This is a repository copy of *Combinatory hybrid elementary analysis of text (CHEAT)*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/82290/

Proceedings Paper:
Atwell, ES (2007) Combinatory hybrid elementary analysis of text (CHEAT). In: Proceedings of CL'2007 Corpus Linguistics Conference. CL'2007 Corpus Linguistics Conference, 27-30 July 2007, CL'2007 Corpus Linguistics Conference. UCREL, Lancaster University .

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Combinatory Hybrid Elementary Analysis of Text

Eric Atwell
School of Computing
University of Leeds
Leeds LS2 9JT, England
eric@comp.leeds.ac.uk

Andrew Roberts
Pearson Longman
Edinburgh Gate
Harlow CM20 2JE, England
andrew.roberts@pearson.com

Abstract

We propose the CHEAT approach to the MorphoChallenge contest: Combinatory Hybrid Elementary Analysis of Text. The idea is: acquire results from a number of other candidate systems; CHEAT will read in the output files of each of the other systems, and then line-by-line select the "majority vote" analysis - the analysis which most systems have gone for. If there is a tie, take the result produced by the system with the highest F-measure; if the other systems' output files are ordered best-first, then this is achieved by simply taking the first of he tied results. To justify our approach, we need to show that this really is unsupervised learning, as defined on the MorphoChallenge website; arguably the CHEAT approach involves super-sized unsupervised learning, as it combines three different layers of unsupervised learning.

1 Our guiding principle: get others to do the work

The reuse of existing components is an established principle in Software Engineering; it is quicker, easier, and overall better to engineer a system using components built by others, than to develop a complex system ourselves. This principle is behind our CHEAT approach to the MorphoChallenge task: to avoid doing work ourselves, we got others to do most of the work, and then copied their results. However, straightforward copying of another entrant's results might be considered unacceptable (perhaps even cheating), so we had to do something a bit smarter. Students generally know that blatant copying of another's work is condemned as plagiarism, and can be detected by text analysis software, eg (Atwell et al 2003); but some students may try to get away with less blatant "smart" copying (Medori et al 2002). We procured results from several candidate systems, and then developed a program to allow "voting" on the analysis of each word: for each word, examine the set of candidate analyses; where all systems were in agreement, the common analysis is copied; but where contributing systems disagree on the analysis, take the "majority vote", the analysis given by most systems. If there is a tie, take the result produced by the system with the highest F-measure; if the other systems' output files are ordered best-first, then this is achieved by simply taking the first of he tied results.

Procuring results from several candidate systems was a challenge by itself, given that entrants were to submit results direct to the MorphoChallenge organizers. These results would not be "on show" until the Workshop, well after the deadline for us to submit our own entry. Our ideal solution was to develop a set of intelligent agents, each of which would learn to develop and submit an entry for the MorphoChallenge contest; we could then use the results of these intelligent agents. However, we did not have sufficient time or AI expertise to build software agents capable of this advanced learning. Fortunately, Eric Atwell had to teach an MSc course in the School of Computing at Leeds University, on Computational Modeling. For assessment, students had to undertake a computational modeling exercise; as the course and the MorphoChallenge contest ran concurrently, this presented the opportunity to set the MorphoChallenge as a student coursework exercise, and require the students to submit their entries to their lecturer (for internal assessment) at the same time as submitting to the organizers.

2 But is this really "unsupervised learning"?

According to the MorphoChallenge website FAQ, "unsupervised learning" means that "*…the program cannot be explicitly given a training file containing "example answers", and nor can example answers be hard-coded into the program.*" We must admit that we originally formulated this definition (to suit our approach) and proposed this to the organizers, who accepted and published it. The presence or absence of "example answers" distinguishes supervised from unsupervised learning: in supervised learning, the system is shown the correct analysis or answer for at least some input words (but not all, otherwise this would not be Machine Learning but dictionary-lookup!) Our CHEAT program is *not* shown definitely correct answers for any word, as it is given not one but several files: although each results file constitutes a set of *candidate/possible* answers, they may not be correct answers, and there is no way of knowing which is correct – the voting system is designed for disagreements between candidates, who cannot all be correct. So, strictly speaking, our CHEAT system **is** an unsupervised learning system.
In fact, there are three cascading layers of unsupervised learning in the overall process, so we call this "Super-sized unsupervised learning":

2.1 Unsupervised learning by autonomous agents: students

Of course, Leeds University MSc students are far more intelligent than any software agent; but they still needed to learn how to tackle the MorphoChallenge task. The Computational Modeling class included students on Cognitive Systems, BioInformatics, GeoInformatics, and Health Informatics programmes, so the students had little or no previous knowledge of morphological analysis or machine learning systems development. Their approach to learning was unsupervised, or at least semi-supervised: Eric Atwell presented lectures on machine learning and linguistic principles underlying morphological analysis, and formulated a coursework specification www.comp.leeds.ac.uk/cmd/assessment.htm and marking scheme involving entry to the contest; but then the students were left to learn for themselves how to develop algorithms and systems. They were **not** explicitly given "example answers" – in this case, example algorithm or code to perform unsupervised learning of morphological analysis. And "example answers" were defnitely **not** hard-coded into the students – in this case, this would mean downloading algorithm or code direct into their brains, something even Leeds University teaching methods can't achieve. The students learning about morphological analysis and machine learning constituted the first phase in the CHEAT cascade: a set of autonomous unsupervised learning agents.

2.2 Unsupervised learning by student programs

The students worked in pairs, each pair designing and implementing a program to perform unsupervised learning of morphological analysis. So, these programs constitute the second phase of the CHEAT cascade: a set of independent unsupervised learning programs, each producing a candidate set of morphological analyses of the contest word-files. Detailed descriptions of the student programs are available in the reports submitted by the students alongside the results files. For our purposes, we treated each student program as a "black box" – all we needed were the results files.

2.3 Unsupervised learning by cheat.py

The third phase in the CHEAT cascade is a simple program to read in the candidate results file, choose the most popular analysis of each word, and output this as the CHEAT result. In the spirit of the CHEAT approach, to avoid doing work by getting others to do it, Eric Atwell tried to avoid having to write this program himself, by asking Andy Roberts to do it – hence our collaboration on this entry. Eric Atwell wrote a basic Python version which worked in theory but not in practice; Andy Roberts supplied a much improved version which coped with the unexpected problems.

3 cheat.py

Python has straightforward yet elegant features for reading, processing and writing text, and "mainstream" syntax similar to Java or C++, so seemed the obvious choice for implementation language. Eric Atwell's first python program is so simple that it should be self-explanatory. The version below reads in 7 candidate results files for the English dataset, ordered by their F-measure scores: hr.txt, cd.txt, b.txt, hz.txt, km.txt. aa.txt. mw.txt. Letters hr, cd etc are initials of the student surnames; b.txt shows one student worked alone.

```
# CHEAT: Combinatory Hybrid
# Elementary Analysis of Text
# Eric Atwell's first PYTHON
# program, 15/01/2006
# first open each result-file,
# open a.txt to write CHEAT result
aa=open('aa.txt','r')
b=open('b.txt','r')
cd=open('cd.txt','r')
hr=open('hr.txt','r')
hz=open('hz.txt','r')
km=open('km.txt','r')
mw=open('mw.txt','r')
a=open('a.txt','w')
# a.txt will be the result file
n=6
# n+1: the no of files to combine
# loop: read each result-file-line
# in array Results[0..n]
# ordered by F-measure score: hr
# was the best, mw was the worst
for Results in
zip(hr,cd,b,hz,km,aa,mw):
# setup array Votes[0..n]
# all values initially 1
  Votes=[1 for x in range(n+1)]
# Votes=[1,1,1,1,1,1] might be
# simpler, but less showoffy...
  for r in range(1,n):
   for t in range(r):
     if Results[r]==Results[t]:
       Votes[t]= Votes[t] + 1
# set Votes[N] to number of copies

# next find the top scoring result
  topscore=1
  topresult=1
  for r in range(n):
   if Votes[r] > topscore:
     topresult=r

# Finally output Results[topresult]
  a.write(Results[topresult])

# after end of loop, close all
# files to terminate cleanly
aa.close()
b.close()
cd.close()
hr.close()
hz.close()
km.close()
mw.close()
a.close()
```

This appeared to work with test samples. However, it assumes the input files are all valid, correctly formatted and containing the analyzed words in the same sequence as the given input. Unexpectedly, this turned out not to be the case with all the student results files. Some of the student programs tried to read in the entire word-file, process and segment words in a program buffer, and then print out the buffer contents in alphabetically sorted order. Unfortunately, the details of sort-ordering are different in some packages or programming languages; in particular, Capital and lower-case letters can be sorted together or separately, and non-alphabetic characters (common in Turkish and Finnish datasets, and found in some loanwords even in the English dataset) may also vary in rank-order. The result was that several student results files did not match the ordering of the input dataset; so the simple cheat.py above was not comparing segmentations of the same words.

4 cheat2.py

Andrew Roberts came to the rescue with a much improved comparison algorithm, which read all the input files into memory, ensured comparisons of “like with like”, then wrote out the majority-vote analysis. Unfortunately the program is too long to include in this paper, but we can assure the reader that it is much more robust, elegant and exception-proof than the first version of cheat.py.

5 Results

We evaluated the final cheat2.py results files using the evaluation.perl program provided by the MorphoChallenge organizers, which compared the results files against small Gold Standard samples of words which we were assured had “correct” segmentation. We then compared the evaluation.perl scores for CHEAT output against the scores for the contributing systems’ outputs: 7 systems for English, but only 4 systems managed to cope with the much larger Turkish and Finnish datasets.

```
Evaluation of segmentation
in English results file
against gold standard
segmentation in file
"goldstdsample.eng":
Number of words in gold
standard: 532 (type count)
Number of words in data set:
167377 (type count)
Morpheme boundary detections
statistics:
```

System	F-measure %	Precision %	Recall %
CHEAT	59.19	60.71	57.74
hr	54.89	53.87	55.94
cd	51.83	48.06	56.23
B	49.10	46.90	51.52
hz	38.62	37.55	39.75
km	36.96	33.04	41.95
aa	30.55	23.17	44.83
mw	28.48	22.01	40.35

```
Evaluation of segmentation
in Turkish results file
against gold standard
segmentation in file
"goldstdsample.tur":
Number of words in gold
standard: 774 (type count)
Number of words in data set:
582935 (type count)
Morpheme boundary detections
statistics:
```

System	F-measure %	Precision %	Recall %
CHEAT	56.63	62.05	52.08
cd	55.94	59.39	52.87
hr	44.38	59.46	35.39
B	42.05	54.51	34.23
mw	40.44	37.40	44.02

```
Evaluation of segmentation
in Finnish results file
against gold standard
segmentation in file
"goldstdsample.fin":
Number of words in gold
standard: 660 (type count)
Number of words in data set:
1636336 (type count)
Morpheme boundary detections
statistics:
```

System	F-measure %	Precision %	Recall %
CHEAT	60.26	66.10	55.37
cd	60.18	64.97	56.04
hr	43.46	67.18	32.12
B	38.69	56.95	32.90
mw	28.30	24.18	34.12

We also downloaded the Morfessor system developed by the MorphoChallenge organizers, as advertised on the website (!), and used it to analyse the English, Turkish and Finnish datasets. We then repeated the previous experiments, this time including the Morfessor output as an additional candidate file. We were very surprised to find that the resulting F-measure, Precision and Recall for CHEAT remained unchanged from the values in the tables above – the Morfessor output seemed to have no influence whatsoever on the votes! We then realized that the version of Morfessor freely available via the contest website had apparently been modified so that none of the words from the three Gold Standard samples are included in the evaluation. Thus Morfessor appeared to yield Precision and Recall scores of 0/0 or 100%, but this presumably did not mean other words in the output were all correct.

```
Evaluation of segmentation
in file "m.txt" against
gold standard segmentation
in file "goldstdsample.fin":
Number of words in gold
standard: 660 (type count)
Number of words in data set:
1636336 (type count)
Number of words evaluated: 0
(0.00% of all words in data
set)
Morpheme boundary detections
statistics:
F-measure:  100.00%
Precision:  100.00%
Recall:     100.00%
```

6 Conclusions

For all three languages (English, Turkish, Finnish), our CHEAT system scored a higher F-measure than any of the contributing systems. It also achieved better Precision and Recall scores, with a couple of exceptions: *cd* had a slightly higher Recall for Turkish and Finnish (but not English, and *cd* had a lower Precision and F-measure for all three languages), and *hr* had a higher Precision for Finnish (but lower Precision and F-measure). Combinatory Hybrid Elementary Analysis of Text is a valid approach to Unsupervised Learning of morphological analysis.

We thought we had dreamt up the CHEAT approach as a clever scam to avoid work, get students to do the hard work while letting us come up with a winning system. However, an anonymous reviewer of our draft paper pointed out that the CHEAT approached seemed similar to, or even a copy of, an approach already known in the Machine Learning literature: a committee of unsupervised learners. It transpires that we have inadvertently adopted an unsupervised learning approach to machine learning research: we de veloped the CHEAT algorithm without use of training material such as the background literature, eg (Banko and Brill 2001), adding a fourth layer to the super-sized unsupervised learning model.

Yet another thing we learnt from searching in http://scholar.google.com for research papers on "committee of unsupervised learners" is that "unsupervised learning" is a recognized term in Education research, referring to student learning with minimal explicit direction from teachers, eg (Pursula 2004). It turns out that super-sized unsupervised learning is not only a valid (and hopefully interesting) approach to Machine Learning for the MorphoChallenge task, but also a valid approach to Student Learning. Student feedback suggests that the MSc students relished the challenge of participating in an international research contest, and this inspired many of them to produce outstanding coursework … which made the CHEAT results even better!

Reference

Eric Atwell, Paul Gent, Julia Medori, Clive Souter. 2003. Detecting student copying in a corpus of science laboratory reports. In: Archer, D, Rayson, P, Wilson, A & McEnery, T (editors) *Proceedings of CL2003: International Conference on Corpus Linguistics*, pp. 48-53 Lancaster University.

Michele Banko, Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics ACL '01* 26-33.

Julia Medori, Eric Atwell, Paul Gent, Ckive Souter. 2002. Customising a copying-identifier for biomedical science student reports: comparing simple and smart analyses. In: O'Neill, M, Sutcliffe, R, Ryan, C, Eaton, M, & Griffith, N (editors) *Artificial Intelligence and Cognitive Science, Proceedings of AICS02*, pp. 228-233 Springer-Verlag.

Matti Purusla. 2004. An integrated model for lifelong learning. In *Proceedings of the 9th World Conference on Continuing Engineering Education*, Tokyo.