

Speech and Natural Language Processing Project Report

Extending the functionality of OCR++

Pourush Sood, Avinab Saha, Vishesh Agarwal and Sohom Chakraborty

Mentor: Mayank Singh

Introduction:

OCR++, an open-source framework designed for a variety of information extraction tasks from scholarly articles including metadata (title, author names, affiliation and e-mail), structure (section headings and body text, table and figure headings, URLs and footnotes) and bibliography (citation instances and references). More information about the software can be looked up [here](#).

Objectives:

Identifying and tagging extracted references from research papers

Currently, OCR++ can extract entire references from research papers as a whole string.

We wished to extend it to identify and tagging of key information parts in a reference like author-name, name of referenced paper etc. The crucial idea for this part is that we want to avoid any and all external dependencies (like the internet, or even the use of a dictionary). Our current plan is to build a machine learning system for the task.

Extracting key phrases and definitions in the research paper

Inspiration from hyperlinks on wikipedia. OCR++ will understand key phrases from papers and will highlight them. We need to identify n gram type of keyphrases, and print them. The key issue for this part is that even though most keyphrases in the research papers are specially stylised (**bolded** or *italicised* or underlined), we are currently not able to use that to our advantage. The problem lies in the fact that PDFs don't store these formatting separately, and thus, this information is not available when we convert the PDF to XML for data processing.

Reference Extraction:

Part 1:

The dataset had over 800 references in a text file with references from scientific papers hosted in arxiv.org. The dataset can be found [here](#).

A few references from the dataset is shown below.

```
<author> A. Aiken and E. L. Wimmers. </author> <title> Type inclusion constraints and type inference. </title> <booktitle> In Proceedings of the ACM SIGPLAN Conference on Functional Programming Languages Computer Architecture, </booktitle> <pages> pages 31-41, </pages> <date> 1993. </date> <br>
<author> A. Avritzer and B. Larson. </author> <title> Load testing software using deterministic state testing. </title> <booktitle> Proceedings of the 1993 International Symposium on Software Testing Analysis ISSA. </booktitle> <publisher> ACM Press, </publisher> <date> June 1993, </date> <pages> pages 82-88. </pages> <br>
<author> A. C. Yao. </author> <title> The entropic limitations on VLSI computations. </title> <booktitle> In Proceedings of the 13 th Annual ACM Symposium on Theory of Computing, </booktitle> <location> Milwaukee, Wisconsin, </location> <pages> pages 308-311, </pages> <date> May 1981. </date> <br>
<author> A. C. Yao. </author> <title> The entropic limitations on VLSI computations. </title> <booktitle> In Proceedings of the 13 th Annual ACM Symposium on Theory of Computing, </booktitle> <location> Milwaukee, Wisconsin, </location> <pages> pages 308-311, </pages> <date> May 1981. </date> <br>
```

As a part of data pre-processing, we extracted every word in the dataset and tagged the word a particular tag as given the dataset like Author, Title, Booktitle, Location, Pages, Date. The full prepared dataset, was divided in 4:1 ratio for training and testing purposes.

A part of the final dataset is shown below.

```
1 A. author
2 Cau, author
3 R. author
4 Kuiper, author
5 and author
6 W.-P. author
7 de author
8 Roever. author
9 Proc. booktitle
10 5th. booktitle
11 BCS-FACS booktitle
12 Refinement booktitle
13 Workshop, booktitle
14 1992. date
15 In editor
16 C. editor
17 B. editor
18 Jones, editor
19 R. editor
20 C. editor
21 Shaw, editor
22 and editor
23 T. editor
24 Denvir, editor
25 editors, editor
26 Formalising title
27 Dijkstra's title
28 development title
29 strategy title
30 within title
31 Stark's title
32 formalism. title
```

We trained a CRF Model on the training dataset and tested it on the test data set. The results were pretty much average. We varied the dependencies of a particular word on the previous and next words. The maximum accuracy obtained was around **78%**.

Part 2:

As a next step towards increasing the accuracy of our model, we included the following features while training:

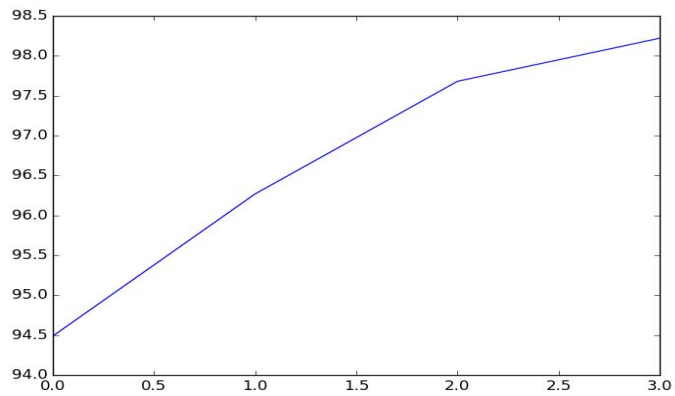
- Including POS tags of words.
- Increasing Feature Space to include more number of previous words.
- Increasing Weights on immediately preceding words.

A snapshot from the dataset is added below.

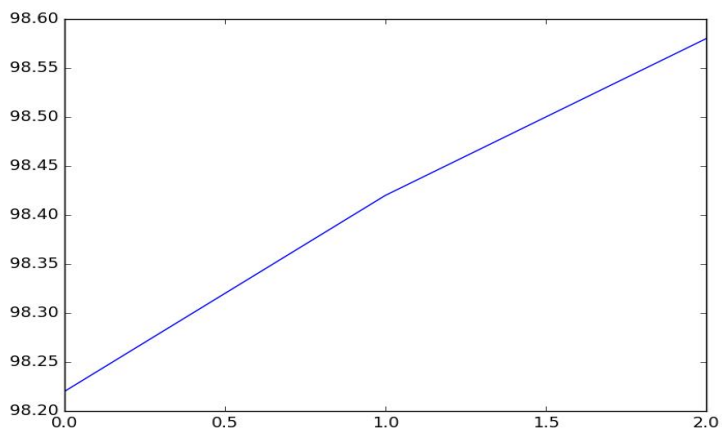
```
133 In IN booktitle
134 Proceedings NNS booktitle
135 of IN booktitle
136 the DT booktitle
137 24th CD booktitle
138 Annual JJ booktitle
139 ACM NN booktitle
140 Symposium NN booktitle
141 on IN booktitle
142 Principles NNS booktitle
143 of IN booktitle
144 Programming VBG booktitle
145 Languages, NNS booktitle
```

We achieved an accuracy of **95.3%** after the following features were taken into account.

Graphical Illustrations:

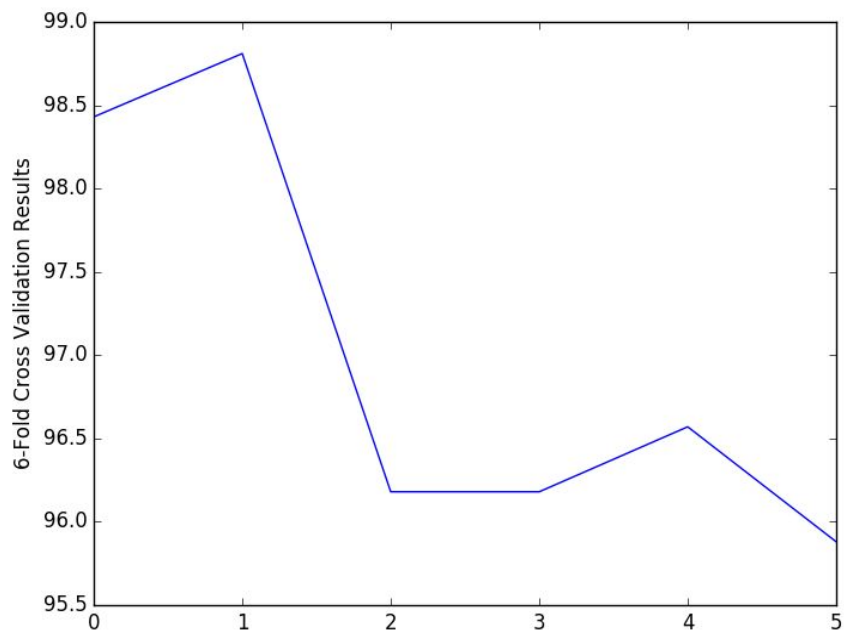


Variation of accuracy of the model with increase in the word window



Variation of accuracy of the model with increase in the weights of POS tags words

To validate our results, we used **6 fold cross validation** to verify our results.



Variation of accuracy for each of the 6 cluster's used in cross validation

Accuracy for Each Tag:

| Tag | Accuracy(%) |
|--------------|-------------|
| Author | 99.28 |
| Booktitle | 91.16 |
| Date | 99.37 |
| Journal Name | 78.88 |
| Location | 89.85 |
| Pages | 91.75 |
| Publisher | 62.96 |
| Volume | 87.32 |

| | |
|-------|-------|
| Title | 97.61 |
|-------|-------|

Part 3:

The dataset had over one lakh files with references from papers hosted in arxiv.org. The corresponding pdfs were downloaded from arxiv.org using a web crawler designed by us. Due to download restrictions of arxiv website against bots accessing their data, we downloaded around 500 PDFs. Of the 500 PDFs downloaded a little over 390 PDFs could be used on OCR++ to extract stylistic features from the Reference section of the documents. The data set was divided into parts for Training and Test purpose in 4:1 ratio. So, Training was done on around 310 PDFs and test on 80 PDFs. For each work token, we mapped it to Author/ Paper Title/ Journal Details using data in our references file(.bbl format).

Features used in Reference Extraction:

For each token extracted, we used the following features:

- POS Tags (From NLTK package)
- Font Name (From OCR++)
- Font Size (From OCR++)
- Italics/ Non Italics (From OCR++)
- Bold/ Non Bold (From OCR++)

After training the model it was found that, Font Name and Font Size did not affect the performance of the model and hence were removed from the set of features.

A snapshot of the training and test data is shown below:

```

252783 IROS NN NimbusRomNo9L 0 0 Journal
252784 2010. CD nimbusromno9l 0 0 Journal
252785 [11] NUM nimbusromno9l 0 0 ReferenceNumber
252786 J. NNP nimbusromno9l 0 0 Author
252787 Redmon NN nimbusromno9l 0 0 Author
252788 S. NNP nimbusromno9l 0 0 Author
252789 Divvala NN nimbusromno9l 0 0 Author
252790 R. NN nimbusromno9l 0 0 Author
252791 Girshick NNP nimbusromno9l 0 0 Author
252792 and CC nimbusromno9l 0 0 Paper
252793 A. DT nimbusromno9l 0 0 Author
252794 Farhadi. NNP nimbusromno9l 0 0 Author
252795 You PRP nimbusromno9l 0 0 Paper
252796 only RB nimbusromno9l 0 0 Paper
252797 look NN nimbusromno9l 0 0 Paper
252798 once: RB nimbusromno9l 0 0 Paper
252799 Unified, RB nimbusromno9l 0 0 paper
252800 real-time NN nimbusromno9l 0 0 Paper
252801 object NN nimbusromno9l 0 0 Paper
252802 detection. NN nimbusromno9l 0 0 Paper
252803 In IN nimbusromno9l 0 0 Journal
252804 CVPR NN nimbusromno9l 0 0 Journal
252805 2016. CD nimbusromno9l 0 0 Journal
252806 [12] NUM nimbusromno9l 0 0 ReferenceNumber
252807 J. NNP nimbusromno9l 0 0 Author
252808 Redmon NN nimbusromno9l 0 0 Author
252809 and CC nimbusromno9l 0 0 Paper
252810 A. DT nimbusromno9l 0 0 Author
252811 Farhadi. NNP nimbusromno9l 0 0 Author
252812 YOLO9000: NNP nimbusromno9l 0 0 Paper
252813 better RBR nimbusromno9l 0 0 Paper

```

Performance Metrics:

We used a CRF algorithm to train our model. The externally developed CRF++ package was used for training and test purposes.

Features used: Only POS Tags

- Training time : 136.44 seconds
- Accuracy: 87.04%

Features used: POS Tags and Stylistic Features

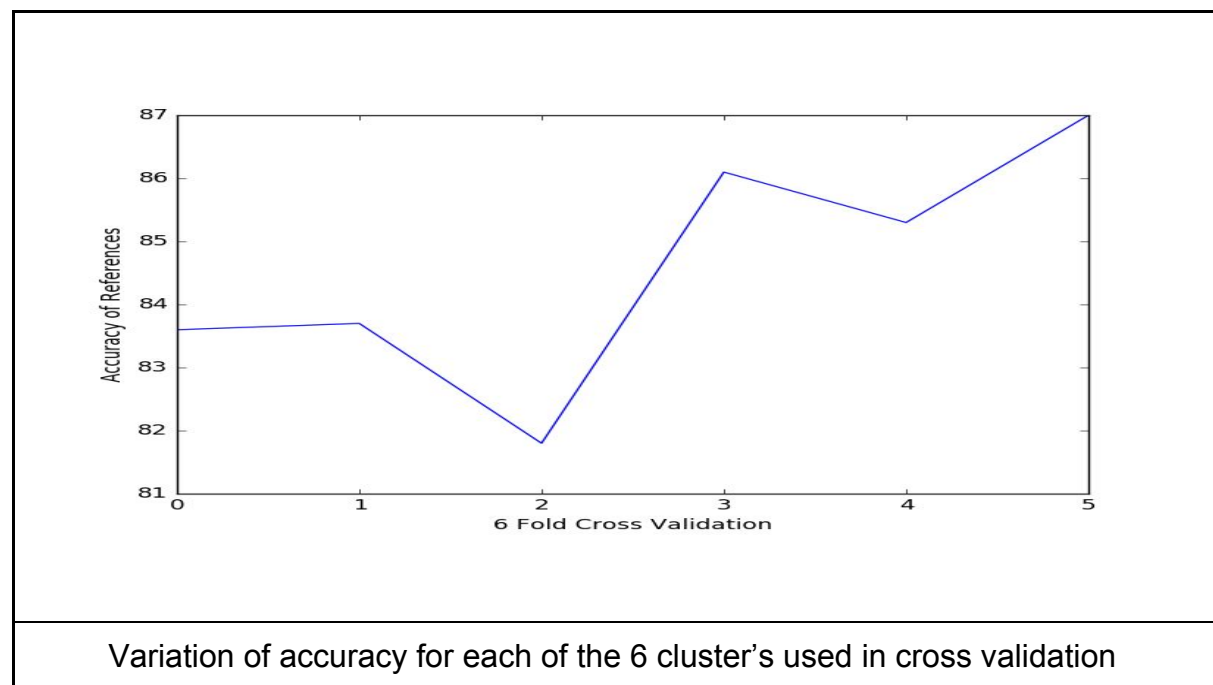
- Training time : 310.18 seconds
- Accuracy: 87.17%

Features used: POS Tags (Double Weightage) and Stylistic Features

- Training time : 315.22 seconds
- Accuracy: 87.97%

It was observed the PDFs had very less stylistic features in them. It is expected that training on PDFs with more stylistic features can significantly help in increasing accuracy.

To validate our results, we used **6 fold cross validation** to verify our results.



Accuracy For Each Sub Category:

Author Name Accuracy = 96.5%

Paper Title Accuracy = 85.4%

Journal/ Conference Details Accuracy: 83.2%

It was observed that the there was Paper Title accuracy was low. When we had a look at the Predicted versus Actual Tags, we found that the low accuracy in those two fields was mainly because of the occurrence of the commonly occurring words like 'and' , 'of' , 'is' etc. that are generally used in the Paper Title as well as the Conference Name. As an alternative, we implemented some rule based heuristics for some commonly occurring words with word length less than 4 and the results were encouraging.

Extracting key phrases from the research paper

Dataset Used:

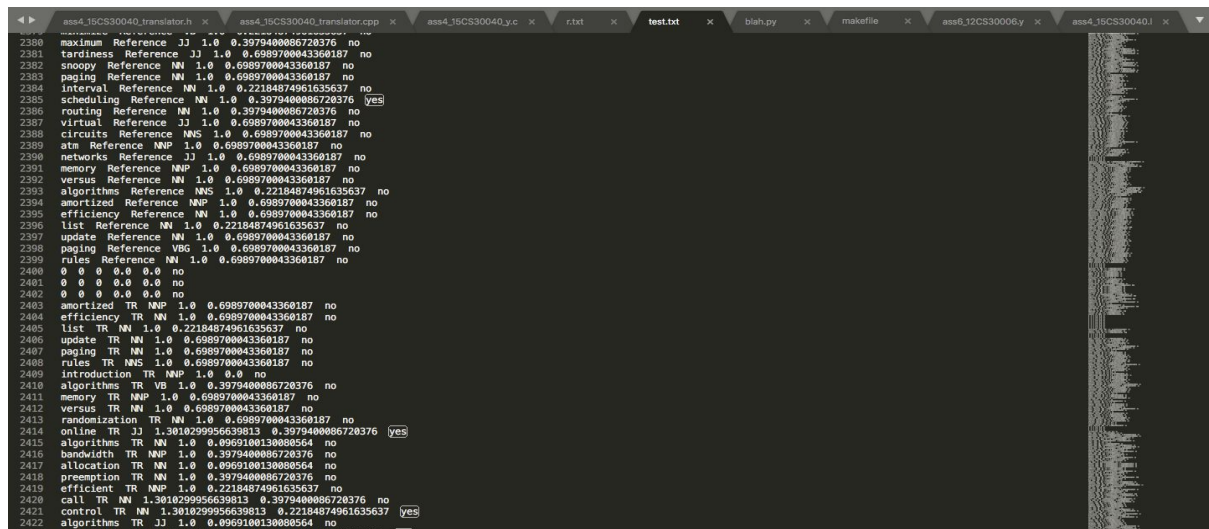
- 1) The dataset used for the initial training can be found [here](#). This dataset^[2] contains text versions of a set of papers published by the ACM in the Computer Science domain in 2003–2005 along with their chunked location in the paper (like title, abstract, etc.). Each text file has another file associated with it containing the key phrases of the paper. The dataset contained ~2300 documents.

- 2) Since the above dataset contained only the text of the paper and not the stylistic features, we constructed a web crawler that downloaded the PDF formats of these papers. So we were able to get around 1950 pdfs along with the stylistic features, using the web crawler. Around 1000 of them had scanned/non-readable text. Thus our final dataset used ~850 pdfs.

Methodology followed:

First we worked on the original dataset, containing only the text and not any stylistic features. We trained a CRF model with the following features:

- *POS tags (of previous, current and next word)*: The word window was determined by examining the dataset. The key phrases were mostly bigrams, and nouns, so the previous word's POS tag was also taken into account.
- *Term frequency*: The key phrases generally had a low frequency compares to other words present in the paper.
- *Inverse Document Frequency*
- *Position in the Paper*: Title/Abstract were more likely to contain keyphrases.
- *Position of the Previous tag*



A snapshot of the feature matrix used to train the CRF model

A CRF model was executed using the CRF++ Library. The result was around 5% precision and recall. We decided to try to incorporate the stylistic features in the system too. We manually downloaded 30 PDFs from the dataset, extracted the stylistic features using the PDF to XML converter called pdftoxml, which comes as a part of the OCR++ package, and then tried to run the same. The results were not promising here, either. The model used here was a unigram model and we decided to use a bigram next since most of the PDFs contained bigram key-phrases.

A web crawler was constructed to download each of the PDFs present in the dataset so that stylistic features can be extracted from each. The web crawler used direct link download and involved providing a small delay associated with the download of each pdf. The PDFs were downloaded using a code which was associated with each paper referred to by the title of the document in the text-only dataset.

Results and Observations

Also, since CRF model did not work with the aforementioned features, we decided to use Random Forests to decide the weight which is required to be given to each feature, if at all used. A bigram model was used. We used a different set of features now:

- *Ratio of tf's of the two words*: In many cases, we observed one of the words to occur with a high tf and the other with a low tf. This was taken into account and defined as tf3, which turned out to be the most important feature in the usage.
- *GM of tf's of the two words in the bigram*: This too turned out to be a much important feature as obtained from the random forest.
- POS Tags
- Stylistic Features
- TF, IDF of bigram
- Zone

The results obtain by the random forest indicating the importance of each feature are mentioned below:

| Features | Description | Importance |
|----------|--|------------|
| TF3 | ratio of tf's of the two words in the bigram | 0.239401 |
| TF2 | GM of tf's of the two words in the bigram | 0.198334 |
| POS | POS Tags of the two words in the bigram | 0.173774 |
| IDF2 | idf corresponding to GM of df's of the two words in the bigram | 0.105704 |
| IDF3 | ratio of df's of the two words in the bigram | 0.101380 |
| IDF1 | idf of the bigram as a whole | 0.098649 |
| TF1 | Tf of the bigram as a whole | 0.043824 |

| | | |
|-------|--|----------|
| Style | Stylistic features: Bold, Italics etc. | 0.021353 |
| Zone | Zone of pdf where bigram was found: Title, Body etc. | 0.017580 |

- It was interesting to note that Stylistic features did not carry much weight in the key-phrase extraction problem, despite the fact that key-phrases are usually bolded or italicized in most cases. This may be because authors tend to italicise even some terms which are not key phrases pertaining to that paper but the domain in general. The Zone feature too did not have much weight.
- The precision and recall results obtained are as follows:
 - Positive Recall: 0.77
 - Positive Precision: 0.78
 - Negative Recall: 0.69
 - Negative Precision: 0.69
 - Accuracy: 0.7
- In the future we would like to work on trying out deep learning to be able to detect latent keywords (i.e. those not present in the text), and also try h2o's random forest implementation as the scikit library random forests can not handle categorical data.

References/Literature:

[Automatic Keyphrase Extraction: A Survey of the State of the Art](#)
[Keyphrases Extraction from Scientific Documents: Improving Machine Learning Approaches with Natural Language Processing](#)
[KEA: Practical Automatic Keyphrase Extraction](#)
[Automatic keyphrase extraction techniques: A review](#)

