IJCLA VOL. 3, NO. 1, JAN-JUN 2012, PP. 121–136
RECEIVED 29/10/11 ACCEPTED 09/12/11 FINAL 15/06/12

Arabic Temporal Entity Extraction using Morphological Analysis

FADI ZARAKET AND JAD MAKHLOUTA

American University of Beirut, Lebanon

ABSTRACT

The detection of temporal entities within natural language texts is an interesting information extraction problem. Temporal entities help to estimate authorship dates, enhance information retrieval capabilities, detect and track topics in news articles, and augment electronic news reader experience. Research has been performed on the detection, normalization and annotation guidelines for Latin temporal entities. However, research in Arabic lags behind and is restricted to commercial tools. This paper presents a temporal entity detection technique for the Arabic language using morphological analysis and a finite state transducer. It also augments an Arabic lexicon with 550 tags that identify 12 temporal morphological categories. The technique reports a temporal entity detection success of 94.6% recall and 84.2% precision, and a temporal entity boundary detection success of 89.7% recall and 90.8% precision.

INTRODUCTION

This paper considers the problem of extracting temporal entities from Arabic text documents. Temporal entities are text chunks that express or infer temporal information. Some entities represent absolute time and dates such as 07/17/2011 or الخامس من آب ٢٠١٠ *ālḫāms mn ʾāb 2010* (August 5, 2010). Some entities represent relative time such as بعد خمسة أيام *bʿd ḫmst ʾayām* (after five days). Other entities represent temporal quantities such as عطلته ١٤ يوما *ʿṭlth 14 ywmā* (his vacation is 14 days).

Industry tools that extract temporal entities from Arabic texts exist [1,2,3]. However, the techniques underlying these tools have not been disclosed and evaluated academically yet. We present the *Arabic temporal entity extractor using morphological analysis* (ATEEMA). To the best of our knowledge, ATEEMA is the first open-source tool to perform the task of temporal entity extraction.

Temporal entity extraction from text includes the task of identifying the temporal chunks of text and then the task of normalizing a temporal chunk into a time quantity structure. Temporal entity recognition detects expressions that express time concepts. Temporal entity normalization understands the temporal chunck and extracts from it a time structure so that ١٩٠٠ *1900*, القرن التاسع عشر *ālqrn āltās‘ šr* (the nineteenth century), and ألف وتسعمئة بعد الميلاد *ʾalf wtsʿmʾyt bʿd ālmylād* (one thousand nine hundred after Christ) all have one normalized canonical form. Ultimately, one may want to store the normal form in a database and process it later.

Research on temporal entity extraction in those languages that use Latin alphabet, such as English, German, French, or Spanish, uses local grammars, finite state automata [4,5,6,7,8,9,10], and neural networks [11] to detect temporal entities. These techniques do not work well directly for Arabic due mainly to the rich morphology and high ambiguity rate of Arabic.

This paper focuses on the task of temporal entity recognition. We will target temporal entity normalization in future work. ATEEMA uses Arabic morphological analysis with part of speech (POS), gloss tagging, and augmented temporal tagging to capture morphological temporal features of Arabic text. ATEEMA passes the temporal morphological features to a knowledge-based finite state transducer that captures temporal entities and detect their boundaries.

In this paper we make the following contributions. **(1)** We present a novel technique for temporal entity extraction from Arabic text based on morphological analysis and finite state transducers. **(2)** We augment an Arabic lexicon with 550 temporal morphological tags that identify 12 separate temporal categories. And **(3)** We provide the first open source temporal entity extractor for Arabic.

Motivation

In this section we discuss several interesting applications that depend on temporal entity extraction to motivate our work. The detection of time

stamps from the content of a digital document rather than its meta-data, such as the last modification and the creation dates of a file, is of interest to the research community. Li et al. from Microsoft filed a patent for their application that extracts authorship dates. They hypothesize that the last modification date of a document is not representative of its date of generation as documents may be uploaded or copied to collaborative websites and the meta-date gets changed to the upload date, which is rarely significant [11].

The work in [5] considers the problem of automatic assignment of event-time periods in documents such as newspaper articles, medical reports and legal documents to facilitate documemt retreival. Time periods of the events under consideration in a document may be of higher importance than the date of the writing of the document.

Extraction of temporal entities from news articles combined with a user profile can help augment the articles with information of interest to the user [4]. For example, the extracted temporal entities can form a navigation timeline that refers to other articles of relevance, and also help augment the articles with answers to queries such as “Where were I when the event took place?”, and “What other events took place at the same time in my neighborhood?”.

German researchers developed an appointment scheduling via emails (COSMA) application based on temporal entity extraction that takes as input the electronic calendar of several parties, automates the time consuming task of scheduling meetings, and reduces the number of correspondence needed to agree on a time [12].

When time entities are extracted, other data mining techniques can discover useful relationships such as the recognition of frequent temporal patterns in newspapers [13], the discovery of interesting events from time varying features in the news corpus [14], and the detection and tracking of new evolving events in the news [15].

Other benefits of temporal extraction include the comparison of parallel accounts and narrations of the same events such as several history books that address the same periods. With temporal entity extraction one can automatically check and detect historical inconsistencies if they exist. Extracted temporal entities can help to sort several different accounts of the same or similar events into a timeline. This can automate merging complementary and partial accounts of the same events. This analysis, such as merging the several narrations of the Bible, was performed manually.

Background

In this section we briefly describe morphological analysis, finite state transducers, and local grammars that are used to extract temporal entities from text.

Morphological analysis. Morphology considers the composition of a word from several *morphemes*. A morpheme is a *stem* or an *affix*. An affix is a *prefix, suffix,* or an *infix*. Prefixes and suffixes are attached to the beginning and end of the word respectively; while infixes introduce changes within the stem. A morpheme is associated with several tags such as *part of speech* (POS) and gloss tags. Words result from the concatenation of compatible morphemes. Morphological analysis helps to group together words which express similar notions such as شهر، شهرين، اشهر، الاشهر، الشهرين that all share a common stem with a gloss tag of 'month', and have affixes with useful glosses such as plural, dual, and definite article ال indicators.

Morphological analysis is is used in techniques that detect Arabic named entities such as proper names [16,17]. It is key and necessary in Arabic entity recognition due to the morphological richness of Arabic. For instance, consider the stem إنتهاء *ʾintihāʾ* (end) with the the suffix ه *-h* (it). Both variations إنتهائه *ʾintihāʾih* and إنتهاؤه *ʾintihāʾoh* are legal based on the context. In addition, morphology isolates clitics such as ف *fa* (so) and و *wa* (and) from other morphemes as in واليوم *wa-āl-yawm* (and + today).

Local grammars. *Local grammars* have been used to extract entities from text [4,5,7]. Local grammars are lexical and syntactic constraints expressed in regular expressions that precisely define the local neighborhood and context of an entity. The rules ignore the rest of the context such as the complete sentence, paragraph, and document. Local grammars simplify the detection of target entities when the entities can be expressed by local small scope rules. They relieve the user from the task of modeling the full language and the expense of understanding the whole context [18].

A local grammar for temporal English expressions includes several rules such as `(Num ≤ 12) "in the afternoon"` that can detect "five in the afternoon", `(Num ≤ 12) "p.m"` that can detect "5 p.m.", `"half past" (Num ≤ 12)` that can detect "half past one", and `(Num ≤ 23) ``:'' (Num ≤ 59)` that can detect "5:44". The quoted subexpressions within the rules are *frozen* expressions and `Num` captures numbers such as 'one', 'two', '1' and '2' [19].

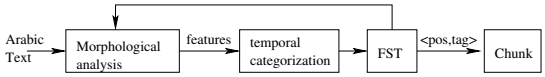

Fig. 1. ATEEMA flow diagram

Finite state transducers. A finite state transducer (FST) is a finite state machine with an input and an output tape. FSTs differ from finite state automata (FSA) in that they have an output tape while FSAs have accept states instead.

Formally, an FST is a tuple $M = (S, S_0, \Sigma, \Gamma, \delta)$ where S is the set of states, $S_0 \subset S$ is the set of initial states, Σ is the input alphabet, Γ is the output alphabet, and $\delta \subseteq S \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times S$ is the transition relation. FSTs have been used extensively in text mining applications where the input is the text and the output is the delimiters of a chunk of text with an associated class [20]. FSTs are attractive due to their efficiency and ease of use.

ATEEMA

The diagram in Figure 1 shows how ATEEMA works. ATEEMA takes as input Arabic text and returns temporal entities therein. ATEEMA passes the input text string to an inhouse Arabic morphological analyzer. Both the analyzer and ATEEMA are both available as open source tools. The analyzer processes the input text and whenever it identifies a morpheme, it calls ATEEMA back with the current solution context and the found morpheme. Note that several morphological solutions may exist for the same input string.

The ATEEMA call back receives the solution context and the morpheme with the associated POS and gloss tags. It either adds the morpheme to a sequence of unresolved morphemes, or resolves the morphemes and produces a temporal category as input to the finite state transducer (FST). Note that this is necessary since words in the Arabic language are not necessarily separated by white space delimiters and thus one white space delimited token may contain several words and produce several categories.

The finite state transducer detects the temporal entities. ATEEMA uses a manually built finite state transducer to accomodate for **(1)** morphological variations which are frequent in Arabic text, and **(2)** ambiguous morphological solutions since Arabic is at least one order of mag-

Table 1. Temporal categories added to the lexicon of the Arabic morphological analyzer.

	TIME					*NUM*			*TIME_PREP*			
Tag	unit	relative	range	nominal	events	digit	word	range	point	relative	approximate	range
Count	32	55	16	94	10	20	152	-	83	54	12	22
Total	207					172			171			

nitude more ambiguous than Latin languages. For example, consider the following morphological variations of entities related to temporal expressions.

- آخِر، اخِر، أَواخِر، أَخِير *ʾāḫir, āḫir, ʾawāḫir, ʾaḫiyr* (last)
- بَدْء، بَدْأ، بَدْؤ، بَدْئ، ابْتِداء *badʾ, badʾa, badʾw, badʾy, ābtidāʾ* (start)

Also consider the ambiguous word إثنين *ʾiṯnyn* which means Monday and the number two. Techniques that use local grammars and work for Latin languages are restricted to regular expressions with frozen phrases and fail upon morphological variations. We believe that our approach can be extended to other morphologically rich languages as well as Latin languages to provide better results.

ATEEMA targets the detection of temporal expressions within the bounds of their temporal context. Consider the following examples.

- ٤ / ٩ / ٢٠٠٣ (4/9/2003): this is a straight forward date with no temporal context.
- بعد ٤ / ٩ / ٢٠٠٣ (*bʿd* after 4/9/2003): this is a date with a temporal preposition.
- في نيسان ٢٠٠٣ *fy nysān 2003* , (in April 2003): this is a partial date with a temporal preposition.
- بعد مرور نحو أربعة أشهر *bʿd mrwr nḥw ʾarbʿa ʾašhr* (after about four months passed): this is an approximate (نحو) range (مرور) with a temporal preposition.
- في ثمانينيات القرن الماضي *fy ṯmānynyāt ālqrn ālmāḍy* , (in the eighties of the last century): this is a temporal expression where the range is inferred from the suffix ـيات of ثمانينيات.

Temporal categories. The Arabic morphological analyzer produces morphemes with POS and gloss tags. ATEEMA is interested in temporal and numerical morphological features as well as temporal prepositions which occur within a neighborhood of temporal expressions. For this purpose we augmented the tags of the morphological analyzer to report the categories presented in Table 1.

Category *TIME* denotes explicit temporal tokens. They are infered from time units such as دقيقة *dqyqt* (minute) and ساعة *sāʕ* (hour), relative entities such as غد *ġd* (tomorrow), nominal entities as in day and month names, range entities as in seasons, and referential entities as in important events هجري *hğry* . Category *NUM* denotes numerals and refers to digits or words such as ثمانين *ṯmānyn* (eighty), أربعة *ʔarbʕ* (four) and ٤ *4* . The prefixes and suffixes of these numbers may divide them into subcategories denoting range as in ثمانينيات. Category *TIME_PREP* denotes temporal prepositions that precede or follow time expressions. They are divided into relative prepositions such as قبل *qabil* (before) and منذ *mnḏ* (since), approximate prepositions such as نحو *nḥw* (about), range prepositions such as خلال *ḫlāl* (during), and point prepositions such as في *fy* (in). The Table reports also the number of lexicon items we tagged with temporal tags. We identified 550 lexicon entries out of 82,170 entries and annotated them with temporal tags. Note that these entries are stems and affixes and can be concatenated with other entities to generate more possible temporal tokens.

ATEEMA associates an order amongst the temporal morphological annotations and categories. ATEEMA uses the order to associate complex temporal entities with a tag that is the result of the concatenation of the morphological temporal tags and categories. This produces a semi-canonical temporal form that can be used in practice as a normalized temporal form.

The finite state transducer

Figure 2 shows a high level overview of the finite state transducer. The FST takes the detected temporal categories as input and also considers two state variables iw_t and iw_m that count the number of temporal categories met while in the *Time* and *Maybe Time* states, respectively. The actual number of states is proportional to θ_t and θ_m which are two thresholds that define the upper limits for the iw_t and iw_m counters, respectively.

The initial state *Nothing* denotes that no temporal entity is being processed. The *Maybe Time* state denotes the cases where the FST is not yet sure whether the currently processed text is a temporal entity. Finally the *Time* state denotes that the FST has recognized a temporal entity and is in the process of computing its boundary.

The transition to the *Maybe Time* state happens when the FST encounters prepositions or numbers but no direct time tokens. Once it detects a

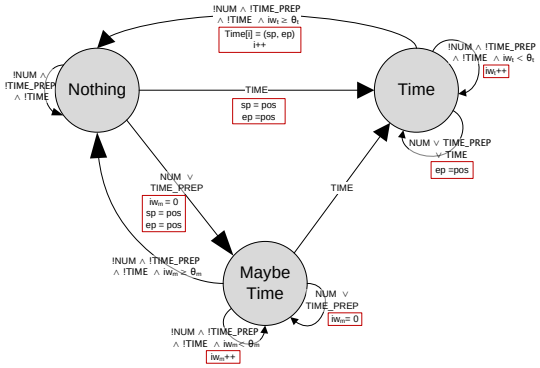

Fig. 2. FST for temporal entity extraction

direct time token, the FST transitions to the *Time* state. This might happen also directly from the *Nothing* state.

The two thresholds θ_m and θ_t encode the flexibility and tolerance of ATEEMA to recognize the several different temporal classes. The θ_m threshold specifies the number of non-temporal tokens that ATEEMA tolerates within a temporal entity. The threshold θ_m is tested before a transition takes place from the *Maybe Time* state to the *Nothing* state. In other words, if the FST suspected a temporal entity because of an indirect temporal feature (*NUM or TIME_PREP*), then the FST will give up that entity if it meets more than θ_m non-temporal tokens before meeting a direct temporal feature. For example, the word خمسين in the text دفع خمسين درهما ليشتري الحنطة والتمر *dfʿ ẖmsyn drhmā lyštry ālḥnṭt wāltmr* (He paid fifty Durhams to buy flour and dates) will activate a transition to a *Maybe Time* state but the rest of the text will drive the FST back to the *Nothing* state.

The θ_t threshold expresses the number of non-temporal words that might intervene between two temporal words in the same temporal entity. For example, the text بعد أربعة أشهر عجاف وخمس من أصعب السنوات *bʿd ʾarbʿ ʾašhr ʿǧāf wẖms mn ʾaṣʿb ālsnwāt* (after four bad months and five of the hardest years) contains non-temporal words (عجاف and من أصعب) that θ_t tolerates.

In Figure 2, actions that are performed on the different transitions are shown inside boxes near the transitions. The actions include setting the values of the counters iw_m, iw_t, and i. The counter iw_m denotes the number of non-temporal tokens met within the *Maybe Time* state. The

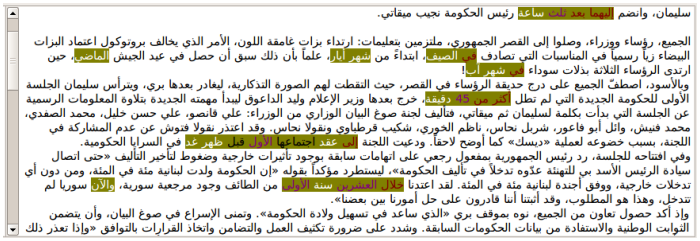

Fig. 3. Screenshot from ATEEMA.

FST increments iw_m if it was in the *Maybe Time* and a non-temporal word was detected, and resets it when entering the *Maybe Time* state or when a temporal word is detected. The counter iw_t denotes the number of non-temporal words met within the *Time* state and is updated in a similar fashion to iw_m.

The counter i is the index of the current temporal expression. Once a temporal expression is fully detected, the FST appends it to the `Time` structure and increments i. The actions also include setting the the start sp and end ep positions of the temporal expression. The symbol pos refers to the current position in the text. Figure 3 shows the output of ATEEMA with context sensitive coloring of the extracted temporal entities.

Heuristic optimizations

After a first run of ATEEMA, we made few observations that led us to introduce the following disambiguation heuristics.

Heuristic 1. We ignore the word العام *āl-ʿām* (the year) if it appears in the definite form (i.e. العام *āl-ʿām* instead of عام *ʿām*) and no other temporal words or prepositions appeared next to it. This is mainly because in newspapers, it is common to find the word العام *āl-ʿām* (the year) with a completely non-temporal meaning (i.e. general, common, and public) as in المال العام *āl-māl āl-ʿām* (public money) and الرأي العام *āl-raʾyi āl-ʿām* (public opinion). The same rule applies to الثانية *āl-ṯāniyat* (second) which may mean either a second rank (2^{nd}) or the time unit (second).

Heuristic 2. The word الأحد *āl-ʾaḥad* has both 'the one' and 'Sunday' meanings. When used in the indefinite form (i.e. أحد *ʾaḥad*), we interpreted the word as a number. When used in the definite form (i.e. الأحد *āl-ʾaḥad*), we interpreted the word as a day of the week.

Table 2. Temporal entity extraction accuracy results for Safir and Akhbar newspapers.

		Detection			Boundary			Statistics	
		recall	precision	F-score	recall	precision	F-score	words	entities
w/o heuristics	Safir 1	0.944	0.648	0.768	0.924	0.922	0.923	5,783	72
	Safir 2	0.926	0.680	0.784	0.829	0.881	0.854	13,457	108
	Akhbar	0.954	0.692	0.802	0.899	0.926	0.912	28,688	503
		0.949	0.685	0.796	0.890	0.918	0.904	47,928	683
with heuristics	Safir 1	0.944	0.850	0.895	0.924	0.931	0.928	5,783	72
	Safir 2	0.926	0.813	0.866	0.829	0.881	0.854	13,457	108
	Akhbar	0.950	0.847	0.895	0.908	0.926	0.917	28,688	503
		0.946	0.842	0.891	0.897	0.919	0.908	47,928	683

RESULTS

Similar to the literature, we chose newspapers as our evaluation corpora. Temporal entities are abundant in newspaper texts. We evaluated ATEEMA against text chosen arbitrarily from two issues of the Lebanese Assafir newspaper [1] and one issue of the Lebanese Al-Akhbar newspaper [2]. Table 2 shows the results for Safir 1, the youth section of the Safir 18/5/2011 issue, Safir 2 cultural section of the Safir 20/5/2011 issue, and Akhbar, the politics section of the Akhbar 17/6/2011 issue. As shown in the Statistics columns of Table 2 the three data sets consisted of about 48 thousand words and contained about 680 temporal entities. We evaluated the accuracy of ATEEMA by comparing the output of ATEEMA against a manually tagged version of the text.

We used recall and precision as our evaluation metrics. We report results for the detection and for the boundaries of temporal entities. Detection accuracy refers to the success of ATEEMA in detecting the temporal entities in the text of the newspaper. Boundary accuracy refers to correctly reporting the start and end positions of the extracted temporal entity in text.

Detection recall refers to the fraction of the temporal entities correctly detected against the total number of temporal entities available. Detection precision refers to the fraction of correctly detected expressions against the total number of extracted temporal entities. The same applies to boundary recall and precision measures. Intuitively, the precision measure denotes whether the system generated false positives.

[1] available online at *www.assafir.com*

[2] available online at *www.al-akhbar.com*

The upper rows of Table 2 show that ATEEMA without the heuristics has a high recall (about 95%) but a relatively low precision (about 69%) on the average. This means that ATEEMA detects almost all temporal expressions due to the flexible nature of the FST which supports captures most temporal expression structures.

The expressions ATEEMA missed such as خلال الحرب الأهلية اللبنانية *ẖlāl ālḥrb āl‘ahlyt āllbnānyt* (during the Lebanese civil war), and بعد انهيار جدران برلين *b‘d ānhyār ǧdrān brlyn* (after the collapse of the walls of Berlin) needed a deep understanding of the text under consideration that is hard to automate.

ATEEMA also scored 89% and 91.2% for the boundary recall and boundary precision respectively. This means that ATEEMA is capable of detecting the great majority of the temporal entities without significantly over-approximating or under-approximating the boundaries of the entity.

The heuristics improved the precision of ATEEMA with practically no loss in the recall metrics as shown in the lower rows of Table 2. ATEEMA achieved more than 84% precision without extensively enumerating all the temporal expression structures. This is mainly due to the fact that temporal features such as time, numbers, and time prepositions occurring in a neighborhood of text are precise at capturing temporal entities when used with an adequate morphological generalization.

We observed that more than 60% of the detected entities exhibited one or more morphological variations. These entities would not have been detected without the use of the morphological analyzer.

ATEEMA reported false positives in cases similar to نحن بأمس الحاجة إلى *nḥn b‘ams ālḥāǧt ‘ilā* (we desperately need) since the morphological analyzer reported a temporal morphological feature for أمس as it also means yesterday. Local grammars can not address such failures as well.

RELATED WORK

Much research considered the detection of temporal entities in Latin text especially English [4,5,6,7,8,9,10,11]. Only commercial tools exist with temporal entity support for the Arabic Language [1,3,2].

Gross [21] studied collocations and lexically *frozen phrases* that can be modeled using formal grammar rules and proposed methods to represent the rules with finite state automata efficiently. A frozen phrase is “a phrase in which certain parts cannot be altered. These parts are subject to restricted syntactical variations without affecting the original meaning

or function of the expression" [22]. Later Matthieu Constant uses local grammars for text parsing [19], and presented a finite state automaton that parses date expressions and that is able to capture expressions such as "five in the afternoon", "5 p.m.", and "half past one".

Llidó et al. [5] presented techniques to extract and normalize temporal entities using local grammars. The extraction phase uses a shallow semantic-syntactic parser. The normalization phase encodes the semantic meaning of the extracted phrases in terms of a hierarchical formal time model that supports temporal points, intervals, and relative temporal entities.

Koen argues that local shallow parsing is better suited for temporal entity extraction and that full grammar parsing can improve the results if applied afterwards to understand the context of located temporal entities [4]. He reported about 90% recall and precision on extracting date, time, interval and velocity entities. He reported 60% recall on extracting the less useful and less frequently occurring age entities. His work fills missing information in partial temporal entities with a reference extracted temporal entity such that the publication or transmission date of an article. It produces false results when a paragraph discusses events that took place at another date.

The Fact Extractor Workbench [23] uses regular expressions to model local grammars along with optimized capabilities such as caching. The work in [8] uses the IDE to extract temporal and other entities. It caches detected entities, uses the cached entities to complete partial entities, and presents better results for relative dates than that of [4]. However, when the detection of reference temporal entities is non-trivial and needs an understanding of the semantics of the document both approaches fail.

ATEEMA is similar to the work of Gross [21] and Constant [19] who encode their local grammars in efficient state machines. The ATEEMA transducer differs in that it is not grammar based, it does not use frozen phrases, and in that ATEEMA can capture temporal entities with varying structures without formally defining each structure.

ATEEMA differs from the rest of the local grammar based approaches in that it uses a manually optimized FST that takes as input a sequence of morphological features. Local grammars expressed in regular expressions have a less expressive power, are less tolerant to morphological variations, and are automatically translated into non-deterministic finite state machines that may be complex and of large size.

An information extraction core system SMES [10] addresses temporal entity extraction from German texts. SMES consists of a tokenizer

that identifies fragment patterns, a lexical morphological analyzer that supports compound expressions, and a shallow parser based on finite-state automata that parses the text to extract temporal entities [12]. It reports a recall of 77% and precision of 88% when evaluated against a corpus of monthly reports about the 'German IFOR mission in former Yugoslavia' [10].

COSMA [12] builds on SMES to provide a system for appointment scheduling via Emails. COSMA resolves partial dates and relative temporal expressions by relating the underspecified expressions to their context. The context of an expression includes text of the email, previous email messages in the same conversation, and the temporal meta-data of the conversation. COSMA considers the contextual hierarchy in order until it resolves the expression. If the process fails, COSMA asks for clarification.

ATEEMA is similar to SMES [10] and COSMA [12]. It differs in that it uses an FST to detect a neighborhood for temporal entities. It does not use frozen words and it does not enumerate several regular expressions that extensively capture temporal entities since these are hard to extensively cover in Arabic because of the rich morphology.

Time Calculus for Natural Language (TCNL) [24] extracts and normalizes English temporal entities in scheduling-related emails. The expression '$\{|1_{mon}|@\{>= \_\}\}$' expresses the phrase 'the coming Monday'. The symbol '_' denotes the temporal anchor to which this relative expression relates; e.g. the timestamp of the email. The Temporal Expression Anchorer (TEA) subsystem determines the anchor and disambiguates the expression using the context, e.g. the tense of the nearest verb. Its favors the most recent event as the anchor of subsequent expressions. The system reported an accuracy of about 80% in normalizing correctly recognized temporal expressions.

Temporal entity detection for Arabic. The Rosette Entity Extractor by Basis Technology supports extracting entities including dates for 13 languages including Arabic [1]. Rosette is based on *aided statistical machine learning* where a computational linguist provides Rosette with information about contextual features to define an entity and a tagged learning set. Rosette then builds a statistical model for extracting the entity. This methodology is language independent and is easily extendable to a large number of languages and entity types including Arabic temporal concepts.

BBN IdentiFinder Text Suite is a named entity extraction tool that supports extraction of Arabic date and time expressions [2]. IdentiFinder

seems to be based on statistical learning methods. Currently it supports English, Arabic, and Chinese. However, the methods used in this tool are language-independent just as Rosette.

Arabic Named Entity Extractor (ANEE) from COLTEC supports time and date [3] entity extraction. No information is disclosed about the underlying techniques except that it is based on linguistic NLP techniques along with statistical methods and claims to be unrestricted by simple look-up tables or rigid rules.

Up to our knowledge, ATEEMA is the first knowledge based temporal entity extractor for the Arabic language and unlike its commercial counterparts [1,2,3], ATEEMA does not use statistical learning.

Comparing results to related work

On a similar genre of evaluation datasets, namely newspapers, ATEEMA working against an Arabic dataset reported higher recall (95%) than [4] (90%) working against an English dataset. ATEEMA reported less precision (84% compared to 90%) resulting in about the same F-score without the need to extensively enumerate all possible temporal expression structures. Note that Arabic is much harder to tackle than English.

A more relevant comparison is to compare ATEEMA to the German temporal entity extractor where morphological analysis is also key [10]. ATEEMA working against an Arabic dataset scored 18% extra recall compared to [10] (77%) working against a German dataset and just 4% less precision (84.2% compared to 88%) difference which amounts to a 7% higher F-score for ATEEMA.

CONCLUSION

We presented ATEEMA, the first open source tool for Arabic temporal entity extraction. ATEEMA extracts temporal morphological features from Arabic text, classifies the features into temporal categories, and passes the features to a finite state transducer (FST). The FST detects temporal entities and uses tolerance parameters to detect the boundaries of the temporal entities. ATEEMA achieved very good results compared to counterpart tools over a set of newspaper text selected arbitrarily.

In the future we will explore extending ATEEMA to normalize the temporal entities. We will also explore applying the same technique to other morphologically rich languages and to Latin languages as well.

REFERENCES

1. Cohen, S.: Entity extraction enables "discovery". Technical report, Basis Technology (2006)
2. Technologies, B.: BBN IdentiFinder Text Suite [Online; accessed 22-April-2010].
3. COLTEC: Anee: Arabic named entity extraction. Technical report, Computer & Language Technology (2007)
4. Koen, D.B., Bender, W.: Time frames: temporal augmentation of the news. IBM Systems Journal **39** (July 2000) 597–616
5. Llidó, D., Berlanga, R., Aramburu, M.J.: Extracting temporal references to assign document event-time periods. In: Proceedings of the 12th International Conference on Database and Expert Systems Applications, Springer Verlag (2001)
6. Setzer, A.: Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study. PhD thesis, University of Sheffield (2001)
7. Setzer, A., Gaizauskas, R.: On the Importance of Annotating Temporal Event-Event Relations in Text. In: Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC2002) Workshop on Annotation Standards for Temporal Information in Natural Language. (2002)
8. Chase, G., Das, J., Davis, S.: Towards developing effective fact extractors. Technical Report DSTO-TR-1729, Australian Goverment, Department of Defence (June 2005)
9. Han, B., Gates, D., Levin, L.: Understanding temporal expressions in emails. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. HLT-NAACL '06, Stroudsburg, PA, USA, Association for Computational Linguistics (2006) 136–143
10. Neumann, G., Backofen, R., Baur, J., Becker, M., Braun, C.: An information extraction core system for real world german text processing. In: Proceedings of 5th conference on Applied natural language processing. ANLC '97, Stroudsburg, PA, USA, Association for Computational Linguistics (1997) 209–216
11. Li, H., Hu, Y., Gao, G., Shnitko, Y., Meyerzon, D., Mowatt, David: Techniques for extracting authorship dates of documents (December 2009)
12. Busemann, S., Declerck, T., Diagne, A.K., Dini, L., Klein, J., Schmeier, S.: Natural language dialogue service for appointment scheduling agents. In: Proceedings of 5th Conference on Applied Natural Language Processing. (1997) 25–32
13. Berlanga, R., Aramburu, M., Barber, F.: Discovering temporal relationships in databases of newspapers. In: Tasks and Methods in Applied Artificial Intelligence. Volume 1416. Springer Berlin / Heidelberg (1998) 36–45
14. Swan, R., Allan, J.: Extracting significant time varying features from text. In: Proceedings of the 8th International conference on Information and knowledge management. CIKM '99, New York, NY, USA, ACM (1999) 38–45

15. Allan, J., Papka, R., Lavrenko, V.: On-line new event detection and tracking. In: ACM SIGIR conference on Research and development in information retrieval. SIGIR '98, ACM (1998) 37–45
16. Traboulsi, H.: Arabic named entity extraction: A local grammar-based approach. In: Computer Science and Information Technology. (October 2009) 139 –143
17. Benajiba, Y., Zitouni, I., Diab, M., Rosso, P.: Arabic named entity recognition: using features extracted from noisy data. In: Proceedings of the ACL 2010 Conference Short Papers. ACLShort '10, Stroudsburg, PA, USA, Association for Computational Linguistics (2010) 281–285
18. Mohri, M.: Local grammar algorithms. In: Inquiries into Words, Constraints, and Contexts. CSLI Publications, Stanford University (2005) 84–93
19. Constant, M.: Grammaires locales pour l'analyse automatique de textes: Méthodes de construction et outils de gestion. PhD thesis, Université de Marne-la-Vallée (september 2003) (252 pp.).
20. Beesley, K.R.: Finite-state morphological analysis and generation of Arabic at xerox research: Status and plans. In: Workshop Proceedings on Arabic Language Processing: Status and Prospects, Toulouse, France (2001) 1–8
21. Gross, M.: Local grammars and their representation by finite automata. In Hoey, M., ed.: Data, Description, Discourse. Papers on the English Language in honour of John McH Sinclair. Harper-Collins (1993) 26–38
22. Chanier, T., Colmerauer, C., Fouqueré, C., Abeillé, A., Picard, F., Zock, M.: Modelling lexical phrases acquisition in l2. Second Language Acquisition Research: The state of the art (1992)
23. Chase, G., Das, J., Davis, S.: Fact extractor system processing engine
24. Han, B., Gates, D., Levin, L.: Understanding temporal expressions in emails. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. HLT-NAACL '06, Stroudsburg, PA, USA, Association for Computational Linguistics (2006) 136–143

FADI ZARAKET
AMERICAN UNIVERSITY OF BEIRUT,
LEBANON
E-MAIL: <FZ11@AUB.EDU.LB>

JAD MAKHLOUTA
AMERICAN UNIVERSITY OF BEIRUT,
LEBANON
E-MAIL: <JEM04@AUB.EDU.LB>