Improved POS-Tagging for Arabic by Combining Diverse Taggers

Maytham Alabbas, Allan Ramsay

Improved POS-tagging for Arabic by combining diverse taggers

Maytham Alabbas and Allan Ramsay

School of Computer Science,
Manchester University,
Manchester, UK
{alabbasm,ramsay}@cs.man.ac.uk

Abstract. A number of POS-taggers for Arabic have been presented in the literature. These taggers are not in general 100% accurate, and any errors in tagging are likely to lead to errors in the next step of natural language processing. The current work shows an investigation of how the best taggers available today can be improved by combining them. Experimental results show that a very simple approach to combining taggers can lead to significant improvements over the best individual tagger.

Key words: Combining systems, Arabic tagging.

1 Introduction

The process of assigning a correct POS tag (i.e. noun, verb, adverb or others) to each word of a sentence is called part-of-speech(POS) tagging. This process is considered an essential step for most natural language applications. In general, however, POS-taggers make mistakes, and since tagging is the first step in most natural language processing (NLP) systems these mistakes will lead to problems in all subsequent stages of analysis. It is thus important to obtain the highest possible accuracy at this initial stage of processing. One popular technique for improving tagging accuracy is *tagger combination*. This approach involves combining different taggers to exploit the unique properties of each tagger and reduce some of the random errors. This technique has been applied for different languages such as English [1], Swedish [2],Telugu [3], Italian [4] and Polish [5] and the results were encouraging, but has not to our knowledge been applied for Arabic. The current work is step forward in this regard. We evaluate different techniques for combining POS-taggers for Arabic.

The key problem for Arabic is that it is more ambiguous than many other languages (e.g. English) for many reasons. Firstly, Arabic is written without diacritics (short vowels and a range of other phonological effects), often leading to multiple ambiguities [6]. This is particularly problematic because the diacritics are often the only difference between different words (especially derived forms) and between inflected forms of the same word. This matter makes analysis of the language morphologically very challenging. This is because a certain lemma

(or lexeme) in Arabic can be interpreted in various ways. Hence, a single word can have various senses, where determining the sense is based on the context in which the word is used. Furthermore, a noun in Arabic can be diacritised in three different ways for the nominative, accusative and genitive cases, which can be even more ambiguous at the structural or grammatical level.

In addition, Arabic is highly syntactically flexible [7]:

- It has a comparatively free word order, where sentence components can be exchanged without affecting the core meaning. This results in structural ambiguity, with each morphological analysis having more than a single meaning. So, besides the regular sentence of verb, subject and object (VSO), Arabic allows other potential surface forms such as SVO and VOS constructions. The potential of allowing variations on the canonical order leads to a large amount of ambiguity.
- Furthermore, Arabic (like Spanish, Italian and Japanese) is a pro-drop language [8]. The pro-drop can lead to structural ambiguity by leaving any syntactic parser with the challenge to determine if there is a dropped pronoun or not in the subject position, which is made worse by the fact that lots of Arabic verbs can have both transitive and intransitive forms, and further that it is generally impossible to tell the difference between active and passive forms by inspecting the surface form. In case that just one noun phrase (NP) follows one of these verbs, the ambiguity appears. In contrast with English because its canonical order is SVO, losing the subject (S) does not cause confusion about the object (O).
- In addition the copula is omitted in simple positive equational sentences (the sentences that did not contain verb), so that a sequence of a noun and a predicative item (i.e. another noun, an adjective or a prepositional phrase (PP)) may make a sentence.
- Finally, Arabic nouns can be linked together without any overt marker, whereas two English nouns are joined together by different markers, such as the suffix "-'s" on possessing noun, a possessive phrase "of" or a pronoun as "his/her". In the construct phrase the first noun must be indefinite solely (which can be in any case: nominative, genitive or accusative). The second noun may be either definite or indefinite (which is always in genitive case).

While these are strictly speaking syntactic issues, they have serious consequences for tagging, since they mean that the local context often fails to supply constraints on tags–to take a simple example, if you do not know whether the item following a verb is its subject or its object then you cannot use the proximity of the verb to determine the case marker.

2 The taggers

We are interested in improving tagging by combine different taggers. We have carried out a number of experiments with state-of-the-art taggers (AMIRA 2.0 [9], MADA 3.1 [10] and a home-grown tagger, MXL [11], with comparable accuracy), using the Penn Arabic Treebank (PATB) [12] as a *gold-standard* corpus.

Gold-standard tags

We used PATB Part 1 v3.0 as a resource for our experiments. The words in the PATB are already tagged. This provides us with a benchmark for tagger evaluation. Even PATB tagging is not guaranteed to be 100% accurate, but it nonetheless provides as good a reference set as can be found.

The PATB uses a very fine-grained set of tags, which carry a great deal of syntactically relevant information (particularly case-marking). This tagset contains 305 tags, with for instance 47 tags for different kinds of verb and 44 for different kinds of noun. We carried out our experiments with a variant of this original tagset, and also with a coarser-grained version obtained by deleting details such as case- and agreement-markers.

AMIRA

The first tagger we use is AMIRA. This tagger is reported to achieve around 97% accuracy on its target tagset, which is about as good as any reported system.

Using AMIRA, however, highlights one of the problems that arise when we try to compare its retagged corpus with other corpora. The PATB is tagged, but with different tags from the ones used by AMIRA.[1] In order to compare AMIRA tags with other corpora tags, we will have to translate between the two tagsets.

This is a non-trivial task. The two tagsets have different numbers of tags (e.g. AMIRA has 130 fine-grained tags compared with 305 in PATB), and more importantly they make different kinds of distinctions. The AMIRA tagset, for instance, uses one tag (`RP`) to cover a range of particles which are subdivided into eight subclasses (`EMPH_PART`, `EXCEPT_PART`, `FOCUS_PART`, `INTERROG_PART`, `RC_PART`, `NEG_PART`, `PART`, `VERB_PART`) in the PATB; and it uses several tags to describe different kinds of verbs (`VB, VBG, VBD, VBN, VBP`) where the PATB just uses three (`IV, PV, CV`).

In order to overcome these problems, we use transformation-based retagging (TBR) [13, 14] to recover from the mismatches between the two tagsets. TBR collects statistics about the local context in which erroneous tags have been assigned, and attempts to find rules based on this information to apply *after* the original tagger has been run. This technique will produce a small improvement in the performance of almost any tagger. Typically, taggers that achieve scores in the mid 90s are boosted by 2-3%–the lower the original accuracy, the greater the typical improvement. When we used it for comparing the original tags produced by AMIRA and the gold tags the score improved from around 89% to just over 95%. Some of this improvement arises simply from rules that spot that the two tagsets use different names for the same things (e.g. that things that are called `JJ` are called `ADJ` in the PATB) but some of it comes from learning how to split coarse-grained AMIRA tags into fine-grained PATB ones.

[1] We used the Extended Reduced Tag Set (ERTS) setting for AMIRA and then removed inflectional markers. This produced a set of tags that is very similar to the 25 tags in the Bies/RTS tagset, but with distinctions between nouns, adjectives and cardinal numbers.

There is a further problem with using AMIRA. The fact that Arabic allows a range of items to be cliticised (conjunctions, prepositions, pronouns) makes it difficult even to tokenise text reliably. This means that not only does AMIRA sometimes assign different tags from the PATB, it sometimes even splits the text into different numbers of tokens (i.e. AMIRA's tokeniser segments the text differently from the way that it is segmented in the PATB–AMIRA's tokeniser gives us around 97% agreement with gold-standard tokenisation).

In order to see whether this was the cause of the difference, we constructed a version of the corpus where we replaced PATB tags by a coarse copy of the AMIRA tags, using hand-coded substitution rules, and then replaced these by fine-grained AMIRA tags where the substituted tags were compatible with tags actually assigned by AMIRA. Thus if the PATB assigned a word the tag `ADJ` we replaced it by the AMIRA tag `JJ`. We then inspected the tags assigned by AMIRA itself: if the tag assigned to this word was one of AMIRA's fine-grained adjective tags, e.g. `JJR`, then we used this instead. If, on the other hand, the hand-coded replacement for the PATB tag was incompatible with the one assigned by AMIRA then we retained the hand-coded one. This gave us a version of the treebank that had the same number of tokens as the original PATB, with as many items as possible given the tags assigned by AMIRA and the others given hand-coded AMIRA equivalents of the original PATB tags.

MADA

MADA [15] uses a slightly extended version of the PATB Part 1 v3.0 tagset, with some extra classification of nouns. The fine-grained MADA retagged version contains 352 tags compared to 305 tags in the PATB corpus. Fortunately the MADA tags are a strict superset of the standard PATB set, and hence can be reduced to either the standard fine-grained version or our coarse version by omitting the extra classification of nouns, so we do not have the same problems using MADA with the PATB as we have with AMIRA.

We also applied TBR to the output of MADA, because although we were not faced with mapping incompatible tagsets in the same way as with AMIRA, using TBR nearly always provides a small improvement, amounting in this case to an increase from 94% to 96.7%. We applied the same technique that we used in AMIRA to get a MADA version of the corpus that has the same size as the gold-standard one because MADA's tokeniser also has small (1%) variance from the gold-standard.

Maximum-likelihood tagger

We also use an in-house maximum-likelihood tagger, which we will refer to as MXL. We have described this tagger in detail in [11]: we will simply outline the basic principles that it is based on and note its accuracy here.

MXL operates in two stages, as follows:

- In the first stage we use two simple kinds of statistic: (i) the conditional likelihood that a word which starts with the same three letters or ends with the same three letters as the one we are trying to tag has a given tag, and (ii) the transition probabilities between tags. We use a weighted combination of these to produce a maximum-likelihood guess at the current tag. This process produces about 95.2% accuracy.
- We then use TBR to refine the original set of hypotheses, leading to a final accuracy of 95.6%.

The advantage of this tagger is that because it was trained on the PATB, the tags it uses are exactly the PATB tags and the tokenisation is exactly the PATB tokenisation. We therefore do not need to overcome problems associated with mismatches between tag sets.

3 Improving POS-tagging accuracy

In the current work, we are interested in both fine-grained and coarse-grained tagsets. We therefore collapsed the original fine-grained set, by deleting inflectional markers (case, number, gender), to the coarse-grained set show in Table 1 (e.g. PATB has 305 fine-grained tags which become 39 coarse-grained tags, for instance, the fine-grained tags `NOUN+CASE_DEF_ACC`, `NOUN+CASE_DEF_GEN`, `NOUN+CASE_DEF_GEN+POSS_PRON_3MS`, `NOUN+CASE_DEF_NOM` are grouped to `NOUN`).

ABBREV	DET+NOUN_PROP	LATIN	PUNC
ADJ	DET+NUM	NEG_PART	PV
ADV	EMPH_PART	NOUN	PVSUFF_DO
CONJ	EXCEPT_PART	NOUN_PROP	RC_PART
CV	FOCUS_PART	NO_FUNC	REL_ADV
CVSUFF_DO	FUT+IV	NUM	REL_PRON
DEM_PRON	INTERJ	PART	SUB
DET	INTERROG_PART	POSS_PRON	SUB_CONJ
DET+ADJ	IV	PREP	VERB_PART
DET+NOUN	IVSUFF_DO	PRON	

Table 1. Coarse-grained tagset

Table 2 summarises the fine-grained and coarse-grained tags for PATB corpus and the three retagged corpora by the taggers: AMIRA, MADA and MXL. Table 3 summarises the accuracy of the three taggers on our gold standard set using the built-in tagsets and coarse versions of each, and shows the improvements that are obtained in each case by applying TBR. This provides a reference point: the best of the three taggers is MADA, which achieves 96.7% on the coarse-grained version of its built-in tagset if we also apply TBR and 93.6% on the fine-grained version, again after applying TBR. The goal of the current paper is

to see whether we can improve on this by utilising the other taggers, despite the fact that their individual performance is worse.

POS	PATB/MXL	AMIRA	MADA
Coarse-grained	39	29	56
Fine-grained	305	130	351

Table 2. Coarse-grained and Fine-grained tags numbers, single tagger

POS	TBR	AMIRA	MXL	MADA
Coarse-grained	×	89.6%	95.2%	94.1%
	√	95.3%	95.6%	**96.7%**
Fine-grained	×	84.3%	89.7%	91.7%
	√	88.8%	91.2%	**93.6%**

Table 3. Tagger accuracies in isolation, with and without TBR

The first observation is that when the taggers agree on how to tag a given item they are more likely to be right than when they disagree. This is fairly obvious–if you have a set of taggers which assign different tags to an item then at least one of them must be wrong! Table 4 substantiates this observation–each column shows the precision (P) and recall (R) for a particular pair of taggers simply taking cases where they agree and leaving words on which they disagree untagged. Thus the combination of MXL and MADA achieves a precision of 99.5% on the coarse-grained tagset and 99% on the fine-grained one. Table 5 shows what happens when we combine all three taggers, either taking the majority view when at least two of them agree or demanding that all three agree. In the latter case we obtain a precision of 99.9% for the coarse-grained tagset and 99.2% for the fine-grained one.

Metrics	POS	TBR	MXL-AMIRA	MXL-MADA	MADA-AMIRA
P	Coarse-grained	×	97.7%	99.3%	96.3%
		√	98.8%	**99.5%**	**99.5%**
	Fine-grained	×	92.9%	98.9%	94.7%
		√	94.3%	**99%**	95.8%
R	Coarse-grained	×	84.3%	90.1%	86.1%
		√	94.1%	94.6%	91.9%
	Fine-grained	×	79.8%	83.3%	84.9%
		√	81.6%	86.4%	85.6%
F-SCORE	Coarse-grained	×	0.905	0.945	0.909
		√	0.964	0.97	0.955
	Fine-grained	×	0.858	0.904	0.895
		√	0.875	0.923	0.904

Table 4. Precision (P) and Recall (R) and F-score for combinations of pairs of taggers, with and without TBR

Condition	POS	TBR	P	R	F-SCORE
At least two agree	Coarse-grained	×	95.1%	93.5%	0.943
		√	98.4%	97.8%	0.981
	Fine-grained	×	90.7%	87.8%	0.892
		√	92.8%	90.5%	0.916
All three agree	Coarse-grained	×	99.5%	83%	0.905
		√	**99.9**%	90.9%	0.952
	Fine-grained	×	99.1%	77.8%	0.871
		√	**99.2**%	79.8%	0.885

Table 5. Precision (P) and Recall (R) and F-score for combinations of three taggers, with and without TBR

The cost, of course, is that the recall goes down, because there are places where they disagree, and in these cases no tag is assigned. What should we do in such cases?

3.1 Backoff strategies

Our first proposal was to take the majority view when at least two of the taggers agreed, and to backoff to one or other when they there was no common view [16]. The results of this are shown in Table 6 where column one shows the result of backing off to MXL when there is no majority view and column two shows the result of backing off to MADA. The results for the coarse-grained tagset are markedly better than for any of the taggers individually, though there is only a very slight improvement over the original MADA scores for the fine-grained set. Interestingly, the best results are obtained by backing off to MXL rather than to MADA (98.2% vs. 97.9%, despite the fact that MADAs individual performance is better than MADA's (96.7% vs. 95.6%) (the results in this table and in Table 8 are for accuracy rather than precision or recall).

POS	TBR	Otherwise, MXL tag	Otherwise, MADA tag
Coarse-grained	×	94.9%	93.8%
	√	**98.2%**	97.9%
Fine-grained	×	89.6%	88.8%
	√	91.8%	91.5%

Table 6. Backoff to MXL or MADA when there is no agreement

The fact that backing-off to either MADA or MXL improved the performance suggested that it was worth investigating other backoff strategies. If majority vote + backoff to an arbitrary tagger is better than any single tagger in isolation, then perhaps there is something we can backoff to which will do even better.

We return to the observation that in cases where all three taggers disagree, at most one of them can be right. Given that they each employ different information

about the material being tagged, it is likely that they are systematically prone to different kinds of errors.

We therefore collected statistics about the *kinds* of things they each get right. How likely, for instance, is MXL to be right when it assigns the tag NEG_PART? Table 7 shows an extract of this data, showing the likelihood that each tagger is right for a given assigned tag, e.g. for the given instance of the word غير *gyr*[2] "other than" the correct tag was NEG_PART (as in gold standard): MADA suggested NOUN, MXL suggested NEG_PART, AMIRA suggested RP. Because MXL is right 98.2% of the time when it suggests NEG_PART, whereas MADA is right 97.9% of the time when it suggests NOUN and AMIRA is right only 8.1% of the time when it suggests RP, NEG_PART was chosen. For إلى *<lA* "to", MADA's suggestion of EXCEPT_PART was accepted because MADA is right 100% of the time when it suggests EXCEPT_PART, which is better than the reliability of either of the other suggestions.

Word	Gold standard	MADA	MXL	AMIRA	TAG
gyr	NEG_PART	NOUN (97.9)	NEG_PART (98.2)	RP (8.1)	NEG_PART
<lA	EXCEPT_PART	EXCEPT_PART (100.0)	SUB_CONJ (96.5)	RP (7.9)	EXCEPT_PART
...	...	...	...	...	...

Table 7. Confidence levels for individual tags

Using this strategy for choosing what to do when all three taggers make different suggestions produces the results in Table 8.[3] The 'default' column reports the results when we simply chose the most confident proposal, whereas for 'backoff unless two agree' we took the majority verdict if two of the taggers agreed and the most confident if all three gave different results.

POS	TBR	default	backoff unless two agree
Coarse-grained	×	97.3%	95.7%
	√	**99.5**%	99.2%
Fine-grained	×	95.6%	93.2%
	√	**96.0%**	94.5%

Table 8. Backoff to most confident tagger

The results in Table 8 are both surprising and compelling. Simply taking the most confident of the three taggers produces 99.5% accuracy for the coarse-grained set, which is going to be hard to beat by much, and even for the fine-grained set it produces 96%. This improves over taking the majority verdict when at least two of the contributing taggers agree and backing off to the most

[2] The transcription of Arabic examples follows Buckwalter's system for transcribing Arabic symbols. Available at: http://www.qamus.org/transliteration.htm.

[3] These results were obtained by 10-fold cross validation.

confident one where there is no agreement,[4] and it beats each of the individual taggers by a fairly wide margin–the original error by MADA of 3.3% for the coarse-grained set has been reduced to 0.5%, a nearly sevenfold reduction.

4 Conclusions

We have shown a rather simple mechanism for combining taggers which can provide considerable improvements in accuracy. If you measure the likelihood that a tagger is right when it suggests a particular tag, and then take the suggestion with highest score in each case then you can decrease the error rate substantially. The key is that the different taggers tend to make different systematic mistakes. The accuracy statistics capture these systematic mistakes, so a low score is likely to reflect a case where the tagger is making one of its characteristic errors, and in such cases we take the output of one of the others. This simple approach outperforms strategies involving more subtle ways of combining the individual taggers, e.g. by taking the majority preference in cases where one exists. This is likely to be because two of the taggers (MADA and AMIRA) use very similar information, and hence where they make systematic mistakes they are likely to make the *same* systematic mistakes. In such cases they will tend to agree, and hence would outvote MXL if allowed the majority view to win, as suggested by the precision results in Table 5. The simple mechanism we have used provides a built-in resilience against this tendency.

In every case, including TBR makes a useful contribution. Again, TBR is most effective when the base tagger makes systematic errors. The version of TBR that we are using includes extra templates looking at the first three and last three letters of words in addition to the standard word-based templates. These extra templates pay attention to prefixes and suffixes, which carry much more information than is the case for English (where TBR has been most extensively applied).

The bottom line is that for the tagset in Table 1 we can obtain 99.5% accuracy when tagging freely occurring Arabic. It is going to be hard to improve substantially on this score. Given that Marton et al. [17] have argued that coarse-grained tagsets are actually more useful than fine-grained ones for parsing, which is the usual next step in the chain, we are fairly satisfied with this result.

Acknowledgments. We would like to thank Dr. Yasser Sabtan (Al-Azhar University, Eygpt) for important suggestions and for helpful discussions. Maytham Alabbas owes his deepest gratitude to Iraqi Ministry of Higher Education and Scientific Research for financial support in his PhD study. Allan Ramsay's contribution to this work was supported in part by Qatar National Research Foundation grant NPRP 09 - 046 - 6 - 001.

[4] Note that taking the majority verdict when all three agree and backing off to the most confident when there is not complete unanimity is **exactly** the same as simply taking the most confident one from the outset.

References

1. Brill, E., Wu, J.: Classifier combination for improved lexical disambiguation. In: Proceedings of the 17th international conference on Computational linguistics-Volume 1, Association for Computational Linguistics (1998) 191–195
2. Sjöbergh, J.: Combining pos-taggers for improved accuracy on swedish text. In: Proceedings of NoDaLiDa 2003. (2003)
3. RamaSree, R., Kusuma Kumari, P.: Combining pos taggers for improved accuracy to create telugu annotated texts for information retrieval. Dept. of Telugu Studies, Tirupathi, India (2007)
4. Søgaard, A.: Ensemble-based pos tagging of italian. IAAI-EVALITA, Reggio Emilia, Italy (2009)
5. Śniatowski, T., Piasecki, M.: Combining polish morphosyntactic taggers. Security and Intelligent Information Systems (2012) 359–369
6. Nelken, R., Shieber, S.: Arabic diacritization using weighted finite-state transducers. In: Proceedings of the Workshop on Computational Approaches to Semitic Languages at 43rd Meeting of the Association for Computational Linguistics (ACL05). (2005) 79–86
7. Daimi, K.: Identifying syntactic ambiguities in single-parse arabic sentence. Computers and the Humanities **35**(3) (August 2001) 333–349
8. Farghaly, A.: Subject pronoun deletion rule. In: Proceedings of the 2nd English Language Symposium on Discourse Analysis (LSDA82). (1982) 110–117
9. Diab, M.: 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, Cairo, Eygpt, The MEDAR Consortium (April 2009) 285–288
10. Habash, N.: Introduction to Arabic Natural Language Processing. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2010)
11. Ramsay, A., Sabtan, Y.: Bootstrapping a lexicon-free tagger for arabic. In: Proceedings of the 9th Conference on Language Engineering ESOLEC2009, Cairo, Egypt (December 2009) 202–215
12. Maamouri, M., Bies, A.: Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In: Proceedings of COLING. (2004) 2–9
13. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics **23(4)** (1995) 543–565
14. Lager, T.: μ-tbl lite: a small, extendible transformation-based learner. In: Proceedings of the 9th European Conference on Computational Linguistics (EACL-99), Bergen, Association for Computational Linguistics (1999) 279–280
15. Habash, N., Rambow, O., Roth, R.: MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In: Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, The MEDAR Consortium (2009)
16. Zeman, D., Žabokrtskỳ, Z.: Improving parsing accuracy by combining diverse dependency parsers. In: Proceedings of the Ninth International Workshop on Parsing Technology, Association for Computational Linguistics (2005) 171–178
17. Marton, Y., Habash, N., Rambow, O.: Improving arabic dependency parsing with lexical and inflectional morphological features. In: Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, Association for Computational Linguistics (2010) 13–21