ArbTE: Arabic Textual Entailment

Maytham Alabbas
School of Computer Science
University of Manchester
Manchester, M13 9PL, UK
alabbasm@cs.man.ac.uk

Abstract

The aim of the current work is to see how well existing techniques for textual entailment work when applied to Arabic, and to propose extensions which deal with the specific problems posed by the language. Arabic has a number of characteristics, described below, which make it particularly challenging to determine the relations between sentences. In particular, the lack of diacritics means that determining which sense of a word is intended in a given context is extremely difficult, since many related senses have the same surface form; and the syntactic flexibility of the language, notably the combination of free word-order, pro-drop subjects, verbless sentences, and compound NPs of various kinds, means that it is also extremely difficult to determine the relationships between words.

1 Introduction

The aim of the work described here is to investigate how well existing techniques for 'Recognising Textual Entailment' (RTE: the task of determining, for two sentences *text (T)* and *hypothesis (H)*, whether '...typically, a human reading *T* would infer that *H* is most likely true' (Dagan et al., 2005)). The RTE tasks contrasts with the standard definition of entailment, which states the *T* entails *H* if *H* is true whenever *T* is. The RTE task is in some ways easier than the classical entailment task, and has led to a number of approaches that diverge from the tradition of translating from natural language into 'logical forms' and using standard theorem proving techniques to determine the relationships between these logical forms (Blackburn et al., 2001).

The current system, Arabic Textual Entailment (ArbTE), will investigate the effectiveness of existing TE approaches when they are applied to Modern Standard Arabic (MSA, or Arabic). These approaches have been developed very recently and have largely been applied to English texts. There is very little work on applying textual entailment techniques to Arabic (we have, in fact, so far found no such work), and little evidence that the existing approaches will work for it. The key problem for Arabic is that it is massively more ambiguous than English, for reasons described below, so that many of the existing approaches to textual entailment are likely to be inapplicable.

Lexical ambiguity:

- the Arabic writing system omits characters corresponding to short vowels and other features that distinguish words. This means that written Arabic resembles textese, but the situation is in fact far worse than this analogy suggests, because Arabic has highly productive derivational morphology, which means that a single root form can give rise to numerous derived forms, *most of which are confusable when the short vowels and other markers are omitted*. For instance, the following table shows the Arabic word (علم), which has 7 different reading with diacritics marks.

Arabic	Meaning
عِلْمٌ	knowledge
عَلَمٌ	flag
عَلِمَ	knew
عُلِمَ	is known
عَلَّمَ	taught
عَلِّمْ	teach!
عُلِّمَ	is taught

Table 1: ambiguity caused by the lack of diacritics.

- Arabic also contains numerous clitic items

Proceedings of the Student Research Workshop associated with RANLP 2011, pages 48–53,
Hissar, Bulgaria, 13 September 2011.

(prepositions, pronouns and conjunctions), so that it is often difficult to determine just what items are present in the first place. For example the word والى can be analyzed as والي *'ruler'*, و+الى+ي *'and to me'*, و+ألي *'and I follow'*, و+آل+ي *'and my clan'* or و+آلي *'and automatic'* (Habash et al., 2009). Each of these cases has a different diacritization.

Syntactic ambiguity (Daimi, 2001):

- Arabic has a comparatively free word order, with VSO, VOS, SVO and OVS all being possible orders for the arguments of a transitive verb under appropriate conditions.

- It is a pro-drop language. According to the pro-drop theory (Baptista, 1995), *"a null class (pro) is permitted in a finite clause subject place if the agreement features on the verb are rich enough to enable its content to be recovered"*. The potential absence of a subject is not unique to Arabic, but it causes more problems here than in a number of other languages because Arabic verbs can typically occur either intransitively or transitively. In such cases, it is hard to tell whether a sequence consisting of a verb and a following NP is actually an intransitive use of the verb, with the NP as subject, or a transitive use with a zero subject (or indeed a passive). For example, the Arabic sentence (سأل الطالب سؤالا) has two different meaning, which are *'(He) asked the student a question.'* or *'The student asked a question.'*

- Nouns can be used as adjectives, or as possessive determiners (in so-called 'construct phrases'), with typically little inflectional morphology to mark such uses. Nouns that are being used as possessive determiners, for instance, should be marked as being genitive, but the case markers are almost always omitted in written MSA and hence this clue is unavailable. For instance,the Arabic construct phrase (مفاتيح السيارة) has many comparables in English: *'the keys of the car'* or *'the car's keys'* or *'the car keys'* (Habash, 2010). In this example, the word (السيارة) specifies, defines, limits or explains the particular identity of the word (مفاتيح).

- The copula is omitted in simple positive equational sentences, so that a sequence of a noun and a predicative item (i.e. another noun, an adjective or a PP) may make a sentence. For instance, the Arabic equational sentence has a PP predicate(المعلم في المدرسة) (the-teacher in the-school)*'The teacher (is) in the school.'*

Taken together, these make assigning a structural analysis to a sequence of Arabic forms an extremely difficult task. We have carried out a number of experiments using state-of-the-art taggers (AMIRA 2.0 (Diab, 2009), MADA 3.1 (Habash et al., 2009; Habash, 2010) and a home-grown tagger, MXL, with comparable accuracy) and parsers (notably MALTParser (Nivre et al., 2007) and (McDonald and Pereira, 2006))), using the Penn Arabic Treebank (PATB) (Maamouri and Bies, 2004) as training data. The PATB contains over 5000 phrase-structure trees whereas we want dependency trees. Therefore, we adapted the algorithm described by (Xia and Palmer, 2001), which uses the idea of a *head percolation table* as explained in (Collins, 1997). In the current work, the head percolation table is semi-automatically generated from the PATB by grouping the related tags in one tag and then finding the possible heads for each one, which order manually according to its priority (e.g. in our work *'CONJ'* has high priority for each entry). The outcome of these experiments is that we can achieve around 80% accuracy when assigning unlabelled dependency trees to input texts, and around 70% when we try to attach labels to these trees. These scores are considerably lower than the scores that have been achieved using these parsers on other languages using similar sized training corpora. This reflects the observations above about Arabic, and especially written Arabic: the numerous sources of uncertainty listed above just make parsing difficult.

Given that most TE algorithms operate over parse trees, we have to consider ways of making these algorithms more robust. It is unlikely that we will find ways of parsing Arabic much more accurately–the taggers and parsers we are using are as good as it gets, and the training sets we are using are comparable in size to training sets that have been used for other languages (and experimentation suggests that the data-set/accuracy curve has indeed levelled off by the time we have exhausted the training set). We will be using a

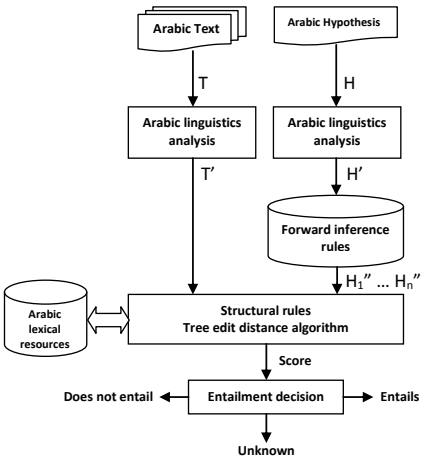

Figure 1: General diagram of ArbTE system.

fairly orthodox TE architecture, as shown in Fig 1. At each stage we will attempt to exploit variations on the standard machinery to help us overcome the extra problems raised by written Arabic.

2 ArbTE

Arabic linguistic analysis: as noted above, we have carried out a number of experiments with state-of-the-art taggers and parsers. These experiments show in particular two main results:

- Combining the output of multiple data-driven dependency parsers can produce more accurate results, even for imperfectly tagged text, than each parser produces by itself for texts with the gold-standard tags. We have recently published preliminary results from these experiments (Alabbas and Ramsay, 2011) and have submitted a more detailed paper to somewhere. Table 2 shows labelled accuracy (LA) of combining MST, MALT1 (*Nivre arc-eager*) and MALT2 (*Stackeager*) compared with retagged (by MADA) and gold-standard corpuses. The first technique (TECHNIQUE1) combines the outputs of the parsers where at least two parsers agree, otherwise the head is taken from MST, whereas the dependency relation is taken from MALT1. The second technique (TECHNIQUE2) combines the outputs of the parsers where MALT1 and MALT2 agree, otherwise the output of MST is taken.

  It is notable that the LA for combining multiple parsers are higher than the LA of the individual parsers for both retagged and gold-standard corpuses.

Best LA for individual parser		TECHNIQUE1	TECHNIQUE2
Retagged	Gold-St.		
0.784	0.793	0.803	0.804

Table 2: LA for combining multiple parsers techniques for MST, MALT1 and MALT2 parsers with voting.

- Combining the output of three different taggers can also produce more accurate results than either parser produces by itself. We describe this result in detail elsewhere. Table 3 shows the results of three taggers when tested on PATB with a coarse-grained tagset.

Taggers accuracy			Combine taggers		
AMIRA	MXL	MADA	P	R	F-score
89.59	95.17	94.05	0.9947	0.83	0.9049

Table 3: Precision (P), recall (R) and F-score for agreement output for three taggers and gold standard.

  It is notable that the precision on the cases where the taggers agree are considerably higher than the accuracy of the individual taggers.

These experiments suggest that obtaining dependency trees from Arabic text is an inherently difficult task. We therefore plan to look more closely at the specific mistakes that the parsers make, in order to identify fragments which are consistently analysed correctly. If we apply our inference rules only to those subtrees which can be trusted then we will improve the accuracy of the inferences that are carried out.

We will also investigate the relative benefits of using labelled and unlabelled dependency trees at this point. Unlabelled trees are, clearly, less reliable as a basis for extracting and applying inference rules. Labelled trees, however, are significantly more difficult for the parsers to get right. We therefore intend to investigate whether it is better to use labelled trees (semantically informative but only found with 70% accuracy) or unlabelled ones (less informative but found with 80% accuracy).

Forward inference rules: we intend to extract transfer-like rules (Hutchins and Somers, 1992) for transforming the parse tree that we extract from the text to other entailed trees, to be used as a set of forward inference rules. The work mentioned above for determining which subtrees can be reliably identified will be exploited here to ensure that we only extract rules from elements of the parse

tree that we trust. We will leave reasoning about open class lexical items to the backward chaining stage that is embodied in the tree matching part of the architecture. As shown in Fig. 1, the forward inference rules are applied before the dependency trees are matched.

Tree edit distance: to match text:hypothesis dependency tree pairs effectively, we will use an extended version of Zhang and Shasha (1989)'s tree edit distance (TED) algorithm, as explained below.

One of the main drawbacks of the TED (Kouylekov, 2006) is that transformation operations (insert, delete and exchange) are applied solely on single nodes and not on subtrees. Heilman and Smith (2010) extended the available operations in standard TED to `INSERT-CHILD`, `INSERT-PARENT`, `DELETE-LEAF`, `DELETE-&- MERGE`, `RELABEL-NODE` and `RELABEL-EDGE`. The authors also identify three new operations, `MOVE-SUBTREE`, which means move a node *X* in a tree *T* to be the last child on the left/right side of a node *Y* in *T* (s.t. *Y* is not a descendant of *X*), `NEW-ROOT` and `MOVE-SIBLING`, to enable succinct edit sequences for complex transformation. This extended set of edit operations allows certain combinations of the basic operations to be treated as single steps, and hence provides shorter (and therefore cheaper) derivations. The fine-grained distinctions between, for instance, different kinds of insertions also make it possible to assign different weights to different variations on the same operation. Nonetheless, these operations continue to operate on individual nodes rather than on subtrees (despite its name, even `MOVE-SUBTREE` appears to be defined as an operation on nodes rather than on subtrees). Therefore, we have extended the basic version of the TED algorithm so that operations that insert/delete/exchange subtrees cost less than the sum of the costs of inserting/deleting/exchanging their parts (e.g. deleting a modifier subtree should be less expensive than the sum of deleting its components individually). This will enable us to find the minimum edit operations to transform one tree to another. Also, this will allow us to be sensitive to the fact that the links in a dependency tree carry linguistic information about relations between complex units, and hence to ensure that when we compare two trees we are paying attention to these relations. For instance, this enables us to be sensitive to the fact that operations involving modifiers, in particular, should be applied to the subtree as a whole rather than to its individual elements. Thus, we transform tree (a) to tree (b) in Fig 2 by deleting *'in the park'* in a single operation, removing the modifier as a whole, rather than three operations removing *'in'*, *'the'* and *'park'* one by one. We have applied the current technique to transform different trees to another trees and obtained encouraging results. So, we just need in this part of the system to find a suitable values of edit operation costs to make matching between two dependency trees more accurate.

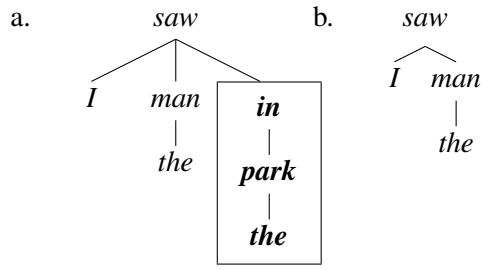

Figure 2: Two dependency trees.

In this part of the system, we also intend to exploit the subset/superset relations encoded by Arabic WordNet (Black et al. (2006)) when exchanging items in a tree. Roughly speaking, if comparing one tree to another requires us to swap two lexical items, we will be happier doing so if the item in the source tree is a hyponym of the one in the target tree. Doing this will allow us to delay making decisions about potentially ambiguous lexical items: it is reasonably safe to assume that if W_1 has a sense which is a hyponym of some sense of W_2 then a sentence involving W_1 will entail a similar sentence involving W_2. This will definitely be quicker, and may be more reliable, than trying to disambiguate the two from first principles and then looking for entailment relationships.

This reflects the widely accepted view that contextual information is the key to lexical disambiguation. Within the RTE task, the text provides the context for disambiguation of the hypothesis, and the hypothesis provides the context for disambiguation of the text. Almost any human reader would, for instance, accept that *T1* entails *H1*, despite the potential ambiguity of the word *'bank'*.

T1: *My money is all tied up at the bank.*
H1: *I cannot easily spend my money.*

We therefore intend to deal with lexical am-

biguity by allowing *T* to entail *H* if there is *any* reading of *T* which entails *any* reading of *H* (this is similar to Hobbs' approach to disambiguation via *'abduction'* (Hobbs et al. (1993); Hobbs (2005))).

Finally, we investigated another Arabic resource to provide us more information about relations between words, *i.e.* Openoffice Arabic dictionary, which contains POS and synonyms for many Arabic words. Currently, we intend to investigate the Microsoft Word Arabic dictionary, which contains a huge amount of Arabic information.

Arabic dataset: in order to train and test our work, we need an appropriate data set. This adds a further level of complexity to our task: there are, to our knowledge, no such data sets available for Arabic, so we have to develop one. We do not want to produce a set of text:hypothesis pairs by hand–partly because doing so is a lengthy and tedious process, but more importantly because hand-coded data sets are liable to embody biases introduced by the developer. If the data set is used for training the system, then the rules that are extracted will be little more than an unfolding of information explicitly supplied by the developers. If it is used for testing then it will only test the examples that the developers have chosen, which are likely to be biased, albeit unwittingly, towards the way they think about the problem.

We therefore need to find some way of extracting such pairs at least semi-automatically. The most promising idea is by posing queries to a search engine and filtering the responses for sentences that do (and don't) entail the query. We are currently building a corpus of text:hypothesis pairs by using headlines from Arabic newspapers and channels TV websites as queries to be input to Google via the standard Google-API, and then selecting the first sentence, which usually represents the most related sentence in the article with the headline, of each of the first N returned pages. This technique produces a large number of potential pairs without any bias in either the texts or the hypotheses. To increase the quality of the sentence's pair that resulted from the query, we add some conditions to filter the results. For instance, the number of common words between both sentences must be less than a specific threshold to avoid having very similar sentences and the length of a headline must be at least more than N words to avoid very small headlines. This technique has different advantages, especially the presence of the same headline in several newspapers but it express in different words for the same day. For instance, one of the CNN headline is *'Berlusconi says he won't seek another term.'* and the related sentence as shown in Table 4. We make the same query but for another website, such as BBC and Reuters and the results as shown in Table 4. Therefore, we can swap between a headline of one newspaper with related sentences from another to increase the quality of the sentences's pair. We have tested this technique on different languages, such as Arabic, English, Spanish, German, Turkish, Bulgarian and French. We carried out a series of informal experiment with native speakers. The results were encouraging, but the nature of the experiments mean that they are not robust.

The Arabic articles that are returned by this process typically contain very long sentences (upwards of 100 words), where only a small part has a direct relationship to the query. This is typical of Arabic text, which is often written with very little punctuation, with elements of the text linked by conjunctions rather than being broken into implicit segments by punctuation marks such as full stops and question marks. We have carried out some initial experiments aimed at segmenting the large parse trees that we obtain for such sentences into smaller linked elements. This is more reliable than simply segmenting the surface strings at conjunctions, since many conjunctions link non-sentential structures, and are therefore not sensible places to break the text.

These pairs still have to be marked-up by human annotators, but at least the process of collecting them is as nearly bias-free as possible. Therefore, to annotated our dataset, we are currently developing an online annotation system, which will be soon available for the annotators. The system normally presents the annotator with sentences that they have not yet seen, but there is also an option to revisit previously annotated examples. Finally, each pair of sentences must be annotated by three different users before we get the result of annotation which represents the agreement of at least two users. The system will be flexible with the number of annotators for each pair of sentences and maybe increase it to five when we need that.

3 Conclusions and Future Work

We have outlined a number of approaches to the task of adapting existing TE algorithms for work-

Website	Headline (Hypothesis)	Related sentence (Text)	Results
CNN	Berlusconi says he won't seek another term.	Italian Prime Minister Silvio Berlusconi said Friday he will not run again when his term expires in 2013.	Entails
BBC	Silvio Berlusconi vows not to run for new term in 2013.	Italian Prime Minister Silvio Berlusconi has confirmed that he will not run for office again when his current term expires in 2013.	Entails
Reuters	Berlusconi says he will not seek new term.	Italian Prime Minister Silvio Berlusconi declared on Friday he would not run again when his term expires in 2013.	Entails

Table 4: Some English text:hypothesis pairs.

ing with a language where we are faced with an exceptional level of lexical and structural ambiguity. As we previously mentioned, there are different options for each stage, so we will try to test different combinations between the system components to find the best structure of ArbTE system.

Also we speculate that further work by marking the 'polarity' of subtrees in the dependency trees obtained by the parser(s) and making rules sensitive to the polarity of the items they are being applied to would further improve ArbTE results. This will make the use of TED as a way of determining consequence relations more reliable for all languages, not just Arabic: the fact that *T2* entails *H2*, whereas *T3* does not entail *H3*, arises from the fact that *'doubt'* reverses the polarity of its sentential complement. Systems that pay no attention to polarity will inevitably make mistakes, and we intend to adapt the TED algorithm so that it pays attention to this issue.

T2: *I believe that a woman did it.*
H2: *I believe that a human being did it.*

T3: *I doubt that a woman being did it.*
H3: *I doubt that a human did it.*

References

Alabbas, M. and Ramsay, A. (2011). Evaluation of dependency parsers for long arabic sentences. In *Proceeding of International Conference on Semantic Technology and Information Retrieval (STAIR'11)*, pages 243–248. IEEE.

Baptista, M. (1995). On the nature of pro-drop in Capeverdean Creole. Technical report, Harvard Working Papers in Linguistics, 5: 3-17.

Black, W., Elkateb, S., Rodriguez, H., and Alkhalifa, M. (2006). Introducing the Arabic WordNet project. In *Proceedings of the Third International WordNet Conference (GWC-06)*.

Blackburn, P., Bos, J., Kohlhase, M., and De Nivelle, H. (2001). Inference and computational semantics. *Studies in linguistics and philosophy*, pages 11–28.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid.

Dagan, I., Magnini, B., and Glickman, O. (2005). The PASCAL recognising textual entailment challenge. In *Proceedings of Pascal Challenge Workshop on Recognizing Textual Entailment*.

Daimi, K. (2001). Identifying Syntactic Ambiguities in Single-Parse Arabic Sentence. *Computers and the Humanities*, 35(3):333–349.

Diab, M. (2009). Second generation tools (amira 2.0): Fast and robust tokenization, pos tagging, and base phrase chunking. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, pages 285–288.

Habash, N. (2010). *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Habash, N., Rambow, O., and R., R. (2009). Mada+tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In Choukri, K. and Maegaard, B., editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt. The MEDAR Consortium.

Heilman, M. and Smith, N. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Hobbs, J. R. (2005). *The handbook of pragmatics*, chapter Abduction in Natural Language Understading, pages 724–740. Blackwell Publishing.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, 63:69–142.

Hutchins, W. J. and Somers, H. L. (1992). *An introduction to machine translation*. Academic Press.

Kouylekov, M. (2006). *Recognizing textual entailment with tree edit distance: Application to question answering and information extraction*. PhD thesis.

Maamouri, M. and Bies, A. (2004). Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of COLING*, pages 2–9.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6, pages 81–88.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Xia, F. and Palmer, M. (2001). Converting dependency structures to phrase structures. In *1st Human Language Technology Conference (HLT-2001)*, pages 1–5, San Diego.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.