Language Resources and Evaluation manuscript No.
(will be inserted by the editor)

MERF: Morphology-based Entity and Relational Entity Extraction Framework for Arabic

Ameen A. Jaber · **Fadi A. Zaraket**

the date of receipt and acceptance should be inserted later

Abstract Rule-based techniques and tools to extract *entities* and *relational entities* from documents allow users to specify desired entities using natural language questions, finite state automata, regular expressions, structured query language statements, or proprietary scripts. These techniques and tools require expertise in linguistics and programming and lack support of Arabic *morphological analysis* which is key to process Arabic text. In this work, we present MERF; a morphology-based entity and relational entity extraction framework for Arabic text. MERF provides a user-friendly interface where the user, with basic knowledge of linguistic features and regular expressions, defines *tag types* and interactively associates them with regular expressions defined over Boolean formulae. Boolean formulae range over matches of Arabic morphological features, and synonymity features. Users define user defined relations with tuples of subexpression matches and can associate code actions with subexpressions. MERF computes feature matches, regular expression matches, and constructs entities and relational entities from user defined relations. We evaluated our work with several case studies and compared 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.

Keywords Arabic · information visualization · information extraction and retrieval · morphology analysis · natural language processing · tagging.

Ameen A. Jaber
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
Tel.: +961-70-962096
E-mail: aaj15@aub.edu.lb

Fadi A. Zaraket
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
Tel.: +961-01-350000 ext. 3484
Fax: +961(1)744-462
E-mail: fz11@aub.edu.lb

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* (Ferilli, 2011; Linckels and Meinel, 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 (Maamouri et al., 2004; Marcus et al., 1993; 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 (Atzmueller et al., 2008; Chiticariu et al., 2010; Müller and Strube, 2006; Settles, 2011; Stenetorp et al., 2012; Urbain, 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. Fassieh lacks support for entity and relational entity extraction.

In this paper, we present a *morphology-based entity and relational entity extraction framework for Arabic text* (MERF). MERF provides a user-friendly interface where the user defines tag types and associates them with MERF formulae that are regular expressions over MERF Boolean formulae. Boolean formulae are terms, negations of terms, and disjunctions 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. We discuss the importance of the morphological features supported in MERF terms in Section 3; in breif, morphological preprocessing is key to Arabic NLP.

We illustrate the target of MERF using a simple example. Given the text in Figure 1 that contains directions to Dubai Mall [1]. 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. MERF allows a regular user to do that interactively in four phases that

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

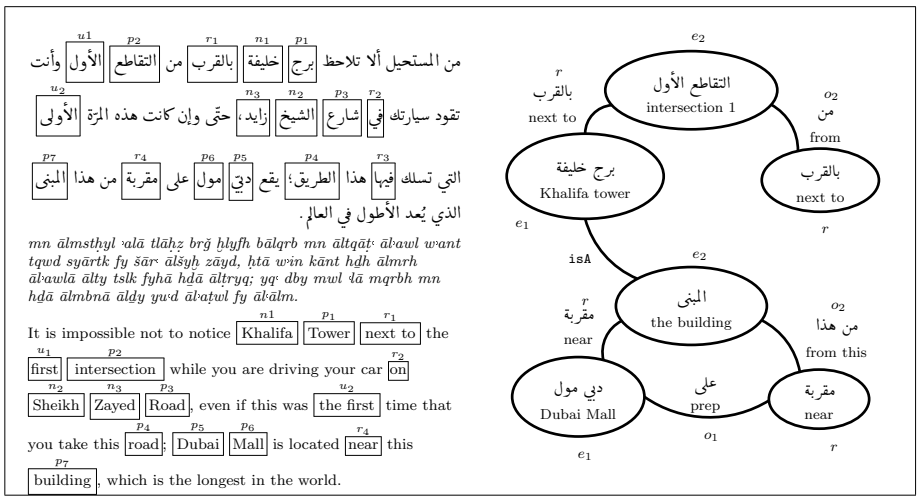

Fig. 1 Direction example with Arabic text, annotated with entities, transliteration, translation, and extracted relational entities in a graph

include morphological analysis, entity extraction based on morphological features, relational entity extraction, and entity cross-reference.

MERF regular expressions support operators such as concatenation, zero or one, zero or more, one or more, up to M, and logical conjunction and disjunction. MERF editor allows the user to associate an action with each sub-expression. The user specifies the action with C++ code and uses the MERF API to access information related to the matches such as text, position, length, morphological features, and numerical values.

MERF takes an Arabic text, a set of user-defined MERF Boolean formulae, and a set of user-defined MERF regular expressions. MERF computes the morphological solutions of the words in the input text then computes matches to the Boolean formulae. 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.

Existing annotation and entity extraction tools

Researchers proposed and evaluated empirical and knowledge-based techniques to extract entities and relational entities from text. We briefly review them here and we discuss them and compare to them in Section 6. The work in (Ekbal and Bandyopadhyay, 2008) presents a language independent approach for NER extraction using *support vector machines*. The work in (AbdelRahman et al., 2010)

integrates a semi-supervised bootstrapping pattern recognition technique, and a supervised classifier based on *conditional random fields* to solve NER problems.

Knowledge-based techniques such as (Traboulsi, 2009; Zaghouani et al., 2010) propose local grammars with morphological stemming to perform NER. (Makhlouta and et al., 2012) presents a method for extracting entities, events, and relations amongst 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. QARAB is a question answering system that takes Arabic natural language queries and provides short answers (Hammo et al., 2002).

Researchers also proposed systems for automatic IE based on user specifications. CPSL is a common pattern specification language for finite-state grammar (Appelt and Onyshkevych, 1998). The work in (Chiticariu et al., 2010) presents SystemT, a system based on an algebraic Approach to Declarative information extraction (IE). TEXTMARKER is a rule-based IE system designed to extract structured data from text (Atzmueller et al., 2008). The work in (Urbain, 2012) presents a user-driven relational model requiring a user natural language query to extract entities and relational entities.

MERF enables a regular user to incrementally create complex annotations for Arabic text 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 Boolean formulae over morphological features, build regular expressions over the resulting Boolean 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. MERF 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 synonymic relation using the Syn^k feature.

This paper significantly extends the position paper (Jaber and Zaraket, 2013) that allows for manual, and basic morphology annotation.

In the rest of this paper, Section 2 presents a motivational example. Section 3 introduces Arabic morphological analysis and its important role in Arabic NLP. Section 4 explains the MERF methodology. Section 5 presents the visual and user friendly interface of MERF. Section 7 presents the evaluation results. Section 6 presents and discusses related work. Finally, we conclude and discuss future work in Section 9.

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	$stem \in \{\text{قرب,في}, \dots\}$	r_1, r_2, r_3, r_4
U	numerical term	$stem \in \{\text{أول,ثاني}, \dots\}$	u_1, u_2

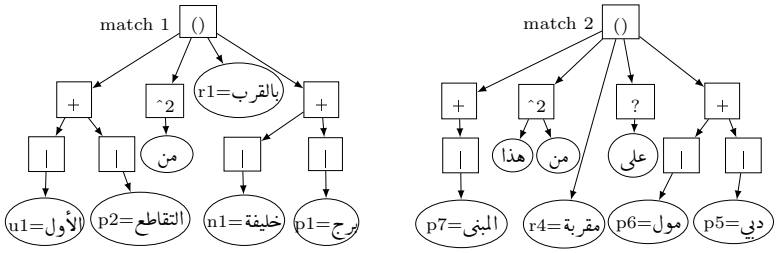

Fig. 2 Boolean formulae, and matches of regular expression $(P|N) +\ O?\ R\ O\hat{}2\ (P|N|U)+$

2 Motivation

We motivate MERF with the example of Figure 1 that shows how to extract direction entities from sample Arabic text. The text includes directions to a shopping mall and the figure presents a transliteration and an English translation of the text. Interesting entities, such as names, places, relative directions, and numerical terms are highlighted with boxes in the text. We are interested in detecting those entities and presenting the direction as relational entities as shown in Figure 1.

Entity detection. As described in the table of Figure 2, 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. Similarly, the user specifies formula P to denote "name of place" entities. The user specifies formula R to denote "relative position" entities, and requires the stem feature to belong to a selected list of stems containing في*fy* and قرب*qrb* . 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*wl* (first), ثَاني*ṯāny* (second), ... عَاشر*āšr* (tenth).

MERF calls an inhouse open source morphological analyzer (Zaraket and Makhlouta, 2012b) and computes matches of the formulae N, P, R, and U. We refer to all *other* words in the text that do not match a user defined formula as *null* words and we denote them by O. The resulting matches are illustrated with boxes and superscripts in Figure 1 and are listed in the table of Figure 2.

The user now interacts with the regular expression editor to specify the direction entities and relations. Intuitively, the directions are names of places (P) related to each other with positional propositions (R). A place name can be a tabulated place name, a street named after a person (N), or a numbered street (U). The text containing the directions might also include words that are not necessary to indicate directions (O) but are necessary to complete the sentence.

The user tries several sequences of the above entities in the editor and checks their matches in the visualizer. Finally, the user is satisfied with the matches of an expression such as $(P|N) + \ O?\ R\ O \wedge 2\ (P|N|U)+$ where $|, +, ?$, and $\wedge k$ 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 $\wedge 2$ in the expression are used to allow for flexible matches. The user reaches the satisfying matches by experimenting with the visualizer and the expression editor which do not require knowledge and expertise in regular expressions.

The match trees in Figure 2 illustrate two matches of the expression computed by MERF. The first match tree refers to the text برج خليفة بَالقرب من التقَاطع الأول*brğ ẖlyfh bālqrb mn āltqāṭʿ āl-ʾwl* (Khalifa Tower next to the first intersection). The second match 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 nodes of the trees are entities and the edges and internal nodes are text, morphology-based, and word distance based relational entities.

User defined direction relations. The user now uses the relation editor to declare user defined relations that relate parts of the matches of the expression with each other. Intuitively, the user wants to create relations between places, names, and numerical entities. A relation between two entities can be a prepositional entity. For example, Figure 1 shows the entities دبي مول*dby mwl* (Dubai Mall) and المبنَى*ālmbnā* (the building) related by the preposition مقربة*mqrbh* (near).

Let e_1, o_1, r, o_2, and e_2 be the $(P|N)+$,$O?$,R,$O \wedge 2$, and $(P|N|U)+$ subexpressions, respectively. The user selected $(P|N)+$ to be an entity after noticing in the visualizer that it happens to capture non-separated sequences of place and name entities denoting a single entity such as `Khalifa tower`. `Relation(`e_1`,`e_2`,`r`)` creates the edge labeled `next to` between `intersection 1` and `Khalifa tower` nodes in match 1 of Figure 2(b), and the edge labeled `near` between `Dubai Mall` and `the building` nodes in match 2 of Figure 2(b).

`Relation(`r`,`e_1`,`o_1`)` creates the edge labeled `prep` between `Dubai Mall` and `near` nodes in match 2 of Figure 2(b). `Relation(`r`,`e_2`,`o_2`)` creates the edge labeled `from` between `intersection 1` and `next to` nodes in match 1 of Figure 2(b), and the edge labeled `from this` between `near` and `the building` nodes in match 2 of Figure 2(b).

After constructing the user defined relations, the user is interested to relate the discovered entities and relational entities that express the same concept; in particular, the same place. MERF provides the `isA` predicate as a default cross-

Table 1 Sample solution vector for فَسَيَأْكُلُها *fasayaʾakulhā* .

	Prefixes			Stem	Suffix
Data	فَ*fa*	سَ*sa*	يَ*ya*	أْكُلُ*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

reference relation which creates the edge between the nodes `Khalifa Tower` and `The building` in Figure 1.

3 Background: Morphological Analyzer

Morphological analysis is key to Arabic NLP (Al-Sughaiyer and Al-Kharashi, 2003) due to the exceptional degree of ambiguity in writing, the rich morphology, and the complex word derivation system. Short vowels, also known as diacritics, are typically omitted in Arabic text and inferred by readers (Habash and Sadat, 2006). For example, the word بنbn can be interpreted as بُنbon ''coffee'' with a *damma* diacritic on the letter بـb or بِنbin (son of) with a *kasra* diacritic on the letter بـb .

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 ياسمينyāsmyn can be interpreted as the "Jasmine" flower, as well as يا (the calling word) followed by the word سمين (obese). Consider the sentence ذهب الوَلدالَى المدرسة *dẖb alwald-ilā 'lmdrsh* (the kid went to school). The letters د and ى have non-connecting end of word forms and the words الى,الولد, 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

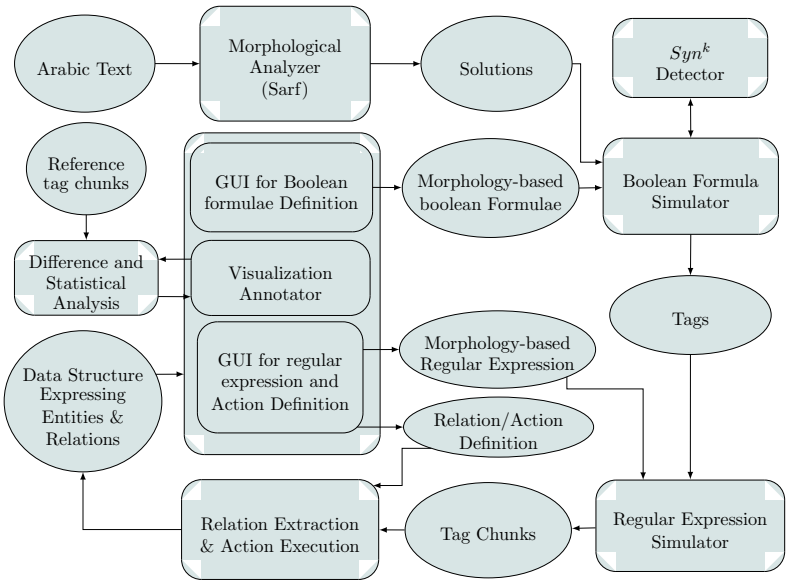

Fig. 3 MERF flow diagram.

includes infix morphological changes. The part-of-speech tag, referred to as POS, assigns a morpho-syntactic 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.

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 $T = \langle t_1, t_2, \ldots, t_M \rangle$ be a set of Arabic words denoting the text documents. MERF uses Sarf to compute a set of morphological solutions $M(t) = \{m_1, m_2, \ldots, m_N\}$ for each word $t \in T$. Each morphological solution $m \in M(t)$ is a tuple of the form $\langle p, s, x, P, G, C \rangle \in \mathcal{P} \times \mathcal{S} \times \mathcal{X} \times POS \times GLOSS \times CAT$.

Table 1 shows the morphological analysis of the word فَسَيَأْكُلُها. The word is composed of the prefix morphemes فَ*fa* , سَ*sa* , and يَ*ya* , followed by the stem أْكُلُ*akul* , 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 and a third person feminine singular object pronoun, respectively. The POS and gloss notations follow the Buckwalter notation (Buckwalter, 2002).

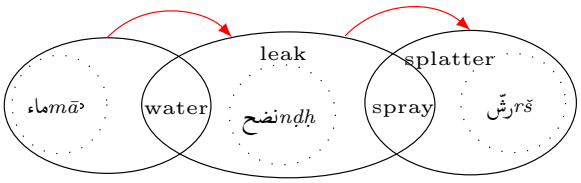

Fig. 4 Syn^2(مَاء $mā'$)

4 MERF

The MERF framework is illustrated in the flow diagram of Figure 3. The Arabic text and the reference tag chunks are the primary inputs to MERF. Solutions, morphology-based Boolean formulae, tags, morphology-based regular expressions, tag chunks, relation and action definitions, and data structures expressing entities and relations are input and output data to processes. The morphological analyzer (Sarf), Syn^k detector, GUI for Boolean formulae definition, visualization annotator, GUI for regular expression and action definition, Boolean formula simulator, regular expression simulator, relation extraction and action execution, and difference and statistical analyzer are processes.

The extended synonymy feature Syn^k

The sets E, A, and L denote all English words, Arabic words, and Arabic lexicon words, 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 English glosses. Function $\gamma : L \rightarrow 2^{\mathcal{S}}$ maps Arabic lexicon words to subsets of relevant Arabic stems.

Given a word $w \in L$, $Sy(w) = \{u \mid u \in \mathcal{S} \wedge \exists s \in \gamma(w) \wedge \ \alpha(u) \cap \alpha(s) \neq \varnothing\}$ 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 $Sy^{i+1}(w) = \{u \mid u \in S \wedge \exists s \in Sy^i(w) \wedge \alpha(u) \cap \alpha(s) \neq \phi\}$. Formally, $Syn^k(w) = \bigcup_{i=1}^{k} Sy^i(w)$ for $i \in [1 \ldots k]$.

The example in Figure 4 illustrates the computation. The word مَاء $mā'$ is related to نضح $nḍḥ$ through the gloss intersection `water`. The Syn^2 feature relates the words مَاء $mā'$ and "رشّ $ršš$ " since نضح $nḍḥ$ and رشّ $ršš$ have the gloss intersection `spray`.

MRE: Morphology-based regular expressions

Let $\mathcal{O} = \{isA, contains\}$ be the set of atomic term predicates, and let $\mathcal{F} = \{\mathcal{P}, \mathcal{S}, \mathcal{X}, POS, GLOSS, CAT\}$ be the set of morphological features. Given a word w, a morphological feature $A \in \mathcal{F}$, 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).

- $a(w) := \exists m \in M(w).m = \langle p, s, x, P, G, C\rangle.r \circ CF$ 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) CF.
- $a(w) := w \in Syn^k(CF), CF \in \mathcal{S}$. Informally, this checks w is an extended synonym of a stem CF.

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

- a and $\neg a$ are MBF formulae where a an MAT, where $\neg$ is the negation (complement) operator.
- $(f \vee g)$ is an MBF where f and g are MBF formulae, and $\vee$ is the disjunction (union) operator.

Formula N in Figure 2 checks whether a solution has a category feature matching category `Name_of_Person`. Formula R is the disjunction of MAT terms that check for solutions matching stems such as قرب*qrb* (near) and في*fy* (in).

A morphology-based regular expression (MRE) is one of the following.

- m is an MRE where m is an MBF.
- fg is an MRE denoting a concatenation or a sequencing operation where f and g are both MRE expressions. This is satisfied by a match of f followed by a match of g.
- $f*, f+, f\hat{}x$, and $f?$ are MRE expressions where f is an MRE, and are satisfied by zero or more matches, one of more matches, up to x matches, and zero or only one match of f, respectively.
- $f\&g$, (conjunction) and $f|g$ (disjunction) are MRE expressions where f and g are MRE expressions, and are satisfied by a a match of both f and g, and a match of either f or g, respectively.

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

User-defined relations and actions

A user-defined relation R defined by a tuple $\langle e_1, e_2, r\rangle$ where e_1, e_2, and r are identifiers associated with MRE subexpressions in as expression f is 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 refer to a member of $R = [\![\langle e_1, e_2, r\rangle]\!]$ as a user defined relational entity.

MERF allows advanced users to write C++ code snippets to process matches of MRE subexpressions. Users can use these actions to compute statistical features, store intermediate results, or apply intelligent entity inference techniques as we

show in the numerical extraction example of Section 7. MERF provides an API that enrich the actions with detailed access to all solution features of an MRE or an MBF match including text, position, length, equivalent numerical value if applicable, and morphological features.

Once MERF computes all match trees, it traverses each match to execute the user defined `pre-match` actions in pre-order manner and the `on-match` actions in post-order manner.

MERF simulator

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 each word $t_i \in T, 0 \leqslant i < n, n = |T|$ MERF computes a Boolean value ($\{true, false\}$) for all MBFs. Then, it computes the set of MBF tags $R_i = \{(t_i, tt) | tt = \langle l, f, d\rangle \wedge f \textit{ is an MBF} \wedge f(t_i)\} \subseteq T \times \mathcal{T}$ 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_o has no tag type match, its tag set R_o is by default the singleton $O = \{NONE\}$ and t_o is referred to as a *null* word.

For each MRE, MERF generates its equivalent non-deterministic finite automaton (NFA) in the typical manner (Sipser, 2006). We support the upto operation ($f\hat{}x$), which is not directly supported in (Sipser, 2006), by expanding it into a regular expression form; for example $f\hat{}3$ is equivalent to $f?|ff|fff$. Consider the example of Figure 2 and the MRE $dir = (P|N) + \ O? \ R \ O \wedge 2 \ (P|N|U)+$. Figure 5 shows part of the corresponding NFA where $q_8, q_9, \ldots, q_{13}$ represent NFA states, and edges are transitions based labeled with MBF tags such as P, and N. Edges labeled with the empty string ϵ are non-deterministic. The expression uses operations |, ?, +, and $\wedge$ to relate places `P`, names of persons `N`, relative positions `R`, numerical terms `U`, and other `O`.

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 tree where the root is the MRE expression, the internal nodes are the MRE and MBF operations, and the leaves are matches of the MAT terms of f. The leave matches form 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 where $r_\ell \in R_\ell, 0 \leqslant k \leqslant \ell \leqslant j < n$. If we have more than one match MERF returns the longest. Figure 2(b) shows two match trees of *dir* extracted from the text of Figure 1. دبي*dby* and مول*mwl* are leaf nodes referring to name of place tags (P). The +, |, and ? MRE operations are internal nodes.

MERF computes the relational entities in a user defined relation $R = [\![\langle e_1, e_2, r\rangle]\!] \subseteq [\![e_1]\!] \times [\![e_2]\!] \times [\![r]\!]$ to be the elements of $[\![e_1]\!] \times [\![e_2]\!] \times [\![r]\!]$ with the smallest nonzero positive distance between the source and the destination where the distance is the number of words between the matches.

In Figure 2(b), MERF names the subexpressions $(P|N)+$, $(P|N|U)+$, $O?$, $O \wedge 2$, and R, as e_1, e_2, o_1, o_2, and r, respectively. The user defines the semantic relations $\langle e_1, e_2, r\rangle$, $\langle r, e_1, o_1\rangle$, and $\langle r, e_2, o_2\rangle$.

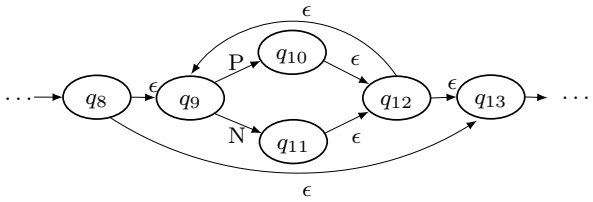

Fig. 5 Equivalent NFA of direction expression

The matches of e_1, e_2, o_1, o_2, and r from the second match tree of Figure 2(b) are دبي مول *dby mwl* (Dubai Mall), المبنَى *ālmbnā* (the building), علَى *ʿlā* (prep), من هذَا *mn hḏā* (from this), and مقربة *mqrbh* (near), respectively. MERF constructs the semantic relation matches and builds the lower part of the entity-relation graph shown in Figure 1.

For the first match برج خليفة بَالقرب من التقَاطع الأول *brǧ ẖlyfh bālqrb mn āltqāṭʿ āl-ʾwl* , the matches of e_1, e_2, o_2, and r are برج خليفة *brǧ ẖlyfh* (Khalifa tower), التقَاطع الأول *āltqāṭʿ āl-ʾwl* (intersection 1), من *mn* (from), and بَالقرب *bālqrb* (next to), respectively. MERF doesn't construct the relation $\langle r, e_1, o_1 \rangle$ since o_1 has no match. Therefore, we get the upper part of the entity-relation graph shown in Figure 1.

After computing the relational entities, MERF computes a cross-reference relation between the extracted entities using a second order synonymy feature (Syn^2). The `isA` edge in the graph of Figure 1 shows the cross-reference relation between برج خليفة *brǧ ẖlyfh* (Khalifa tower) from the first match with المبنَى *ālmbnā* (the building) from the second match.

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.

MERF saves the tags and the tag types in a user friendly format using the JavaScript Object Notation (JSON) format (Nolan and Lang, 2014). The tag file keeps the paths of the separate text and tag type files. It also contains a list of tags with their tag type identifiers, position in text, length, and word index. The tag type file contains MBF and MRE tag types. Each tag type contains the name, description, expression, and the visualization legends.

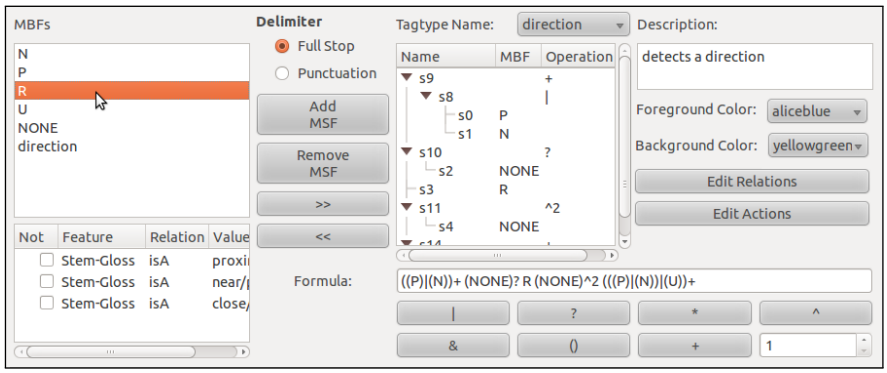

Fig. 6 MERF tag type regular expression editor.

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.

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.

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

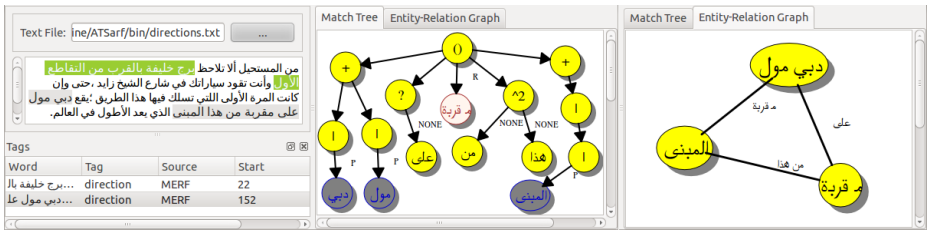

Fig. 7 MRE annotated Text, MRE match tree, and entity-relation graph

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 sub-expression.

To specify a binary operation the user selects two sub-expressions 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.

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.

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

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 sub expressions and select features of the matches of the sub expressions 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 match 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.

Table 2 Comaprison of MERF with SystemT(Chiticariu et al., 2010), TEXTMARKER(Atzmueller et al., 2008), Urbain(Urbain, 2012), QARAB(Hammo et al., 2002)

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

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: 1) the common tag types $\mathcal{T}_1 \cap \mathcal{T}_2$, 2) the tag types in $\mathcal{T}_1$ but not in $\mathcal{T}_2$, and 3) 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.

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. CPSL consists of (1) a declaration part specifying names and labels, (2) a rule definition part specifying patterns, regular expressions and associated actions, and (3) and macro text substitution part (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 performance and expressivity 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 allows for overcoming the overlapping annotation problem.

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. Then it extends the extracted entities into a query model, and uses the query model to extract matches from the document. 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 that computes limited features of Arabic words compared to the morphological analysis of MERF. QARAB then classifies the query against a set of known question types.

Table 2 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 quickly building classifiers for text processing tasks using active learning and semi-supervised learning (Settles, 2011). The classifier is interactive as it queries the annotator for entity detetction correction and annotation correction. 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.

WordNet is a lexical reference system that mimics human lexical memory and that 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. An overview of annotation tools and their Arabic-English word alignment issues concludes with a set of rules and guidelines needed in an Arabic annotation alignment tool (Kholidy and Chatterjee, 2010). The work in (Dukes et al., 2013) presents a collaborative effort towards morphological and syntactic annotation of the Quran. (Dorr et al., 2010) presents 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.

MMAX2 is a manual multi-level linguistic annotation tool with an XML based data model (Müller and Strube, 2006). It enables the user to create, browse, visualize, and query annotations and may be able to resolve coreference tags. 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 (Stenetorp et al., 2012). They can be extended through plug-ins to enable automatic annotators and customized annotation visualization specifications. 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 ontologies. However, it doesn't support the Arabic language.

Table 3 MERF compared to task specific applications.

Task	Development time	Run time(s)	Accuracy Recall	Accuracy Precision	Ease of Composition
ANGE (Zaraket and Makhlouta, 2012c)	2 months	1.79	0.99	0.99	3000+ lines of code
MERF	3 hours	7.24	0.99	0.93	8 MBFs and 4 MREs
ATEEMA (Zaraket and Makhlouta, 2012a)	1.5 months	2.53	0.88	0.89	1000+ lines of code
MERF	3 hours	3.14	0.91	0.81	3 MBFs and 2 MREs
Genealogy tree (Makhlouta and et al., 2012)	3 weeks	0.74	0.96	0.98	3000+ lines of code
MERF	4 hours	2.28	0.84	0.93	3 MBFs and 3 MREs
NUMNORM	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

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.

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) uses 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 compare MERF implementations of the narrator chain, temporal entity, and genealogy entity extration tasks to the task specific techniques proposed to solve them in ANGE (Zaraket and Makhlouta, 2012c), ATEEMA (Zaraket and Makhlouta, 2012a), and GENTREE (Makhlouta and et al., 2012), respectively. We also compare a MERF number normalization task to a task specific implementation. In the online appendix [2], we report on eight additional MERF case studies.

Table 3 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.

For the temporal and number normalization cases, we evaluated the techniques against arbitrary text from issues of the Lebanese Assafir and Al-Akhbar newspapers [3]. For the narrator chain case, we used Musnad Ahmad, a hadith book,

[2] available at `http://webfea.fea.aub.edu.lb/fadi/pdfs/merfappendix.pdf`

[3] available at `http://www.assafir.com` and `http://www.al-akhbar.com`.

Table 4 MERF MBF and user-defined relation accuracy

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

for evaluation. For the genealogical tree extraction we used an extract from the Genesis biblical text.

Table 3 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 2 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.

MERF required reasonably more runtime than the task specific implementations and reported acceptable and slightly less precision metrics with around the same recall values.

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 named entities and relations including narrator chains (Zaraket and Makhlouta, 2012c).

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 صلّى, الله, علي, and سلّم, respectively.

Table 5 presents the defined MREs. 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 fam-

Table 5 Narrator chain example.

```
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*	*ğryr*	*ḥdṯnā*	*sʿyd*	*bn*	*qtybh*	*ḥdṯnā*
PN	FAM	PN	TOLD	PN	TOLD	PN	FAM	PN	TOLD
name		name		name		name		name	
nar				nar		nar			
nchain									

ily 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.

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 sub-expressions `TOLD` and `nar`, respectively. We define the relation $\langle s_2, s_2', s_1 \rangle$ to relate sequences of narrators with edges labelled by the tags of `TOLD` where s_2' denotes the next match of `nar` in the one or more MRE subexpression.

Table 4 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.

For brevity, we omit the details of MERF temporal entity extraction, genealogy tree, and number normalization case studies, describe them shortly below and provide a full description in the online Appendix.

Temporal entity extraction

Temporal entities are text chunks that express temporal information. Some represent absolute time such as ٢٠١٠ آب من الخَامس *ālḫāms mn ʿāb 2010* . Others represent relative time such as بعد خمسة أَيَام *bʿd ḫmsh ʿayām* , and quantities such as يومًا ١٤ *14 ywmā* . *ATEEMA* presents a temporal entity detection technique for the Arabic language using morphological analysis and finite state transducers (Zaraket and Makhlouta, 2012a).

Table 4 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.

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;
```

Fig. 8 Actions for TMB, DT, and H MRE expressions.

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 وولد هَارَان لوطَا *w wld hārān lwṭā* meaning "and Haran became the father of Lot".

GENTREE (Makhlouta and et al., 2012) automatically extracts the genealogical family trees using morphology, finite state machines, and graph transformations. Table 4 shows that MERF detected MBF matches with 99% recall, and 75% precision, and extracted user-defined relations with 81% recall and 96% precision.

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 sub-expressions `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 the text (`$s1.text`) and the numeric value (`$s1.number`). of literal numbers such as digits and numbers from one to ten.

Table 4 shows high accuracy in MBF tagging with 99% recall and 99% precision, and high accuracy in user-defined relation extraction with 97% recall and 95% precision.

8 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 edit the corpora and fix the annotations.

9 Conclusion

In this work, we presented a morphology-based entity and relational entity information extraction framework for Arabic text; MERF. MERF provides a user-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 sub-expression and to define semantic relations between sub-expressions. MERF uses Sarf, an Arabic morphological analyzer, to compute morphological and thereafter regular expression matches, and relational entities. We evaluated MERF with several case studies and compared 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.

Currently, MERF supports one built in cross-reference predicate based on the Syn^2 feature. In the future, MERF will support user-defined cross-reference predicates. Currently the user selects the morphological features to specify the MBF. We will explore techniques that can infer the features from example words that the user judges as relevant to the basic concept in question.

References

S. AbdelRahman, M. Elarnaoty, M. Magdy, and A. Fahmy. Integrated machine learning techniques for arabic named entity recognition. *International Journal of Computer Science Issues*, 7(4):27–36, 2010.

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

M. Alrahabi, A. H. Ibrahim, and J.-P. Desclés. Semantic annotation of reported information in arabic. In *FLAIRS Conference*, volume 6, pages 263–268, 2006.

D. Appelt and B. Onyshkevych. The common pattern specification language. In *TIPSTER workshop*. ACL, 1998.
M. Attia, M. A. Rashwan, and M. Al-Badrashiny. Fassieh¯, a semi-automatic visual interactive tool for morphological, pos-tags, phonetic, and semantic annotation of arabic text corpora. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(5):916–925, 2009.
M. Atzmueller, P. Kluegl, and F. Puppe. Rule-based information extraction for structured data acquisition using textmarker. In *Proceedings of LWA*, pages 1–7. Citeseer, 2008.
T. Buckwalter. Buckwalter Arabic morphological analyzer version 1.0. Technical report, 2002.
L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *Proceedings of the Association for Computational Linguistics*, pages 128–137, 2010.
B. J. Dorr, R. J. Passonneau, D. Farwell, R. Green, N. Habash, S. Helmreich, E. Hovy, L. Levin, K. J. Miller, T. Mitamura, et al. Interlingual annotation of parallel text corpora: a new framework for annotation and evaluation. *Natural Language Engineering*, 16(3):197–243, 2010.
K. Dukes, E. Atwell, and N. Habash. Supervised collaboration for syntactic annotation of quranic arabic. *Language resources and evaluation*, 47(1):33–62, 2013.
A. Ekbal and S. Bandyopadhyay. Named entity recognition using support vector machine: A language independent approach. *IJCSSE*, 4(2), 2008.
S. Ferilli. Natural language processing. In *Automatic Digital Document Processing and Management*. Springer, 2011.
J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. Tu. The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123, 2003.
N. Habash. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 2010.
N. Habash and F. Sadat. Arabic preprocessing schemes for statistical machine translation. In *NAACL*, pages 49–52, 2006.
B. Hammo, H. Abu-Salem, and S. Lytinen. Qarab: A question answering system to support the arabic language. In *Proceedings of the workshop on Computational approaches to semitic languages*, pages 1–11. ACL, 2002.
A. Jaber and F. Zaraket. MATAr: Morphology-based tagger for arabic. In *AICCSA*, Fes, Morocco, May 2013.
H. Kholidy and N. Chatterjee. Towards developing an Arabic word alignment annotation tool with some Arabic alignment guidelines. In *ISDA*, pages 778–783. IEEE, 2010.
S. Kulick. Consistent and flexible integration of morphological annotation in the arabic treebank. *LREC*, 2010.
S. Linckels and C. Meinel. Natural language processing. In *E-Librarian Service*, pages 61–79. Springer, 2011.
M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, 2004.
J. Makhlouta and et al. Arabic entity graph extraction using morphology, finite state machines, and graph transformations. In *CICLing*, 2012.
M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.
G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244, 1990.
T. Morton and J. LaCivita. Wordfreak: an open tool for linguistic annotation. In *HLT/NAACL*. ACL, 2003.
C. Müller and M. Strube. Multi-level annotation of linguistic data with MMAX2. *Corpus technology and language pedagogy: New resources, new tools, new methods*, 3:197–214, 2006.
D. Nolan and D. Lang. Javascript object notation. In *XML and Web Technologies for Data Sciences with R*. Springer, 2014.
P. Ogren. Knowtator: a protégé plug-in for annotated corpus construction. In *NAACL-Demonstrations*. ACL, 2006.
L. Saleh and H. Al-Khalifa. AraTation: an Arabic semantic annotation tool. In *Proceedings of IIWAS*. ACM, 2009.
B. Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of EMNLP*, pages 1467–1478. ACL, 2011.

M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

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

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

P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *European Chapter of the Association for Computational Linguistics Demonstrations*, pages 102–107. Association for Computational Linguistics, 2012.

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

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

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238, 2005.

W. Zaghouani, B. Pouliquen, M. Ebrahim, and R. Steinberger. Adapting a resource-light highly multilingual named entity recognition system to arabic. In *Proceedings of LREC*, pages 563–567, 2010.

F. Zaraket and J. Makhlouta. Arabic temporal entity extraction using morphological analysis. *IJCLA*, 3:121–136, 2012a.

F. Zaraket and J. Makhlouta. Arabic morphological analyzer with agglutinative affix morphemes and fusional concatenation rules. In *COLING*, Mumbai, India, December 2012b.

F. A. Zaraket and J. Makhlouta. Arabic cross-document NLP for the hadith and biography literature. In *FLAIRS*, May 2012c.