Combining strategies for tagging and parsing Arabic

Maytham Alabbas
Department of Computer Science
University of Basrah
Basrah, Iraq
maytham.alabbas@gmail.com

Allan Ramsay
School of Computer Science
University of Manchester
Manchester M13 9PL, UK
Allan.Ramsay@manchester.ac.uk

We describe a simple method for combining taggers which produces substantially better performance than any of the contributing tools. The method is very simple, but it leads to considerable improvements in performance: given three taggers for Arabic whose individual accuracies range from 0.956 to 0.967, the combined tagger scores 0.995–a sevenfold reduction in the error rate when compared to the best of the contributing tools.

Given the effectiveness of this approach to combining taggers, we have investigated its applicability to parsing. For parsing, it seems better to take pairs of similar parsers and back off to a third if they disagree.

1 Introduction

If you have several systems that perform the same task, it seems reasonable to suppose that you can obtain better performance by using some judicious combination of them than can be obtained by any of them in isolation. A large number of combining strategies have been proposed, with majority voting being particularly popular (Stefano et al., 2002). We have investigated a range of such strategies for combining taggers and parsers for Arabic: the best strategy we have found for tagging involves asking each of the contributing taggers how confident it is, and accepting the answer given by the most confident one. We hypothesise that the reason for the effectiveness of this strategy for tagging arises from the fact that the contributing taggers work in essentially different ways (different training data, different underlying algorithms), and hence if they make systematic mistakes these will tend to be different. This means, in turn, that the places where they *don't* make mistakes will be different.

This strategy is less effective for parsing. We have tried combining two members of the MALTParser family (Nivre et al., 2006; Nivre et al., 2007; Nivre et al., 2010) with MSTParser (McDonald et al., 2006a; McDonald et al., 2006b). The best strategy here seems to be to accept the output of the two versions of MALTParser when they agree, but to switch to MSTParser if the MALTParser versions disagree. It may be that this is because the MALTParser versions are very similar, so that when they disagree this suggests that there is something anomalous about the input text, and that neither of them can be trusted at this point.

2 Tagging

We present a very simple strategy for combining part-of-speech (POS) taggers which leads to substantial improvements in accuracy. A number of combination strategies have been proposed in the literature (Zeman and Žabokrtskỳ, 2005). In experiments with combining three Arabic taggers (AMIRA (Diab, 2009), MADA (Habash et al., 2009) and a simple affix-based maximum-likelihood Arabic tagger (MXL) (Ramsay and Sabtan, 2009)) the current strategy significantly outperformed voting-based strategies.

We used the Penn Arabic Treebank (PATB) Part 1 v3.0 as a resource for our experiments. The words in the PATB are already tagged, which thus provides us with a widely-accepted Gold standard. Even PATB tagging is not guaranteed to be 100% accurate, but it nonetheless provides as good a reference set as can be found.[1]

The PATB uses the tags provided by the Buckwalter morphological analyser (Buckwalter, 2004; Buckwalter, 2007), which carry a great deal

[1]The PATB is the largest easily available tagged Arabic corpus, with about 165K words in the section we are using. Thus for each fold of our 10-fold testing regime we are training on 150K words and testing on 15K, which should be enough to provide robust results.

Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 73–77,
October 25, 2014, Doha, Qatar.

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. The very fine distinctions between different kinds of nouns and verbs (e.g. between subject and object case nouns) in the absence of visible markers make this an extremely difficult tagset to work with. It is in general virtually impossible to decide the case of an Arabic noun until its overall syntactic role is determined, and it is similarly difficult to decide the form of a verb until the overall syntactic structure of the sentence is determined. For this reason taggers often work with a coarser set of tags, of which the 'Bies tagset' (Maamouri and Bies, 2004) is widely used (see for instance the Stanford Arabic parser (Green and Manning, 2010)). We carried out our experiments with a variant of the original fine-grained tagset, and also with a variant of the coarser-grained Bies set obtained by deleting details such as case- and agreement-markers. We carried out two sets of experiments, with a coarse-grained set of tags (a superset of the Bies tagset with 39 tags, shown in Figure 1) and the original fine-grained one with 305 tags.

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

Table 1: Coarse-grained tagset

The accuracy of a tagger clearly depends on the granularity of the tagset: the contributing taggers produced scores from 0.955 to 0.967 on the coarse-grained tagset, and from 0.888 to 0.936 on the fine-grained one. We applied transformation-based retagging (TBR) (Brill, 1995; Lager, 1999) to the output of the basic taggers, which produced a small improvement in the results for MADA and MXL and a more substantial improvement for AMIRA. Table 2 shows the performance of the three taggers using the two tagsets with and without TBR. The improvement obtained by using

POS	TBR	AMIRA	MXL	MADA
Coarse	×	0.896	0.952	0.941
	√	0.953	0.956	**0.967**
Fine	×	0.843	0.897	0.917
	√	0.888	0.912	**0.936**

Table 2: Tagger accuracies in isolation, with and without TBR

TBR for AMIRA arises largely from the fact that in some cases AMIRA uses tags similar to those used in the English Penn Treebank rather than the ones in the the tags in the PATB, e.g. JJ for adjectives where the PATB uses ADJ. TBR provides a simple and reliable mechanism for discovering and patching systematic renamings of this kind, and hence is extremely useful when working with different tagsets. A significant component of the remaining errors produced by AMIRA arise because AMIRA has a much coarser classification of particles than the classification provided by the Buckwalter tagset. Since AMIRA assigns the same tag to a variety of different particles, TBR cannot easily recover the correct fine-grained tags, and hence AMIRA makes a substantial number of errors on these items.

The key to the proposed combining strategy is that each of the contributing taggers is likely to make systematic mistakes; and that if they are based on different principles they are likely to make *different* systematic mistakes. If we classify the mistakes that a tagger makes, we should be able to avoid believing it in cases where it is likely to be wrong. So long as the taggers are based on sufficiently different principles, they should be wrong in different places.

We therefore collected confusion matrices for each of the individual taggers showing how likely they were to be right for each category of item–how likely, for instance, was MADA to be right when it proposed to tag some item as a noun (very likely–accuracy of MADA when it proposes NN is 0.98), how likely was AMIRA to be right when it proposed the tag RP (very unlikely–accuracy of 0.08 in this case)? Given these tables, we simply took the tagger whose prediction was most likely to be right.[2]

Table 3 shows an excerpt from the output of the

[2] All the tagging results reported below were obtained by using 10-fold cross validation, i.e. carrying out 10 experiments each of which involved removing 10% of the data for testing and training on the remaining 90%.

Word	Gold standard	MADA	MXL	AMIRA	TAG
…	…	…	…	…	…
gyr	NEG_PART	NOUN (0.979)	NEG_PART (0.982)	RP (0.081)	NEG_PART
<lA	EXCEPT_PART	EXCEPT_PART (1.00)	SUB_CONJ (0.965)	RP (0.790)	EXCEPT_PART
…	…	…	…	…	…

Table 3: Confidence levels for individual tags

three individual taggers looking at a string containing the two words *gyr* and *<lA*, with the tags annotated with the accuracy of each tagger on the given tag, e.g. in this sequence MADA has tagged *gyr* as a noun, and MXL has tagged it as a negative particle and AMIRA has tagged it as RP; and when MADA suggests NOUN as the tag it is right 97.9% of the time, whereas when MXL suggests NEG_PART it is right 98.2% of the time and AMIRA is right just 8.1% of the time when it suggests RP. It is important to note that the tags are assigned to words in context, but the confidence levels are calculated across the entire training data. The fact that MADA is right 97.9% of the time when it assigns the tag NOUN is not restricted to the word *gyr*, and certainly not to this occurrence of this word.

We compared the results of this simple strategy, which is similar to a strategy proposed for image classification by Woods at el. (1997), with a strategy proposed by (2005), in which you accept the majority view if at least two of the taggers agree, and you back off to one of them if they all disagree, and with a variation on that where you accept the majority view if two agree and back off to the most confident if they all disagree. The results are given in Table 4.

All four strategies produce an improvement over the individual taggers. The fact that majority voting works better when backing off to MXL than to MADA, despite the fact that MADA works better in isolation, is thought-provoking. It seems likely to be that this arises from the fact that MADA and AMIRA are based on similar principles, and hence are likely to agree *even when they are wrong*. This hypothesis suggested that looking at the likely accuracy of each tagger on each case might be a good backoff strategy. It turns out that it is not just a good backoff strategy, as shown in the third column of Table 4: it is even better when used as the main strategy (column 5). The differences between columns 4 and 5 are not huge,[3] but that should not be too surprising, since these two strategies will agree in every case where all three of the contributing taggers agree, so the only place where these two will disagree is when one of the taggers disagrees with the others *and* the isolated tagger is more confident than either of the others.

The idea reported here is very simple, but it is also very effective. We have reduced the error in tagging with fairly coarse-grained tags to 0.05%, and we have also produced a substantial improvement for the fine grained tags, from 0.936 for the best of the individual taggers to 0.96 for the combination.

3 Parsing

Given the success of the approach outlined above for tagging, it seemed worth investigating whether the same idea could be applied to parsing. We therefore tried using it with a combination of dependency parsers, for which we used MSTParser (McDonald et al., 2006a; McDonald et al., 2006b) and two variants from the MALTParser family (Nivre et al., 2006; Nivre et al., 2007; Nivre et al., 2010), namely Nivre arc-eager, which we will refer to as MALTParser$_1$, and stack-eager, which we will refer to as MALTParser$_2$. The results in Table 5 include (i) the three parsers in isolation; (ii) a strategy in which we select a pair and trust their proposals wherever they agree, and back-off

[3] In terms of error rate the difference looks more substantial, since the error rate, 0.005, for column 5 for the fine-grained set is 62.5% of that for column 4, 0.008; and for the coarse-grained set the error rate for column 5, 0.04, is 73% of that for column 4, 0.055

Tagset	Majority voting (back off to MXL)	Majority voting (back off to MADA)	Majority voting (back off to AMIRA)	Majority voting (most confident)	Just most confident
Coarse-grained	0.982	0.979	0.975	0.992	**0.995**
Fine-grained	0.918	0.915	0.906	0.945	**0.96**

Table 4: Modified majority voting vs proposed strategy

	Parser	LA
(i)	MSTParser	0.816
	MALTParser_1	0.797
	MALTParser_2	0.796
(ii)	Use MSTParser & MALTParser_1 if they agree, backoff to MALTParser_2	0.838
	Use MSTParser & MALTParser_2 if they agree, backoff to MALTParser_2	0.837
	Use MALTParser_1 & MALTParser_2 if they agree, backoff to MSTParser	**0.848**
(iii)	Use MSTParse & MALTParser_1 if they agree, backoff to most confident	0.801
	Use MSTParser & MALTParser_2 if they agree, backoff to most confident	0.799
	Use MALTParser_1& MALTParser_2 if they agree, backoff to most confident	0.814
(iv)	If at least two agree use their proposal, backoff to most confident	0.819
	If all three agree use their proposal, backoff to most confident	0.797
	Most confident parser only	0.789

Table 5: Labelled accuracy (LA) for various combinations of MSTParser, MALTParser_1 and MALTParser_2 five fold cross-validation with 4000 training sentences and 1000 testing

to the other one when they do not; (iii) a strategy in which we select a pair and trust them whenever they agree and backoff to the parser which is most confident (which may be one of these or may be the other one) when they do not; (iv) strategies where we either just use the most confident one, or where we take either a unanimous vote or a majority vote and backoff to the most confident one if this is inconclusive. All these experiments were carried using fivefold cross-validation over a set of 5000 sentences from the PATB (i.e. each fold has 4000 sentences for training and 1000 for testing).

These results indicate that for parsing, simply relying on the parser which is most likely to be right when choosing the head for a specific dependent in isolation does not produce the best overall result, and indeed does not even surpass the individual parsers in isolation. For these experiments, the best results were obtained by asking a predefined pair of parsers whether they agree on the head for a given item, and backing off to the other one when they do not. This fits with Henderson and Brill (2000)'s observations about a similar strategy for dependency parsing for English. It seems likely that the problem with relying on the most confident parser for each individual daughter-head relation is that this will tend to ignore the big picture, so that a collection of relations that are individually plausible, but which do not add up to a coherent overall analysis, will be picked.

4 Conclusions

It seems that the success of the proposed method for tagging depends crucially on having taggers that exploit different principles, since under those circumstances the *systematic* errors that the different taggers make will be different; and on the fact that POS tags can be assigned largely independently (though of course each of the individual taggers makes use of information about the local context, and in particular about the tags that have been assigned to neighbouring items). The reason why simply taking the most likely proposals in isolation is ineffective when parsing may be that global constraints such as Henderson and Brill's 'no crossing brackets' requirement are likely to be violated. Interestingly, the most effective of our strategies for combining parsers takes two that use the same learning algorithm and same feature sets but different parsing strategies (MALTParser_1 and MALTParser_2), and relies on them when they agree; and backs off to MSTParser, which exploits fundamentally different machinery, when these two disagree. In other words, it makes use of two parsers that depend on very similar underlying principles, and hence are likely to make the same systematic errors, and backs off to one that exploits different principles when they disagree.

We have not carried out a parallel set of experiments on taggers for languages other than Arabic because we do not have access to taggers where we have reason to believe that the underlying principles are different for anything other than Arabic. In situations where three (or more) distinct approaches to a problem of this kind are available,

it seems at least worthwhile investigating whether the proposed method of combination will work.

Acknowledgements

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 partially supported by the Qatar National Research Fund (grant NPRP 09-046-6-001).

References

E Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, 23(4):543–565.

T Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data Consortium.

T Buckwalter. 2007. Issues in Arabic morphological analysis. *ARabic computational morphology*, pages 23–41.

M. Diab. 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, Cairo, Eygpt, April. The MEDAR Consortium.

S Green and C D Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 394–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

N. Habash, O. Rambow, and R. Roth. 2009. 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.

J C Henderson and E Brill. 2000. Exploiting diversity in natural language processing: Combining parsers. *CoRR*, cs.CL/0006003.

T Lager. 1999. μ-tbl lite: a small, extendible transformation-based learner. In *Proceedings of the 9th European Conference on Computational Linguistics (EACL-99)*, pages 279–280, Bergen. Association for Computational Linguistics.

M Maamouri and A Bies. 2004. Developing an Arabic treebank: methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, pages 2–9, Geneva.

R McDonald, K Lerman, and F Pereira. 2006a. Multilingual dependency parsing with a two-stage discriminative parser. In *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, New York.

R McDonald, K Lerman, and F Pereira. 2006b. Multilingual dependency parsing with a two-stage discriminative parser. In *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, New York.

J. Nivre, J. Hall, and J. Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 2216–2219.

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

J Nivre, L Rimell, R McDonald, and C Gómez-Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 833–841, Beijing.

A. Ramsay and Y. Sabtan. 2009. Bootstrapping a lexicon-free tagger for Arabic. In *Proceedings of the 9th Conference on Language Engineering*, pages 202–215, Cairo, Egypt, December.

Claudio De Stefano, Antonio Della Cioppa, and Angelo Marcelli. 2002. An adaptive weighted majority vote rule for combining multiple classifiers. In *ICPR (2)*, pages 192–195.

Kevin Woods, W. Philip Kegelmeyer, Jr., and Kevin Bowyer. 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):405–410, April.

D. Zeman and Z. Žabokrtskỳ. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 171–178. Association for Computational Linguistics.