, rule tags, entities and relational entities either using the GUI or directly through readable output files. The files are in the javascript object notation (JSON) (Nolan and Lang, 2014) format that is intuitive to read and modify. MERF separates the user defined formulae, rules, actions and relations in a MERF tag type file and the matching tags in a tags files. The separation serves the user to apply the tag types to multiple case studies and to obtain a separate file of resulting tags for each.

4. MERF Components

4.1. *The extended synonymy feature* Syn^k

Up to our knowledge, Syn^k provides the first light Arabic WordNet based on the lexicon of Sarf. The sets E, A, and L denote all English words, Arabic words, and Arabic lexicon words, respectively. Recall that $GLOSS$ and $\mathcal{S}$ denote the set of glosses and stems in the morphological analyzer, respectively. We have $GLOSS \subset E$ and $\mathcal{S} \subset L \subset A$. Function $\alpha : \mathcal{S} \rightarrow 2^{GLOSS}$ maps Arabic stems to subsets of related

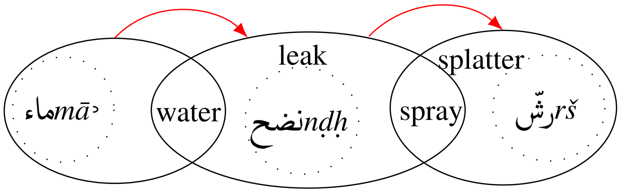


Figure 3. $Syn^2($مَاء $m\bar{a}ʾ)$.

English glosses, where 2^{GLOSS} denotes the power set of $GLOSS$ which is the set of all subsets of $GLOSS$. Function $\gamma : L \rightarrow 2^{\mathcal{S}}$ maps Arabic lexicon words to subsets of relevant Arabic stems.

Given a word $w \in L$, $Sy(w) = \{u \mid u \in \mathcal{S} \wedge \exists s \in \gamma(w) \wedge \alpha(u) \cap \alpha(s) \neq \emptyset\}$ is the set of Arabic stems directly related to w through the gloss map. Let $Sy^i(w)$ denote stems related to w using the gloss map of order i recursively such that $Sy^1(w) = Sy(w)$ and $Sy^{i+1}_k(w) = \{u \mid u \in S \wedge \exists s \in Sy^i(w) \wedge \ \alpha(u) \cap \alpha(s) \neq \emptyset\}$. Formally, $Syn^k(w) = \bigcup\limits_{i=1} Sy^i(w)$ for $i \in [1 \ldots k]$. The example in Figure 3 illustrates the computation. Let w denote an input Arabic word مَاء *māʾ* , which has the gloss `water`, i.e. `water` $\in \alpha(w)$. w shares this gloss with the stem نضح *nḍḥ* , denoted s_1, i.e. $s_1 \in Sy^1(w)$. Next, the stem رشّ *ršš* , denoted s_2, shares the gloss `spray` with s_1, i.e. $s_2 \in Sy^1(s1) \subset Sy^2(w)$. Therefore, $Syn^2(w)$ relates the words مَاء *māʾ* and رشّ *ršš* .

4.2. *MRE: Morphology-based regular expressions*

Let $\mathcal{O} = \{isA, contains\}$ be the set of atomic term predicates, where isA and $contains$ denote exact match and containment, respectively. Also, let $\mathcal{F} = \{\mathcal{P}, \mathcal{S}, \mathcal{X}, POS, GLOSS, CAT\}$ be the set of morphological features where each morphological feature $A \in \mathcal{F}$ is in turn a set of morphological feature values. Given a word w, a user defined constant feature value $CF \in A$, and an integer $k, 1 \leqslant k \leqslant 7$, the following are morphology-based atomic terms (MAT), *terms* for short.

– $a(w) := \exists m \in M(w).\ m = \langle p, s, x, P, G, C\rangle.r \circ CF$ where $\circ \in \mathcal{O}$, $r \in \{p, s, x, P, G, C\}$, and $r \in A$. Informally, a solution vector of w exists with a feature containing or exactly matching the user-chosen feature value CF.

MBF	description	formula	matches
N	name of person	$category\ =\ Name\_of\_Person$	n_1, n_2, n_3
P	name of place	$category\ =\ Name\_of\_Place$	$p_1, p_2, ..., p_7$
R	relative position	$stem \in \{$قرب,في,...$\}$	r_1, r_2, r_3, r_4
U	numerical term	$stem \in \{$أول,ثاني,...$\}$	u_1, u_2

Table 2. *Boolean formulae corresponding to task in Figure 1.*

– $a(w) := w \in Syn^k(CF), CF \in \mathcal{S}$. Informally, this checks if w is an extended synonym of a stem CF. We limit k to a maximum of 7 since we practically noticed that (1) values above 7 introduce significant semantic noise and (2) the computation is expensive without a bound.

A morphology-based Boolean formula (MBF) is of the following form.

– a and $\neg a$ are MBF formulae where a is a MAT and $\neg$ is the negation operator.

– $(f \vee g)$ is an MBF where f and g are MBF formulae, and $\vee$ is the disjunction (union) operator.

Moreover, MERF provides O to be a default Boolean formula that tags all *other* words in the text that do not match a user defined formula. We also refer to those words as *null* words.

Consider the task we discussed in the introduction (Figure 1) and recall that we are interested in identifying names of people, names of places, relative positions, and numerical terms. Table 2 presents the defined formulae. The user denotes the "name of person" entities with formula N which requires the *category* feature in the morphological solution of a word to be `Name_of_Person`. The entities n_1, n_2, and n_3 are matches of the formula N in the text. Similarly, the user specifies formula P to denote "name of place" entities. The user specifies formula R to denote "relative position" entities, and defines it as a disjunction of terms that check for solutions matching stems such as قرب*qrb* ("near") and في*fy* ("in"). Similarly, U denotes numerical terms and is a disjunction of constraints requiring the stem feature to belong to a set of stems such as أول*ʾwl* ("first"), ثاني*ṯāny* ("second"), . . . عاشر*ʿāšr* ("tenth").

Next, we define a morphology-based regular expression (MRE) as follows.

– m is an MRE where m is an MBF.

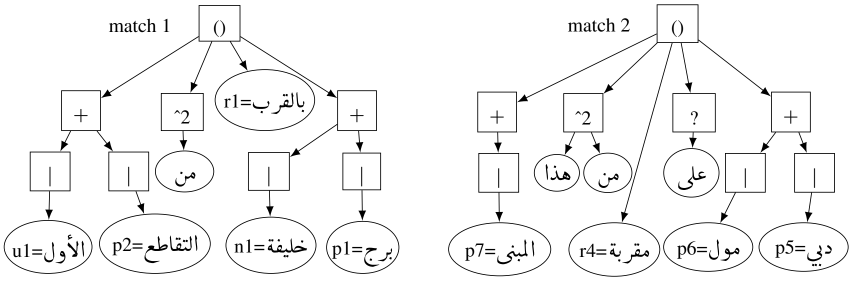


Figure 4. *Matches of regular expression* $(P|N)+\ O?\ R\ O\hat{\ }2\ (P|N|U)+$.

– fg is an MRE where f and g are both MRE expressions. A match of f followed by a match of g satisfies this concatenation operation.

– $f*$, $f+$, f^x, and $f?$ are MRE where f is an MRE, and are satisfied by zero or more, one or more, up to x matches, and an optional single match of f, respectively.

– $f\&g$, (conjunction) and $f|g$ (disjunction) are MRE where f and g are MRE, and are satisfied by the intersection of f and g matches, and the union of the f and g matches, respectively.

We denote by $[\![f]\!]$ the set of matches of an MRE f.

Back to the example in Figure 1. We use the formulae defined in Table 2 to construct an MRE such as $(P|N)+\ O?\ R\ O\hat{\ }2\ (P|N|U)+$ where $|, +, ?,$ and ^ k denote disjunction, one or more, zero or one, and up to k matches, respectively. The expression specifies a sequence of places or names of persons, optionally followed by a null word, followed by one relative position, followed by up to two possible null words, followed by one or more match of name of place, name of person, or numerical term. $O?$ and O^2 are used in the expression to allow for flexible matches.

The matching parse trees in Figure 4 illustrate two matches of the expression computed by MERF. The first tree refers to the text برج خليفة بَالقرب من التقَاطع الأول *brǧ ẖlyfh bālqrb mn āltqāṭʿ āl-ʾwl* ("Khalifa Tower next to the first intersection"). The second tree refers to the text دبي مول علَى مقربة من هذَا المبنَى *dby mwl ʿlā mqrbh mn hḏā ālmbnā* ("Dubai Mall is located near this building"). The leaf nodes of the trees are matches to formulae and the internal nodes represent roots to subexpression matches. For instance, برج خليفة *brğ ẖlyfh* in match 1 tree corresponds to the subexpression $(P|N)+$.

4.3. *User-defined relations and actions*

A relation is defined by the user as a tuple $\langle e_1, e_2, r\rangle$ where e_1, e_2, and r are identifiers associated with subexpressions of an MRE f. Matches of the relation are a set of labeled binary edges where matches of e_1 and e_2 are the source and destination nodes and matches of r are the edge labels. We denote $[\![\langle e_1, e_2, r\rangle]\!]$ to be the set of matches of the corresponding relation, and we refer to them as relational entities.

We are interested in constructing the relational entity graph in Figure 1. Let e_1, o_1, r, o_2,, and e_2 be identifiers to the subexpressions $(P|N)+$, $O?$, R, $O \wedge 2$, and $(P|N|U)+$, respectively. The matches to e_1, r, o_2, and e_2 in match 1 (Fig. 4) are برج خليفة*brğ ẖlyfh* ("Khalifa Tower"), بَالقرب*bālqrb* ("next"), من*mn* ("to"), and التقَاطع الاول*āltqāṭʿ āl-ʾwl* ("first intersection"). Note that there is no match to the optional O formula in match 1. Similarly, the matches to e_1, o_1, r, o_2, and e_2 in the second matching tree are دبي مول*dby mwl* ("Dubai Mall"), علَى*ʿlā* ("is located"), مقربة*mqrbh* ("near"), من هذَا*mn hḏā* ("this"), and المبنَى*ālmbnā* ("building"), respectively.

We define the semantic relations $\langle e_1, e_2, r\rangle$, $\langle r, e_1, o_1\rangle$, and $\langle r, e_2, o_2\rangle$. Relation $\langle e_1, e_2, r\rangle$ creates the edge labeled `next to` between `Khalifa tower` and `intersection 1` nodes from match 1, and the edge labeled `near` between `Dubai Mall` and `the building` nodes from match 2. Relation $\langle r, e_1, o_1\rangle$ creates the edge labeled `prep` between `Dubai Mall` and `near` nodes from match 2. Relation $\langle r, e_2, o_2\rangle$ creates the edge labeled `from` between `intersection 1` and `next to` nodes in match 1, and the edge labeled `from this` between `near` and `the building` nodes in match 2.

Moreover, MERF allows advanced users to write C++ code snippets to process matches of subexpressions. Each subexpression can be associated with two computational actions: `pre-match` and `on-match`. MERF provides an API that enriches the actions with detailed access to all solution features of an expression or a formula match including text, position, length, equivalent numerical value when applicable, and morphological features. The API follows a decorator pattern in that it incrementally adds the action results to the matching entities. Once MERF computes all matching parse

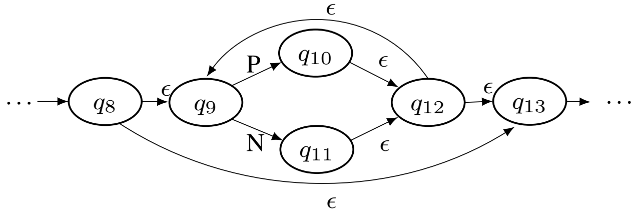


Figure 5. *Equivalent NFA of direction expression.*

trees, it traverses each tree to execute the user defined `pre-match` actions in pre-order manner and the `on-match` actions in post-order manner. This follows an observer pattern that notifies listeners with each produced match.

4.4. MERF *simulators*

The set of tag types $\mathcal{T}$ contains tuples of the form $\langle l, f, d\rangle$ where l is a text label with a descriptive name, f is an MRE, and d is a visualization legend with font and color information. For the example of Figure 1, l is “direction”, f is $(P|N)+\ O?\ R\ O\hat{}2\ (P|N|U)+$, and d is italic.

For each word $t_i \in T, 0 \leqslant i < |T|$. MERF computes a Boolean value for all MBFs. For example, برج *brğ* matches MBF P. Then, it computes the set of MBF tags $R_i = \{(t_i, tt)|tt = \langle l, f, d\rangle \wedge f\ is\ an\ MBF \wedge f(t_i)\} \subseteq T \times \mathcal{T}$ which tags a word t_i with tt iff the MBF f associated with tag type tt is true for t_i. The MBF evaluation results in a sequence of tag sets $\langle R_0, R_1, \ldots, R_{n-1}\rangle$. If a word t_i has no tag type match, its tag set R_i is by default the singleton $O = \{NONE\}$. For example, the tag sets for the text in Figure 2 follows $\{\{NONE\},\{NONE\},\{NONE\},\{NONE\},\{($ برج *brğ* $,P)\},\{($ خليفة *ḫlyfh* $,N)\},\ldots\}$.

For each MRE, MERF generates its equivalent non-deterministic finite automaton (NFA) in the typical manner (Sipser, 2012). We support the upto operation $(f\hat{}x)$, which is not directly supported in Sipser (2012), by expanding it into a regular expression form; for example $f\hat{}3$ is equivalent to $f?|ff|fff$. Consider the example of Figure 1 and the corresponding expression $(P|N)+\ O?\ R\ O\hat{}2\ (P|N|U)+$. Figure 5 shows part of the corresponding NFA where $q_8, q_9, \ldots, q_{13}$ represent NFA states, and edges are transitions based on MBF tags such as P, and N. Edges labeled with the empty string ϵ are non-deterministic.

MERF simulates the generated NFA over the sequence of tag sets matching the MBF formulae. A simulation match m of an expression f is a parse tree where the root spans the expression, the internal nodes are roots to subexpressions of f, and the leaves are matches of the MBF formulae of f, e.g. Figure 4. The sequence of leaf matches forms a vector of tags $\langle r_k, r_{k+1}, \ldots, r_j\rangle$ corresponding to the text sequence $\langle t_k, t_{k+1}, \ldots, t_j\rangle$ where $r_\ell \in R_\ell, 0 \leqslant k \leqslant \ell \leqslant j < n$. If we have more than one match for an expression, MERF returns the longest.

Finally, MERF computes the relational entities corresponding to each user defined relation $[\![\langle e_1, e_2, r\rangle]\!] \subseteq [\![e_1]\!] \times [\![e_2]\!] \times [\![r]\!]$.

5. MERF GUI

MERF provides a user friendly interface to specify the atomic terms, the MBFs, the MREs, the tag types, and the legends. The GUI also allows the user to modify and correct the tag set R. The GUI allows the user also to compute accuracy results that compare different tag sets and that can serve well as inter annotation agreement results when the tag sets come from two human annotators, or as evaluation results when comparing with reference tag sets.

5.1. *Tag type Boolean formula editor*

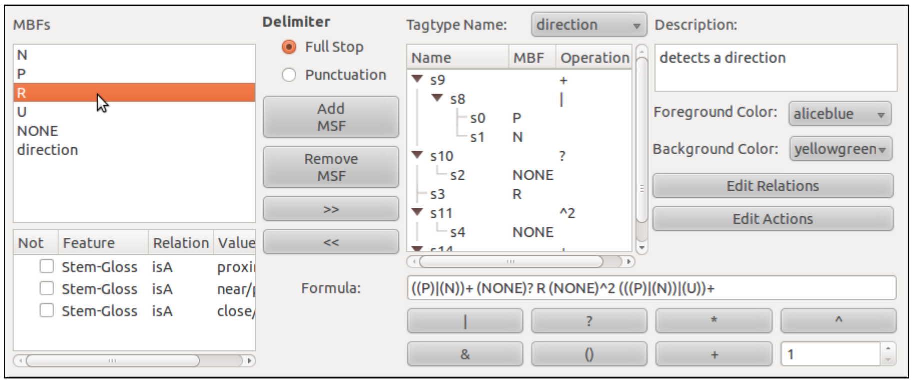
The user writes MBF tag types with the tag type editor introduced in Jaber and Zaraket (2013). First the user specifies atomic terms by selecting a feature from $\mathcal{F}$. The user can also choose whether to require an exact match using the `isA` predicate, or a substring match using the `contains` predicate option.

The user can add and remove feature values to the atomic terms using push buttons. A check box in the “Feature” column allows negating the term, and the “Relation” column switches the predicate between `isA` and `contains`. The list of feature and value pairs is interpreted as a disjunction to form the MBF. A right pane shows a description of the tag type and a set of legend descriptors. When the stem or gloss features are selected, the user has the option to use the Syn^k feature.

In the direction extraction task example, the user specifies four MBF-based tag types with labels N, P, R, and U with “name of person”, “name of place”, “relative position”, and “numerical term” descriptions, respectively. For each MBF, the user selects the morphological features, specifies the constant value CF, and adds it to the Boolean formula editor.

5.2. *MBF match visualization*

The MBF match visualizer shows color sensitive text view, the tag list view, and the tag description view. The tag description view presents the details of the selected tag along with the relevant tag type information. The user can edit the tags using a context sensitive menus. MERF GUI also allows manual tag types and corresponding tags that are not based on morphological features. This enables building reference corpora without help from the morphological analyzer.



MBFs

N
P
R
U
NONE
direction

Not	Feature	Relation	Value
☐	Stem-Gloss	isA	proxi
☐	Stem-Gloss	isA	near/
☐	Stem-Gloss	isA	close/

Delimiter
● Full Stop
○ Punctuation

Tagtype Name: direction

Name	MBF	Operation
s9		+
s8		|
s0	P	
s1	N	
s10		?
s2	NONE	
s3	R	
s11		^2
s4	NONE	

Description: detects a direction

Foreground Color: aliceblue

Background Color: yellowgreen

Formula: ((P)|(N))+ (NONE)? R (NONE)^2 (((P)|(N))|(U))+

Figure 6. MERF *tag type regular expression editor.*

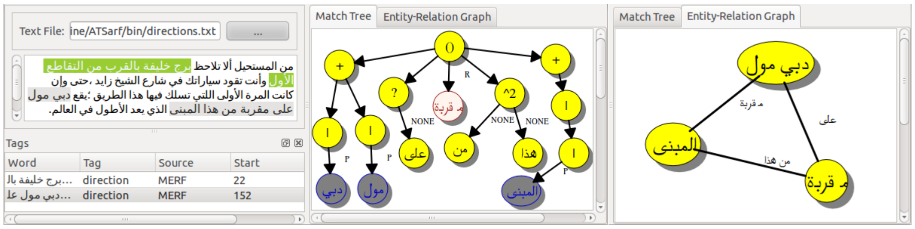
5.3. *Tag type regular expression editor*

After interacting with the MBF editor, the user moves to specify the regular expressions. The MRE editor of Figure 6 allows the definition of an MRE tag type in a user-friendly manner. The user first adds the required MBF formulae by selecting a label from $\mathcal{T}$ under MBFs. The Boolean formula of a highlighted tag type is shown in the table on the lower left pane. Each selected MBF is associated with an automatic name. The user can nest the MRE expression using a tree view of the MRE operations. The tree features the name, MBF, and operation for each subexpression.

To specify a binary operation the user selects two subexpressions and clicks the corresponding operation button. The operations include disjunction, conjunction, zero or one, sequence, zero or more, one or more, and up to a user defined constant. The right pane shows a description of the tag type and a set of legend descriptors.

5.4. *MRE match visualization*

While specifying an MRE, the user can interact with the visualization and editor views to make sure the MRE expresses the intent. The color-sensitive text view in Figure 7 shows the highlighted tag matches after the user called the MRE simulator using the `Tagtypes` menu.



Text File: ine/ATSarf/bin/directions.txt ...

من المستحيل ألا تلاحظ برج خليفة بالقرب من التقاطع
الأول وأنت تقود سيارتك في شارع الشيخ زايد ،حتى وإن
كانت المرة الأولى اللتي تسلك فيها هذا الطريق ؛يقع دبي مول
على مقربة من هذا المبنى الذي يعد الأطول في العالم.

Tags

Word	Tag	Source	Start
برج خليفة بال...	direction	MERF	22
دبي مول عل...	direction	MERF	152

Figure 7. *MRE annotated Text, MRE matching parse tree, and entity-relation graph.*

The matching parse tree view shows the selected match in a graph view. Figure 7 shows the matching parse tree of the direction task دبي مول علَى مقربة من هذَا المبنَى/*dby mwl ʿlā mqrbh mn hḏā ālmbnā* ("Dubai Mall is located near this building").

5.5. *User defined relation editor*

After the user is satisfied with the MRE matches, the user moves to define relations and code actions. The relation editor allows the user to define relations by specifying $\langle e_1, e_2, r\rangle$ tuples, where e_1 and e_2 denote source and destination entities, and r denotes the label. The editor shows the MRE tree and allows the user to select the subexpressions and select features of the matches of the subexpressions to define the three components of the relation.

A snapshot of the GUI in Figure 7 shows in an interactive graph view the entity-relation graph of the match of the user defined relation extracted from the matching parse tree of the MRE. In the computational action editor, an advanced user can enter C++ code and use the MERF API to program and process subexpression matches.

5.6. *Analysis*

In the analysis view, the user provides two tag sets R_1 and R_2 and two tag type sets $\mathcal{T}_1$ and $\mathcal{T}_2$ as input. The tag type difference view shows the text annotated in three panes: (i) the common tag types $\mathcal{T}_1 \cap \mathcal{T}_2$, (ii) the tag types in $\mathcal{T}_1$ but not in $\mathcal{T}_2$, and (iii) the tag types in $\mathcal{T}_2$ and not in $\mathcal{T}_1$. Similarly, the tag difference view shows $R_1 \cap R_2$, R_1/R_2 and R_2/R_1 in addition to precision, recall and F-measure values. The user selects a predicate to compute the metrics from the following predicates: (1) "Intersection": a tag from R_1 intersects in text with a tag in R_2, (2) "Exact": a tag from R_1 exactly matches a tag in R_2, (3) "A includes B": a tag from R_1 contains a tag from R_2, and (4) "B includes A": a tag from R_2 contains a tag from R_1.

Features	MERF	SystemT	TEXTMARKER	Urbain	QARAB
Query type	MRE	AQL	matching rules	natural language	natural language
Morphology support	✓	-	-	OpenNLP	Parser
Relations	✓	-	-	✓	-
Actions	✓	-	-	-	-
Editor	✓	-	✓	-	-
Tag visualization	✓	-	✓	-	-
Graph visualization	✓	-	-	-	-

Table 3. *Comparison of* MERF *with SystemT, TEXTMARKER, Urbain, QARAB.*

6. Related Work

In this section we review the literature on entity and relation IE and on automatic and manual annotation techniques and compare to MERF.

Information Extraction. The common pattern specification language (CPSL) targets system independent IE specifications (Appelt and Onyshkevych, 1998). MERF extends CPSL with Arabic morphological features, code actions, and user defined relations. SystemT (Chiticariu *et al.*, 2010) aims to overcome the limitations of CPSL. It is based on an algebraic approach to declarative information extraction, uses the declarative annotation query language (AQL), and uses an optimizer to generate high performance execution plans for the AQL rules. MERF supports multiple tags per word, and supports the MRE conjunction operator which overcomes the overlapping annotation problem discussed in SystemT.

TEXTMARKER is a semi-automatic rule-based IE system for structured data acquisition (Atzmueller *et al.*, 2008). Both TEXTMARKER and MERF provide the user with GUI editor and result visualizer.

The work in Urbain (2012) presents a user-driven relational model and targets entity and relation extraction. The user enters a natural language query, and uses the OpenNLP toolkit to extract tags and relations from the query. Similar to MERF, the system constructs entities and relations.

QARAB is an Arabic question answering system that takes an Arabic natural language query and provides short answers for it (Hammo *et al.*, 2002). QARAB uses traditional information retrieval techniques and an outdated Arabic NLP analyzer with limited features of Arabic words compared to the morphological analysis of MERF.

Table 3 summarizes the comparison between MERF and other systems. MERF differs in that it provides code actions, user defined relations, and an interactive graph visualization of the relational entities. It also differs in that it fully supports Arabic morphological analysis while only QARAB supports Arabic linguistic features using a parser, and the work in Urbain (2012) uses OpenNLP that currently lacks full support for Arabic morphological features. Similar to TEXTMARKER, MERF has the advantage of providing a user-friendly interactive interface to edit the entity and relational specifications and visualize the results.

DUALIST is an annotation system for building classifiers for text processing tasks using machine learning techniques (Settles, 2011). MERF doesn't support classification tasks. However, MERF provides an interactive GUI where the user can edit MBF and MRE tags. This interactive environment contributes to the regular expression extraction and semantic relation construction which increases the overall accuracy.

Another track in the literature targets specific tasks such as NER using statistical and machine-learning techniques such as maximum entropy, optimized feature sets and conditional random fields (Benajiba *et al.*, 2007; Benajiba *et al.*, 2008; Ekbal and Bandyopadhyay, 2008; AbdelRahman *et al.*, 2010). Knowledge-based techniques such as Zaghouani *et al.* (2010) and Traboulsi (2009) propose local grammars with morphological stemming. Makhlouta *et al.* (2012) extract entities and events, and relations among them, from Arabic text using a hierarchy of manually built finite state machines driven by morphological features, and graph transformation algorithms. Such techniques require advanced linguistic and programming expertise.

WordNet is a lexical reference system that mimics human lexical memory and relates words based on their semantic values and their functional categories: nouns, verbs, adjectives, adverbs, and function words (Miller *et al.*, 1990). The Syn^k feature in MERF is inspired by WordNet.

Annotation tools. MMAX2 is a manual multi-level linguistic annotation tool with an XML based data model (Müller and Strube, 2006). BRAT (Stenetorp *et al.*, 2012) and WordFreak (Morton and LaCivita, 2003) are manual multi-lingual user-friendly web-based annotators that allow the construction of entity and relation annotation corpora. Knowtator (Ogren, 2006) is a general purpose incremental text annotation tool implemented as a Protégé (Gennari *et al.*, 2003) plug-in. Protégé is an open-source platform with a suite of tools to construct domain models and knowledge-based applications with ontology. However, it doesn't support the Arabic language.

MERF differs from MMAX2, BRAT, WordFreak, and Knowtator in that it is an automatic annotator that allows manual corrections and sophisticated tag type and relation specifications over Arabic morphological features.

Kholidy and Chatterjee (2010) present an overview of annotation tools and concludes with a set of rules and guidelines needed in an Arabic annotation alignment tool. The work in Dukes *et al.* (2013) presents a collaborative effort towards morphological and syntactic annotation of the Quran. Dorr *et al.* (2010) present a framework for interlingual annotation of parallel text corpora with multi-level representations. Kulick (2010) presents the integration of the Standard Arabic Morphological Analyzer (SAMA) into the workflow of the Arabic Treebank.

The work in Smrz and Pajas (2004) presents a customizable general purpose tree editor, with the Arabic MorphoTrees annotations. The MorphoTrees present the morphological analyses in a hierarchical organization based on common features.

Task specific annotation tools such as Alrahabi *et al.* (2006) use enunciation semantic maps to automatically annotate directly reported Arabic and French speech.

AraTation is another task specific tool for semantic annotation of Arabic news using web ontology based semantic maps (Saleh and Al-Khalifa, 2009). We differ in that MERF is general, and not task specific, and it uses morphology-based features as atomic terms. Fassieh is a commercial Arabic text annotation tool that enables the production of large Arabic text corpora (Attia *et al.*, 2009). The tool supports Arabic text factorization including morphological analysis, POS tagging, full phonetic transcription, and lexical semantics analysis in an automatic mode. Fassieh is not directly accessible to the research community and requires commercial licensing. MERF is open source and differs in that it allows the user to build tag types and extract entities and relations from text.

7. Results

In this section we evaluate MERF with four case studies. We perform a survey-like evaluation where developers manually built task specific information extraction tools for the case studies and other developers built equivalent MERF tools. The aim of the comparison is to showcase that MERF enables fast development of linguistic applications with similar accuracy and a reasonable affordable overhead in computational time. We report development time, size of developed code versus size of grammar, running time, and precision-recall as metrics of cost, complexity, overhead, and accuracy, respectively.

We survey three case studies from the literature: (1) narrator chain, (2) temporal entity, and (3) genealogy entity extraction tasks, and we use the reported development time for the task specific techniques proposed in ANGE (Zaraket and Makhlouta, 2012a), ATEEMA (Zaraket and Makhlouta, 2012c), and GENTREE (Makhlouta *et al.*, 2012), respectively. We also compare a MERF number normalization task to a task specific implementation.

We evaluated ANGE with *Musnad Ahmad*, a hadith book, where we constructed an annotated golden reference containing 1,865 words. We evaluated ATEEMA with articles from issues of the Lebanese *Al-Akhbar* newspaper where we constructed an annotated golden reference containing 1,677 words. For the genealogical tree extraction we used an extract from the Genesis biblical text with 1,227 words. Finally, we used an annotated article from the Lebanese *Assafir* newspaper with 1,399 words to evaluate the NUMNORM case study[2]. In the online appendix[3], we report on eight additional MERF case studies. Manual annotators inspected the outcome and provided corrections where tools made mistakes. The corrections form the manual gold annotation that we compared against.

Table 4 reports the development time, extraction runtime, recall and precision of the output MRE tags, the size of the task in lines of code or in number of MERF rules, for both the standalone task specific and the MERF implementations. The develop-

[2] Available at `http://www.assafir.com` and `http://www.al-akhbar.com`.

[3] Available at `http://research-fadi.aub.edu.lb/pdfs/merfappendix.pdf`.

Task	Size (words)	Development time	Run time(s)	Accuracy		Ease of Composition
				Recall	Precision	
ANGE	1,865	2 months	1.79	0.99	0.99	3,000+ lines of code
MERF		3 hours	7.24	0.99	0.93	8 MBFs and 4 MREs
ATEEMA	1,677	1.5 months	2.53	0.88	0.89	1,000+ lines of code
MERF		3 hours	3.14	0.91	0.81	3 MBFs and 2 MREs
Genealogy tree	1,227	3 weeks	0.74	0.96	0.98	3,000+ lines of code
MERF		4 hours	2.28	0.84	0.93	3 MBFs and 3 MREs
NUMNORM	1,399	1 week	0.32	0.91	0.93	500 lines of code
MERF		1 hour	1.53	0.91	0.90	3 MBFs/1 MRE/57 lines

Table 4. MERF *compared to task specific applications.*

ment time measures the time required for developing the case study. For instance, ANGE (Zaraket and Makhlouta, 2012a) required two months of development by a research assistant with 6 and 14 hours of course work and teaching duties, respectively. Recall refers to the fraction of the entities correctly detected against the total number of entities. Precision refers to the fraction of correctly detected entities against the total number of extracted entities.

Table 4 provides runtime results of MERF compared to the task specific implementations while running MBF and MRE simulations jointly. This is a rough estimate of the complexity of the MERF simulator. The complexity of the MBF simulation is the total number of morphological solutions for all the words multiplied by the number of user-defined MBFs. We do not provide a limit on the number of user defined formulae. In practice, we did not encounter more than ten formulae per case study. As for the complexity of MRE simulation, converting the rules into non-deterministic finite state machines (NDFSM) is done once. Simulating an NDFSM over the MBF tags is potentially exponential. In practice, all our case studies terminated within a predetermined time bound of less than 30 minutes. MERF required reasonably more runtime than the task specific implementations and reported acceptable and slightly less precision metrics with around the same recall.

Table 4 shows that MERF has a clear advantage over task specific techniques in the effort required to develop the application at a reasonable cost in terms of accuracy and run time. Developers needed three hours, three hours, four hours, and one hour to develop the narrator chain, temporal entity, genealogy, and number normalization case studies using MERF, respectively. However, the developers of ANGE, ATEEMA, GENTREE, and NUMNORM needed two months, one and a half months, three weeks, and one week, respectively. MERF needed eight MBFs and four MREs for narrator chain, three MBFs and two MREs for temporal entity, three MBFs and three MREs for genealogy, and three MBFs, one MRE, and 57 lines of code actions for the number normalization tasks. However, ANGE, ATEEMA, GENTREE, and NUMNORM required 3,000+, 1,000+, 3,000+, and 500 lines of code, respectively.

```
name:   PN ((MEAN)? PN)*;
nar:    name ((NONE)^3 FAM (NONE)^3 name)*;
pbuh:   BLESS GOD UPONHIM GREET;
nchain: (s1 =TOLD s2 =nar)+ ((PN|FAM|NONE)^8 pbuh)?
```

القعقاع	بن	عمارة	عن	جرير	حدثنا	سعيد	بن	قتيبة	حدثنا
ālqʿqāʿ	*bn*	*ʿmārh*	*ʿn*	*ǧryr*	*ḥdṯnā*	*sʿyd*	*bn*	*qtybh*	*ḥdṯnā*
PN	FAM	PN	TOLD	PN	TOLD	PN	FAM	PN	TOLD
name		name		name		name		name	
nar				nar		nar			
nchain									

Table 5. *Narrator chain example.*

7.1. *Narrator chain case study*

A narrator chain is a sequence of narrators referencing each other. The chain includes proper nouns, paternal entities, and referencing entities. ANGE uses Arabic morphological analysis, finite state machines, and graph transformations to extract entities and relations including narrator chains (Zaraket and Makhlouta, 2012a).

Table 5 presents the MREs for the narrator chain case study. MBF `PN` checks the abstract category `Name of Person`. MBF `FAM` denotes "family connector" and checks the stem gloss "son". MBF `TOLD` denotes referencing between narrators and checks the disjunction of the stems حدث("spoke to"), عن("about"), سمع("heard"), أخبر("told"), and أنبأ("inform"). MBF `MEAN` checks the stem عني("mean"). MBFs `BLESS`, `GOD`, `UPONHIM`, and `GREET` check the stems صلَّى, الله, علي, and سلّم, respectively.

MRE *name* is one or more `PN` tags optionally followed with a `MEAN` tag. MRE `nar` denotes narrator which is a complex Arabic name composed as a sequence of Arabic names (`name`) connected with family indicators (`FAM`). The `NONE` tags in `nar` allow for unexpected words that can occur between names. MRE `pbuh` denotes a praise phrase often associated with the end of a hadith ("peace be upon him"), and is the satisfied by the sequence of `BLESS`, `GOD`, `UPONHIM`, and `GREET` tags. MRE `nchain` denotes narrator chain, and is a sequence of narrators (`nar`) separated with `TOLD` tags, and optionally followed by a `pbuh` tag.

Task	MBF accuracy		relation accuracy	
	Recall	Precision	Recall	Precision
Narrator chain	0.99	0.85	0.99	0.98
Number normalization	0.99	0.99	0.97	0.95
Temporal entity	0.99	0.52	0.98	0.89
Genealogy tree	0.99	0.75	0.81	0.96

Table 6. MERF *MBF and user-defined relation accuracy.*

The first row in Table 5 is an example narrator chain, the second is the transliteration, the third shows the MBF tags. Rows 4, 5, and 6 show the matches for `name`, `nar`, and `nchain`, respectively. MERF assigns the symbols s_1 and s_2 for the MRE subexpressions `TOLD` and `nar`, respectively. We define the relation $\langle s_2, s'_2, s_1 \rangle$ to relate sequences of narrators with edges labeled by the tags of `TOLD` where s'_2 denotes the next match of `nar` in the one or more MRE subexpression. Table 6 shows that MERF detected almost all the MBF matches with 99% recall and 85% precision and extracted user-defined relations with 98% recall and 99% precision.

7.2. *Temporal entity extraction*

Temporal entities are text chunks that express temporal information. Some represent absolute time such as الخَامس من آب ٢٠١٠ *ālẖāms mn ʾāb 2010.* Others represent relative time such as بعد خمسة أَيام *bʿd ẖmsh ʾayām*, and quantities such as ١٤ يومًا *14 ywmā*. *ATEEMA* presents a temporal entity detection technique for the Arabic language using morphological analysis and finite state transducers (Zaraket and Makhlouta, 2012c). Table 6 shows that MERF detected almost all the MBF matches with 99% recall, however it shows low precision (52%). As for the semantic relation construction, MERF presents a 98% recall and 89% precision.

7.3. *Genealogy tree*

Biblical genealogical lists trace key biblical figures such as Israelite kings and prophets with family relations. The family relations include wife and parenthood. A sample genealogical chunk of text is و ولد هَارَان لوطًا *w wld hārān lwṭā* meaning "and Haran became the father of Lot". GENTREE (Makhlouta *et al.*, 2012) automatically extracts the genealogical family trees using morphology, finite state machines, and graph transformations. Table 6 shows that MERF detected MBF matches with 99% recall, and 75% precision, and extracted relations with 81% recall and 96% precision.

TMB algorithm

```
cout << $s1.text;
if(isHundred) {
  if(current != 0) {
    previous += current;
  }
  current = currentH * $s1.number;
  currentH = 0;
  isHundred = false;
  isKey = true;
} else if(current == 0) {
  current = $s1.number;
  isKey = true;
} else if(!isKey) {
  isKey = true;
  current = current * $s1.number;
} else {
  previous += current;
  current = $s1.number;}
```

DT algorithm

```
if(isHundred) {currentH += $s0.number;
} else if(current == 0) {
  current = $s0.number;
} else if(isKey) {
  previous += current;
  current = $s0.number;
} else {current += $s0.number; }
isKey = false;
```

H algorithm

```
isHundred = true;
if(current == 0)  {
  currentH = $s2.number;
} else if(!isKey) {
  currentH = current * $s2.number;
  current = 0;
} else {currentH = $s2.number;}
isKey = false;
```

Figure 8. *Actions for TMB, DT, and H MRE expressions.*

7.4. *Number normalization*

We implemented a number normalization extractor using MERF and compared it with *NUMNORM*, a C++ implementation for number normalization. First, we defined the MBFs `DT`, `H`, and `TMB` to denote (1) digits and tens, (2) hundreds, and (3) thousands, millions, and billions, respectively. The `num` MRE `(DT|TMB|H)+` is one or more `DT`, `TMB`, or `H` tags. MERF assigns the symbols s_1, s_2, and s_3 for the subexpressions `DT`, `TMB`, and `H`, respectively. Figure 8 shows the actions associated with the `DT`, `TMB`, and `H` subexpressions that cumulatively compute the numeric value of the numeric expression match. The actions use MERF API to access features of the matches such as text (`$s1.text`) and numeric value (`$s1.number`) of literal numbers such as numbers from one to ten. Table 6 shows high accuracy in MBF tagging and relation extraction with 99% and 97% recall and 99% and 95% precision, respectively.

7.5. *Discussion*

The results show that MERF provides a friendly environment to develop entity and relational entity extraction tasks with acceptable accuracy and runtime overheads compared to task specific applications. MERF requires the user to understand and interact with basic linguistic concepts such as readable values of morphological features, sequences, repetitions, and bounded repetitions. The user interacts with the MBF editor to specify basic concepts and visualize their matches over highlighted text. Then, the user interacts with the MRE editor to specify sequences of the concepts and visualize the matches in a graph, in conjunction with the highlighted text.

The two levels of interaction allow the user to separate between concepts that relate to word features, and more sophisticated entities that relate to sequences and context. The MBF, MRE, and user defined relations can be used to generate large annotated corpora in a fast manner. MERF visualization can be used later to refine the annotation. The case studies showed that MERF requires some linguistic expertise to successfully execute the tasks. In contrast, the case specific implementations require more sophisticated linguistic and programming expertise to attain similar results.

We notice that ANGE, ATEEMA, and Genealogy tree report higher precision than MERF. This is mainly due to their capacity to learn words and relations that may not have a match in the morphological analyzer based on co-occurrence relations. For example, the sequence $p_1t_1p_2$ where p_1 and p_2 are persons and t_1 is a tell relationship helps indicate that x is a tell relationship in p_1xp_2 even if the morphological analyzer did not return the required feature for x to match a tell relationship. MERF does not have that capacity yet unless it is encoded in the C++ actions.

8. Conclusion

In this work, we present a morphology-based entity and relational entity extraction framework for Arabic text. MERF provides a friendly interface where the user defines tag types and associates them with regular expressions defined over Boolean formulae. The Boolean formulae are in turn defined over matches of Arabic morphological features and a novel extended synonymy feature (Syn^k). MERF allows the user to associate code actions with each regular subexpression and to define semantic relations between subexpressions. We evaluate MERF with several case studies and compare with existing application-specific techniques. The results show that MERF requires shorter development time and effort compared to existing techniques and produces reasonably accurate results within a reasonable overhead in run time. In the future, MERF will support user-defined cross-reference predicates, and will infer morphological features from relevant example words to express a concept.

9. Acknowledgment

The authors would like to thank the Lebanese National Council for Scientific Research (CNRS) for their support.

10. References

AbdelRahman S., Elarnaoty M., Magdy M., Fahmy A., "Integrated Machine Learning Techniques for Arabic Named Entity Recognition", *International Journal of Computer Science Issues*, vol. 7, n$^{\text{o}}$ 4, p. 27-36, 2010.

Al-Sughaiyer I., Al-Kharashi I., "Arabic morphological analysis techniques: A comprehensive survey", *JASIST*, 2003.

Alrahabi M., Ibrahim A. H., Desclés J.-P., "Semantic Annotation of Reported Information in Arabic", *FLAIRS Conference*, vol. 6, p. 263-268, 2006.

Appelt D., Onyshkevych B., "The common pattern specification language", *TIPSTER workshop*, ACL, 1998.

Attia M., Rashwan M., Al-Badrashiny M., “Fassieh, a semi-automatic visual interactive tool for morphological, PoS-Tags, phonetic, and semantic annotation of Arabic text corpora”, *IEEE transactions on audio, speech, and language processing*, vol. 17, n° 5, p. 916-925, 2009.

Atzmueller M., Kluegl P., Puppe F., “Rule-Based Information Extraction for Structured Data Acquisition using TextMarker”, *Proceedings of LWA*, Citeseer, p. 1-7, 2008.

Benajiba Y., Diab M., Rosso P., “Arabic named entity recognition using optimized feature sets”, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, p. 284-293, 2008.

Benajiba Y., Rosso P., Benedíruiz J. M., “Anersys: An Arabic named entity recognition system based on maximum entropy”, *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, p. 143-153, 2007.

Buckwalter T., Buckwalter Arabic Morphological Analyzer Version 1.0, Technical report, University of Pennsylvania, 2002.

Chiticariu L., Krishnamurthy R., Li Y., Raghavan S., Reiss F. R., Vaithyanathan S., “SystemT: an algebraic approach to declarative information extraction”, *Proceedings of the Association for Computational Linguistics*, p. 128-137, 2010.

Dorr B. J., Passonneau R. J., Farwell D., Green R., Habash N., Helmreich S., Hovy E., Levin L., Miller K. J., Mitamura T. *et al.*, “Interlingual annotation of parallel text corpora: a new framework for annotation and evaluation”, *NLE*, vol. 16, n° 3, p. 197-243, 2010.

Dukes K., Atwell E., Habash N., “Supervised collaboration for syntactic annotation of Quranic Arabic”, *Language resources and evaluation*, vol. 47, n° 1, p. 33-62, 2013.

Ekbal A., Bandyopadhyay S., “Named Entity Recognition using Support Vector Machine: A Language Independent Approach”, *IJCSSE*, 2008.

Ferilli S., “Natural Language Processing”, *Automatic Digital Document Processing and Management*, Springer, 2011.

Gennari J., Musen M., Fergerson R., Grosso W., Crubézy M., Eriksson H., Noy N., Tu S., “The evolution of Protégé: an environment for knowledge-based systems development”, *International Journal of Human-Computer Studies*, vol. 58, n° 1, p. 89-123, 2003.

Habash N., “Introduction to Arabic natural language processing”, *Synthesis Lectures on Human Language Technologies*, 2010.

Habash N., Sadat F., “Arabic Preprocessing Schemes for Statistical Machine Translation”, *NAACL*, p. 49-52, 2006.

Hammo B., Abu-Salem H., Lytinen S., “QARAB: A question answering system to support the Arabic language”, *Computational approaches to semitic languages*, ACL, p. 1-11, 2002.

Jaber A., Zaraket F., “MATAr: Morphology-based Tagger for Arabic”, *AICCSA*, May, 2013.

Kholidy H., Chatterjee N., “Towards developing an Arabic word alignment annotation tool with some Arabic alignment guidelines”, *ISDA*, IEEE, p. 778-783, 2010.

Kulick S., “Consistent and flexible integration of morphological annotation in the Arabic Treebank”, *LREC*, 2010.

Linckels S., Meinel C., “Natural Language Processing”, *E-Librarian Service*, Springer, p. 61-79, 2011.

Maamouri M., Bies A., Buckwalter T., Mekki W., “The penn Arabic treebank: Building a large-scale annotated Arabic corpus”, *NEMLAR Conference on Arabic Language Resources and Tools*, p. 102-109, 2004.

Makhlouta J., Zaraket F., Harkous H., “Arabic entity graph extraction using morphology, finite state machines, and graph transformations”, *CICLing*, Springer, p. 297-310, 2012.

Marcus M. P., Marcinkiewicz M. A., Santorini B., “Building a large annotated corpus of English: The Penn Treebank”, *Computational linguistics*, vol. 19, n$^{\text{o}}$ 2, p. 313-330, 1993.

Miller G. A., Beckwith R., Fellbaum C., Gross D., Miller K. J., “Introduction to WordNet: An on-line lexical database”, *International journal of lexicography*, vol. 3, n$^{\text{o}}$ 4, p. 235-244, 1990.

Morton T., LaCivita J., “WordFreak: an open tool for linguistic annotation”, *HLT/NAACL*, 2003.

Müller C., Strube M., “Multi-level annotation of linguistic data with MMAX2”, *Corpus technology and language pedagogy: New resources, new tools, new methods*, 2006.

Nolan D., Lang D., “JavaScript Object Notation”, *XML and Web Technologies for Data Sciences with R*, Springer, 2014.

Ogren P., “Knowtator: a protégé plug-in for annotated corpus construction”, *NAACL-Demonstrations*, ACL, 2006.

Pasha A., Al-Badrashiny M., Diab M. T., El Kholy A., Eskander R., Habash N., Pooleery M., Rambow O., Roth R., “MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic”, *LREC*, vol. 14, p. 1094-1101, 2014.

Saleh L., Al-Khalifa H., “AraTation: an Arabic semantic annotation tool”, *IIWAS*, 2009.

Settles B., “Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances”, *Proceedings of EMNLP*, ACL, p. 1467-1478, 2011.

Shahrour A., Khalifa S., Taji D., Habash N., “Camelparser: A system for Arabic syntactic analysis and morphological disambiguation”, *COLING Demonstrations*, p. 228-232, 2016.

Sipser M., *Introduction to the Theory of Computation*, Cengage Learning, 2012.

Smrz O., Pajas P., “Morphotrees of Arabic and their annotation in the TrEd environment”, *NEMLAR International Conference on Arabic Language Resources and Tools*, 2004.

Soudi A., Neumann G., Van den Bosch A., *Arabic computational morphology: knowledge-based and empirical methods*, Springer, 2007.

Stenetorp P., Pyysalo S., Topić G., Ohta T., Ananiadou S., Tsujii J., “BRAT: a web-based tool for NLP-assisted text annotation”, *EACL Demonstrations*, ACL, p. 102-107, 2012.

Traboulsi H., “Arabic named entity extraction: A local grammar-based approach”, *IMCSIT*, IEEE, 2009.

Urbain J., “User-driven relational models for entity-relation search and extraction”, *Proceedings of JIWES*, ACM, 2012.

Xue N., Xia F., Chiou F.-D., Palmer M., “The Penn Chinese TreeBank: Phrase structure annotation of a large corpus”, *Natural language engineering*, vol. 11, n$^{\text{o}}$ 2, p. 207-238, 2005.

Zaghouani W., Pouliquen B., Ebrahim M., Steinberger R., “Adapting a resource-light highly multilingual Named Entity Recognition system to Arabic”, *LREC*, p. 563-567, 2010.

Zaraket F. A., Makhlouta J., “Arabic Cross-Document NLP for the Hadith and Biography Literature”, *FLAIRS*, May, 2012a.

Zaraket F., Makhlouta J., “Arabic Morphological Analyzer with Agglutinative Affix Morphemes and Fusional Concatenation Rules”, *COLING*, Mumbai, India, December, 2012b.

Zaraket F., Makhlouta J., “Arabic Temporal Entity Extraction using Morphological Analysis”, *IJCLA*, vol. 3, p. 121-136, 2012c.