Proceedings of the BioCreative V.5 Challenge Evaluation Workshop
This workshop is part of the OpenMinTeD and the ELIXIR-EXCELERATE projects funded from the European Union’s Horizon 2020 research and innovation programme under grant agreements No 654021 and No 676559, respectively.
Funding: Encomienda MINETAD-CNIO as part of the Plan for the Advancement of Language Technology for funding (Plan de Impulso de Tecnologías del Lenguaje) OTG Sanidad Plan TL for funding.
Index

8 The BioCreative V.5 evaluation workshop: tasks, organization, sessions and topics.

11 Evaluation of chemical and gene/protein entity recognition systems at BioCreative V.5: the CEMP and GPRO patents tracks

19 Benchmarking biomedical text mining web servers at BioCreative V.5: the technical Interoperability and Performance of annotation Servers - TIPS track

28 DUTIR at the BioCreative V.5.BeCalm Tasks: A BLSTM-CRF Approach for Biomedical Entity Recognition in Patents

40 HITextracter System for Chemical and Gene/Protein Entity Mention Recognition in Patents

47 CRFVoter: Chemical Entity Mention, Gene and Protein Related Object recognition using a conglomerate of CRF based tools.

54 Neji: Recognition of Chemical and Gene Mentions in Patent Texts

61 Chemlistem - chemical named entity recognition using recurrent neural networks

69 Recognition of Chemical Entity Mention in Patents Using Feature-rich CRF

73 Towards Robust Chemical Recognition with TaggerOne at the BioCreative V.5 CEMP Task

81 A hybrid text mining system for chemical entity recognition and classification using dictionary look-up and pattern matching @ BeCalm challenge evaluation workshop

89 IBEnt: Chemical Entity Mentions in Patents using ChEBI

96 ChemGrab: Identification of Chemical Names Using a Combined Negative-Dictionary and Rule-Based Approach

104 Combining the BANNER tool with the DINTO ontology for the CEMP task of BioCreative V.5

108 An Ensemble Algorithm for Sequential Labelling: A Case Study in Chemical Named Entity Recognition

115 Statistical Principle-based Approach for Gene and Protein Related Object Recognition

122 Tagger: BeCalm API for rapid named entity recognition

130 MER: a Minimal Named-Entity Recognition Tagger and Annotation Server

138 SIA: Scalable Interoperable Annotation Server

146 High-throughput, interoperability and benchmarking of text-mining with BeCalm biomedical metaserver

156 Performance and interoperability assessment of Disease Extract Annotation Server (DEAS)

163 TextImager as an interface to BeCalm

167 Olelo's named-entity recognition web service in the BeCalm TIPS task
175 OGER: OntoGene’s Entity Recogniser in the BeCalm TIPS Task

183 READ-Biomed-Server: A Scalable Annotation Server Using the UIMA Concept Mapper

191 Neji: DIY web services for biomedical concept recognition

196 NTMU-SCHEMA BeCalm API in BioCreative V.5

205 Micro-RNA Recognition in Patents in BioCreative V.5
The BioCreative V.5 evaluation workshop: tasks, organization, sessions and topics

Martin Krallinger*1, Martin Pérez-Pérez2,3, Gael Pérez-Rodríguez2,3, Aitor Blanco-Míguez2,3, Florentino Fdez-Riverola2,3, Salvador Capella-Gutierrez4, Anália Lourenço*2,3,5, Alfonso Valencia6,7,8,9

1Biological Text Mining Unit, Structural Biology and BioComputing Programme, Spanish National Cancer Research Centre, C/Melchor Fernández Almagro 3, E-28029 Madrid, Spain
2ESEI - Department of Computer Science, University of Vigo, Ourense, Spain
3CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain
4Spanish National Bioinformatics Institute Unit, Structural Biology and BioComputing Programme, Spanish National Cancer Research Centre (CNIO), Madrid 28029, Spain
5CEB - Centre of Biological Engineering, University of Minho, Braga, Portugal
6Life Science Department, Barcelona Supercomputing Centre (BSC-CNS). C/Jordi Girona, 29-31, E-08034 Barcelona, Spain
7Joint BSC-IRB-CRG Program in Computational Biology. Parc Científic de Barcelona. C/ Baldiri Reixac 10, E-08028 Barcelona, Spain
8Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig de Lluís Companys 23, E-08010 Barcelona, Spain
9Spanish Bioinformatics Institute INB-ISCIII ES-ELIXIR, Madrid 28029, Spain

{martiperez, gaeperez, aiblanco, riverola, analia}@.uvigo.es;
alfonso.valencia@bsc.es;
{scapella, mkrallinger}@cnio.es

The BioCreative (Critical Assessment of Information Extraction in Biology) is a community-wide effort with the aim of evaluating biomedical text mining and information extraction tools. It is organized in the form of shared tasks or challenges. The evaluation workshop linked to each BioCreative event serves to analyze the results obtained for each track, and to present the used Gold Standard datasets/evaluation settings. For each track, top scoring teams were invited to present a system’s description talk to uncover the used methodology and system’s implementation. Building on the achievements of previous BioCreative evaluations and workshops (BioCreative I, II, II.5, III, 2012 workshop, IV and V [1–5]) we have organized the BioCreative V.5 challenge evaluation workshop in Barcelona, Spain on April 26th-27th, 2017. To promote synergies, this workshop was co-located with the ELIXIR-EXCELERATE workshop on text mining infrastructure requirements, with a particular emphasis on
the role of Text Mining systems for Data Curation and Knowledge Management in the Life Sciences domain. The goal of BioCreative V.5 was to address some of the major barriers to the adoption and use of text mining tools, related to assessment, accessibility, interoperability, robustness and integration. Two traditional BioCreative tracks focused on monitoring progress on the recognition of relevant bio-entities (chemicals – CEMP track and genes/proteins – GPRO track). A novel track called TIPS (Technical interoperability and performance of annotation servers) focused on the technical aspects of the evaluation of continuous text annotation web services. The topics that were addressed during the workshop included: (1) continuous evaluation/stability of text mining tools, (2) enabling of interoperability of multiple text annotation systems at the technical level (design of compatible annotation schemas), (3) extraction of textual content from heterogeneous document sources, and (4) visualization and comparative assessment of automatic and manual annotations. Through the contribution of invited speakers and panel sessions, we also discussed aspects related to annotation formats, technical integration of text mining components, the experience of text mining techniques for the DARPA Big Mechanism program and the use of text processing web services and workflows. During the collocated workshops, the OpenMinted project and the open call for tenders were presented.

Acknowledgment

We acknowledge the OpenMinted (654021) and the ELIXIR-EXCELERATE (676559) H2020 projects, and the Encomienda MINETAD-CNIO as part of the Plan for the Advancement of Language Technology for funding. The Spanish National Bioinformatics Institute (INB) unit at the Spanish National Cancer Research Centre (CNIO) is a member of the INB, PRB2-ISCIII and is supported by grant PT13/0001/0030, of the PE I+D+i 2013-2016, funded by ISCIII and ERDF.

REFERENCES


Evaluation of chemical and gene/protein entity recognition systems at BioCreative V.5: the CEMP and GPRO patents tracks

Martin Pérez-Pérez1,2, Obdulia Rabal3, Gael Pérez-Rodríguez1,2, Miguel Vazquez4, Florentino Fdez-Riverola1,2, Julen Oyarzabal3, Alfonso Valencia5,6,7,8, Anália Lourenço*1,2,9, Martin Krallinger*4

1ESEI - Department of Computer Science, University of Vigo, Ourense, Spain
2CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagos-Marcosende, 36310, Vigo, Spain
3Small Molecule Discovery Platform, Molecular Therapeutics Program, Center for Applied Medical Research (CIMA), University of Navarra, Pamplona, Spain 08010 Barcelona, Spain
4Biological Text Mining Unit, Structural Biology and BioComputing Programme, Spanish National Cancer Research Centre, C/Melchor Fernández Almagro 3, E-28029 Madrid, Spain
5Life Science Department, Barcelona Supercomputing Centre (BSC-CNS). C/Jordi Girona, 29-31, E-08034 Barcelona, Spain
6Joint BSC-IRB-CRG Program in Computational Biology. Parc Científic de Barcelona. C/ Baldirí Reixac 10, E-08028 Barcelona, Spain
7Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig de Lluís Companys 23, E-
8Spanish Bioinformatics Institute INB-ISCIII ES-ELIXIR, Madrid 28029, Spain
9CEB - Centre of Biological Engineering, University of Minho, Braga, Portugal

{martiperez,gaeperez,riverola,analia}@.uvigo.es; {orabal,julenoyarzabal}@unav.es; {mvazquezg,mkrallinger}@cnio.es; alfonso.valencia@bsc.es

Abstract. This paper presents the results of the BioCreative V.5 offline tasks related to the evaluation of the performance as well as assess progress made by strategies used for the automatic recognition of mentions of chemical names and gene in running text of medicinal chemistry patent abstracts. A total of 21 teams submitted results for at least one of these tasks. The CEMP (chemical entity mention in patents) task entailed the detection of chemical named entity mentions. A total of 14 teams submitted 56 runs. The top performing team reached an F-score of 0.90 with a precision of 0.88 and a recall of 0.93. The GPRO (gene and protein related object) task focused on the detection of mentions of gene and protein related objects. The 7 participating teams (30 runs) had to detect gene/protein mentions that could be linked to at least one biological database, such as SwissProt or EntrezGene. The best F-score, recall and precision in this task were of 0.79, 0.83 and 0.77, respectively.
The CEMP and GPRO gold standard corpora included training sets of 21,000 records and test sets of 9,000 records. Similar to the previous BioCreative CHEMDNER tasks, evaluation was based on micro-averaged F-score. The BeCalm platform supported prediction submission and evaluation (http://www.becalm.eu).

**Keywords.** CEMP; GPRO; ChemNLP; BioCreative; Named Entity Recognition; Chemical compounds; Genes/proteins; Text Mining

1 Introduction

The BioCreative V.5 challenge encompassed two offline tasks, which followed the evaluation settings used for previous BioCreative competitions, in addition to a novel online task, which was geared towards the continuous evaluation of named entity annotation web servers. BioCreative is a community challenge with the aim of evaluating biomedical text mining efforts [1].

This paper describes the results obtained by participating teams for the offline tasks, which addressed the automatic extraction of chemical and biological data from medicinal chemistry patents. The CEMP (chemical entity mention in patents) and GPRO (gene and protein related object) tasks entailed the detection of chemical named entity mentions and mentions of gene and protein related objects in patent titles and abstracts, respectively.

Some of the general difficulties for such automatic name recognition in the scientific literature have been already highlighted in previous BioCreative CHEMDNER tasks [2, 3]. Indeed, the settings of the hereby described tasks were very similar to the counterparts in BioCreative V [2]. Briefly, given a set of patent documents, participating teams had to correctly detect the start and end indices corresponding to all the chemical entities (CEMP) and the gene and protein related objects (GPRO). All entities were manually annotated by domain experts using well-defined annotation guidelines [4]. In particular, the covered GPRO entities had to be annotated at a sufficient level of granularity to be able to determine whether the labelled mention could or could not be linked to a specific gene or gene product (represented by an entry of a biological annotation database such as SwissProt [5] or EntrezGene [6]).

The BeCalm Web metaserver platform supported prediction submission and evaluation. Participants could submit a total of five runs per task for final evaluation. The micro-averaged recall, precision and F-
score statistics were used for final prediction scoring, and F-score was selected as main evaluation metric.

2 Task description

The used patent abstract records were released in the form of plain-text, UTF8-encoded patent abstracts in a tab-separated format with the following three columns: (1) patent identifier, (2) title of the patent, (3) abstract of the patent. The annotated document sets used for training were produced with the intent of supporting the improvement of the automatic prediction tools enrolled in the challenge. Conversely, the test sets were used in the controlled comparison of the performance of the participating systems.

<table>
<thead>
<tr>
<th>Table 1: CEMP and GPRO corpora overview.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training set</strong></td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Patent abstracts</td>
</tr>
<tr>
<td>CEMP mentions</td>
</tr>
<tr>
<td>GPRO mentions</td>
</tr>
<tr>
<td>GPRO type 1 mentions</td>
</tr>
<tr>
<td>GPRO type 2 mentions</td>
</tr>
<tr>
<td>Tokens</td>
</tr>
</tbody>
</table>

Furthermore, the annotation carried out for the GPRO task encompassed two types of GPRO entity mentions: GPRO entity mention type 1, i.e. covering those GPRO mentions that can be normalized to a bio-entity database record; GPRO entity mention type 2, i.e. covering those GPRO mentions that in principle cannot be normalized to a unique bio-entity database record (e.g. protein families or domains).

The BeCalm Web metaserver platform enabled both the examination of automatic predictions by participants and final submission benchmarking (Figure 1).
BeCalm support for chemical and gene entity recognition at BioCreative V.5 offline tasks.

BeCalm provided micro- and macro-average standard performance statistics, such as precision, recall and F-score \cite{7, 8}. Furthermore, it enabled the examination of annotation mismatches, i.e. false positive annotations. In final evaluation, three main result types were examined: false negative (FN) results corresponding to incorrect negative predictions (i.e. cases that were part of the gold standard, but missed by the automatic system), false positive (FP) results being cases of incorrect positive predictions (i.e. wrong results predicted by the automatic system that had no corresponding annotation in the gold standard) and true positive (TP) results consisting of correct positive predictions (i.e. correct predictions matching exactly with the gold standard annotations).

Correspondingly, recall (Eq. 1) is the percentage of correctly labelled positive results over all positive cases, being a measure of the ability of a system to identify positive cases.

\[
\text{recall} = \frac{TP}{(TP + FN)} \quad (\text{Eq. 1})
\]

Precision (Eq. 2) represents the percentage of correctly labelled positive results over all positive labelled results, i.e. it is a measure of the reproducibility of a classifier of the positive results.
\[
\text{precision} = \frac{TP}{(TP + FP)} \quad (\text{Eq. 2})
\]

Lastly, F-score (or balanced F-measure) stands for the harmonic mean between precision and recall (Eq. 3).

\[
F - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (\text{Eq. 3})
\]

Partial hits, i.e. predictions that only in part overlapped with the manually defined gold standard annotations, were not taken into account in the analyses. Micro-average statistics were calculated globally by counting the total true positives, false negatives and false positives. Conversely, macro-average statistics were calculated by counting the true positives, false negatives and false positives on a per-document basis and then, averaged across documents.

During the test phase, teams were requested to generate automatic annotations (according to predefined evaluation format) for a blinded collection of documents, and submit them after a short period of time. Teams could submit for each of the tasks up to five predictions (runs). The micro-averaged recall, precision and F-score statistics were used for final prediction scoring, and F-score was selected as main evaluation metric. Furthermore, the statistical significance of each prediction with respect to the other final submissions was examined by means of a Bootstrap resampling simulation, in a similar way to what was done in previous CHEMDNER challenges [2, 3]. This statistical analysis was done for both the CEMP and GPRO tasks by taking 4,500 bootstrapped samples from all the annotated documents in the test sets (a total of 9,000 documents in each set). The micro-average F-scores for each team on each sample were calculated and these 2,500 resampled results were further used to calculate the standard deviation of the F-score of each team (SDs). Teams were grouped based on statistically significant difference (at two SD) between results.

The annotation guidelines (as well as the GPRO guidelines) were published together with the manually annotated corpora in order for teams to actually understand how the annotations were done and to make it possible to examine how their systems could consider the annotation rules.
3 Results

A total of 21 teams submitted results for at least one of the two offline tasks. For both tasks, the training set consisted of 21,000 patent records and the test set consisted of 9,000 patent records.

A total of 14 teams submitted 56 runs for the CEMP task. As illustrated in Table 2, the top performing team reached an F-score of 0.90 with a precision of 0.88 and a recall of 0.93. The top scoring run in terms of F-score was generated by team 121 (from a total of 5 runs). The three top performing teams, namely teams 121, 112 and 107, reached an F-score of over 0.90. The highest precision was obtained by team 107 (0.90) while the highest recall was obtained by team 116 (0.93).

<table>
<thead>
<tr>
<th>Row</th>
<th>Team</th>
<th>F-score</th>
<th>Precision</th>
<th>Recall</th>
<th>SD</th>
<th>Range</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>90.42</td>
<td>88.32</td>
<td>92.62</td>
<td>0.25%</td>
<td>A-C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>112</td>
<td>90.37</td>
<td>88.97</td>
<td>91.82</td>
<td>0.27%</td>
<td>A-C</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>107</td>
<td>90.32</td>
<td>90.02</td>
<td>90.62</td>
<td>0.27%</td>
<td>A-C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>153</td>
<td>89.14</td>
<td>88.02</td>
<td>90.28</td>
<td>0.3%</td>
<td>D-E</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>116</td>
<td>88.47</td>
<td>84.39</td>
<td>92.97</td>
<td>0.23%</td>
<td>D-F</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>144</td>
<td>87.29</td>
<td>87.42</td>
<td>87.15</td>
<td>0.34%</td>
<td>E-G</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>102</td>
<td>86.59</td>
<td>89.01</td>
<td>84.29</td>
<td>0.33%</td>
<td>F-J</td>
<td>5</td>
</tr>
<tr>
<td>H</td>
<td>142</td>
<td>85.68</td>
<td>83.1</td>
<td>88.42</td>
<td>0.32%</td>
<td>G-J</td>
<td>6</td>
</tr>
<tr>
<td>I</td>
<td>117</td>
<td>85.44</td>
<td>88.42</td>
<td>82.64</td>
<td>0.32%</td>
<td>G-J</td>
<td>6</td>
</tr>
<tr>
<td>J</td>
<td>127</td>
<td>85.31</td>
<td>87.32</td>
<td>83.38</td>
<td>0.36%</td>
<td>G-J</td>
<td>6</td>
</tr>
<tr>
<td>K</td>
<td>135</td>
<td>83.95</td>
<td>85.68</td>
<td>82.28</td>
<td>0.37%</td>
<td>K</td>
<td>7</td>
</tr>
<tr>
<td>L</td>
<td>125</td>
<td>82.45</td>
<td>83.1</td>
<td>81.81</td>
<td>0.3%</td>
<td>L</td>
<td>8</td>
</tr>
<tr>
<td>M</td>
<td>110</td>
<td>59.24</td>
<td>52.93</td>
<td>67.26</td>
<td>0.35%</td>
<td>M</td>
<td>9</td>
</tr>
<tr>
<td>N</td>
<td>170</td>
<td>49.25</td>
<td>47.18</td>
<td>51.52</td>
<td>0.19%</td>
<td>N</td>
<td>10</td>
</tr>
</tbody>
</table>

The 7 teams that participated in the GPRO task submitted a total of 30 runs. Here, evaluation was two-fold: based only on annotations of type 1 (i.e. those that can be normalized to a bio-entity database record), and considering both annotation types (i.e. normalized or not to a bio-entity database record). For the GPRO type 1 evaluation, team 121 was the best performing team (achieved an F-score of 0.79), team 133 got the best recall (0.83) and team 144 obtained the best precision (0.77) (Table 3).
Table 3. GPRO type 1 evaluation results (best runs per team only).

<table>
<thead>
<tr>
<th>Row</th>
<th>Team</th>
<th>F-score</th>
<th>Precision</th>
<th>Recall</th>
<th>SD%</th>
<th>Range</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>79.19</td>
<td>76.65</td>
<td>81.91</td>
<td>0.1%</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>112</td>
<td>76.34</td>
<td>75.23</td>
<td>77.49</td>
<td>0.08%</td>
<td>B-C</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>153</td>
<td>76.13</td>
<td>72.06</td>
<td>80.68</td>
<td>0.1%</td>
<td>B-C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>133</td>
<td>73.73</td>
<td>66.53</td>
<td>82.68</td>
<td>0.1%</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>142</td>
<td>73.18</td>
<td>74.79</td>
<td>71.63</td>
<td>0.15%</td>
<td>E-F</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>144</td>
<td>73.07</td>
<td>76.86</td>
<td>69.62</td>
<td>0.17%</td>
<td>E-F</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>102</td>
<td>71.3</td>
<td>71.52</td>
<td>71.09</td>
<td>0.14%</td>
<td>G</td>
<td>5</td>
</tr>
</tbody>
</table>

For GPRO type 1 and type 2 evaluation, team 133 achieved top performing F-score (0.79) and recall (0.79) while team 153 obtained the best precision (0.84) (Table 4).

Table 4. GPRO type 1 and type 2 evaluation results (best runs per team only).

<table>
<thead>
<tr>
<th>Row</th>
<th>Team</th>
<th>F-score</th>
<th>Precision</th>
<th>Recall</th>
<th>SD%</th>
<th>Range</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>133</td>
<td>78.66</td>
<td>78.63</td>
<td>78.7</td>
<td>0.05%</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>153</td>
<td>77.11</td>
<td>83.95</td>
<td>71.3</td>
<td>0.06%</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>112</td>
<td>75.91</td>
<td>80.41</td>
<td>71.89</td>
<td>0.04%</td>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>144</td>
<td>74.92</td>
<td>79.78</td>
<td>70.63</td>
<td>0.09%</td>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>121</td>
<td>72.28</td>
<td>81.56</td>
<td>64.89</td>
<td>0.12%</td>
<td>E</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>142</td>
<td>64.96</td>
<td>74.99</td>
<td>57.3</td>
<td>0.1%</td>
<td>F</td>
<td>6</td>
</tr>
<tr>
<td>G</td>
<td>102</td>
<td>62.24</td>
<td>77.75</td>
<td>51.89</td>
<td>0.1%</td>
<td>G</td>
<td>7</td>
</tr>
</tbody>
</table>

4 Discussion

A total of 14 teams have participated in BioCreative V.5. Compared to the systems that participated in the previous BioCreative V CHEMDNER task, the average results were better with 0.82 vs 0.76, 0.81 vs 0.77 and 0.83 vs 0.74, in terms of f-score, precision and recall, respectively. This time, the best f-score was 0.90 (three teams), slightly better than in BioCreative V CHEMDNER task (0.88). In view of the results, this task has reached the maximum performance one could expect taking into account the intrinsic difficulty of the task and the provided annotation quality.

Participation in the GPRO task has improved in BioCreative V.5. Considering GPRO entity mention type 1, i.e. GPRO mentions that can be normalized to a bio-entity database, the results for the best team were slightly worse than in BioCreative V CHEMDNER task, i.e. the f-score declined from 0.81 to 0.79, but the average team results were better with
0.74 vs 0.65, 0.73 vs 0.66 and 0.76 vs 0.64, in terms of f-score, precision and recall, respectively.
In the present GPRO task, entity mention type 2, i.e. non normalised mentions, were also evaluated. Comparing results for GPRO entity mention type 1 to results for GPRO entity mentions type 1 and 2, it is observable that systems have a better average recall for entity mention type 1 (0.76 vs 0.66) while precision is better when considering both types (0.73 vs 0.79).
Overall, the obtained results in CEMP and GPRO are considered competitive enough to derive in tools that not only could assist manual curation, but also could be used to automatic annotation extraction and patent abstract chemical indexing.

5 Acknowledgment
We acknowledge the OpenMinted (654021) and the ELIXIR-EXCELERATE (676559) H2020 projects, and the Encomienda MINETAD-CNIO as part of the Plan for the Advancement of Language Technology for funding. The Spanish National Bioinformatics Institute (INB) unit at the Spanish National Cancer Research Centre (CNIO) is a member of the INB, PRB2-ISCIII and is supported by grant PT13/0001/0030, of the PE I+D+i 2013-2016, funded by ISCIII and ERDF.

REFERENCES
Benchmarking biomedical text mining web servers at BioCreative V.5: the technical Interoperability and Performance of annotation Servers - TIPS track

Martin Pérez-Pérez1,2, Gael Pérez-Rodríguez1,2, Aitor Blanco-Míguez1,2, Florentino Fdez-Riverola1,2, Alfonso Valencia3,4,5,6, Martin Krallinger*7, Anália Lourenço*1,2,8

1ESEI - Department of Computer Science, University of Vigo, Ourense, Spain
2CINBIO - Centro de Investigaciones Biomédicas, University of Vigo, Campus Universitario Lagoas-Marcosende, 36310, Vigo, Spain
3Life Science Department, Barcelona Supercomputing Centre (BSC-CNS). C/Jordi Girona, 29-31, E-08034 Barcelona, Spain
4Joint BSC-IRB-CRG Program in Computational Biology. Parc Científic de Barcelona. C/Baldri Reixa 10, E-08028 Barcelona, Spain
5Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig de Lluis Companys 23, E-08010 Barcelona, Spain
6Spanish Bioinformatics Institute INB-ISCIII ES-ELIXIR, Madrid 28029, Spain
7Biological Text Mining Unit, Structural Biology and BioComputing Programme, Spanish National Cancer Research Centre, C/Melchor Fernández Almagro 3, E-28029 Madrid, Spain
8CEB - Centre of Biological Engineering, University of Minho, Braga, Portugal

{martiperez, gaeperez, riverola, analia}@.uvigo.es; alfonso.valencia@bsc.es, mkrallinger@cnio.es

Abstract. The TIPS track consisted in a novel experimental task under the umbrella of the BioCreative text mining challenges with the aim to, for the first time ever, carry out a text mining challenge with particular focus on the continuous assessment of technical aspects of text annotation web servers, specifically of biomedical online named entity recognition systems.

A total of 13 teams registered annotation servers, implemented in various programming languages, supporting up to 12 different general annotation types. The continuous evaluation period took place from February to March 2017. The systematic and continuous evaluation of server responses accounted for testing periods of low activity and moderate to high activity. Moreover three document provider settings were covered, including also NCBI PubMed. For a total of 4,092,502 requests, the median response time for most servers was below 3.74 s with a median of 10 annotations/document. Most of the servers showed great reliability and stability, being able to process 100,000 requests in 5 days.
Keywords. TIPS; BeCalm metaserver; Document provider; Annotation server; Continuous evaluation; BioCreative; Text Mining

1 Introduction

There is an increasing demand in being able to effectively access, evaluate, compare, visualise and integrate multiple text mining systems in order to process natural language document collections. Several BioCreative tasks tried to promote the development of online text annotation servers [1–4]. In particular, the BioCreative Meta-Server was the first distributed prototype platform requesting, retrieving and unifying biomedical textual annotations [5]. Despite the relevance of those previous efforts, several crucial aspects have not been sufficiently or only partially addressed, including continuous evaluation, extraction of textual content from heterogeneous sources, harmonisation of multiple different biomedical text annotation types and visualisation and comparative assessment of automatic and manual annotations. This inspired the conception of the BeCalm Technical Interoperability and Performance of annotation Servers (TIPS) task for the BioCreative V.5 challenge.

This novel task focused on the technical aspects of making text-mining systems available, interoperable and continuously evaluating the underlying named entity recognition web annotation servers. The participant annotation servers could be fully developed in-house or integrate/adapt third party recognition software as building block components. Furthermore, there were no restrictions in terms of named entity types/classes, thus covering entity type such as genes, proteins, chemicals, diseases or species among others.

In line with the efforts of ELIXIR/EXCELERATE in benchmarking the ELIXIR catalogue of methods and the OpenMinted interoperability specifications (http://openminted.eu/), both a minimal set of functional specifications (metadata info) and the use of a common communication protocol for serializing and distributing text annotations were reinforced. Specifically, the TIPS task considered three levels of evaluation: data level (i.e., data formats), technical level (i.e., stability and response time), and functional specification level (i.e., metadata requirements).

TIPS was supported by the BeCalm biomedical annotation metaserver (http://www.becalm.eu/) that enabled the continuous evaluation of annotation server performance as well as individual server monitoring by the
participating teams. Annotation servers were asked to implement a Representational State Transfer (REST) API application that listens and responds to the requests made by the BeCalm metaserver. Annotation/prediction requests were issued on a regular basis, emulating different daily request loads during the months of February and March, 2017. Servers were forbidden to cache the documents, i.e., each document should be downloaded from the specified source whenever requested. Servers also should not cache the generated predictions, i.e., each document should be analysed for every request.

The aim of this paper is to describe the TIPS task and the specific support provided by BeCalm metaserver. The next sections present the architectural design of the metaserver, how the platform was utilised by the participants throughout the competition, and TIPS evaluation results.

2 BeCalm metaserver platform

The fundamental aim of the BeCalm biomedical annotation platform is to provide users with annotations on biomedical texts gathered from different systems. The platform is to be regarded as a distributed system requesting, retrieving and unifying textual annotations, to further deliver these data to the user at different levels of granularity.

For communication purposes, the system utilizes the REST API protocol [6]. The metaserver sends requests to annotate documents to all known/registered annotation servers. Once the annotation servers have finished processing the text, the predictions are returned to the metaserver and stored in its central repository. BeCalm REST API is publicly available at http://www.becalm.eu/api.

In assistance to TIPS competition (Figure 1), the BeCalm platform provided a user-friendly monitoring environment, where participating teams could manage annotation servers and examine their performance throughout the TIPS competition. Moreover, this monitoring environment offered participants the possibility of testing communication between the metaserver and the server, so that they could acquire insights on possible server improvements.

Regarding TIPS administration and functioning, the BeCalm platform enabled the registration of participants, the scheduling of annotation/prediction requests for continuous evaluation, the systematic calculation of server performance metrics, and a detailed log of events at both metaserver and server levels.
BeCalm interface is based on the open source CakePHP framework [7] and involves mainstream Web user-system interaction technologies, such as HTML5 (http://www.w3.org/TR/html5/), CSS3 (http://www.css3.info/), Ajax and JQuery technologies (http://jquery.com/).

3 TIPS competition

TIPS competition evaluates the technical aspects of making available and evaluating text annotation servers for continuous named entity recognition. At this first edition, servers were evaluated on the basis of single document requests.

3.1 TIPS evaluation levels

TIPS evaluation encompassed data format considerations, technical metrics and functional specifications. At the data level, evaluation addressed the ability of annotation servers to return NER annotation results as structured data, represented in one or several of the following formats XML/BioC, JSON/BioCJSON or TXT/TSV. The ability to retrieve and
process documents from different providers (i.e., patents server, abstract server, and PubMed) was also examined. Stability and response time were at the core of technical assessments. Stability metrics aimed to describe server ability to respond to continuous requests, to respond within a stipulated time window, and to provide updated server status information. Conversely, response time statistics described the time taken by the annotation server to respond to a request, measured in terms of the number and contents of the requested documents and the volume of predictions returned.

Functional specifications were inspired by the OpenMinTeD interoperability project (http://openminted.eu/). Server registration encompassed mandatory, recommended and optional metadata. Mandatory metadata included server name, institution/company, server administrator, programming language (main language, if using several), integration of third-party recognition software, recognised annotation types (e.g., chemical entities, genes, proteins, diseases, organisms, cellular lines and types, and mutations), supported annotation formats (e.g., XML/BioC, JSON/BioCJSON or TXT/TSV) and version control. Software license, specification of third-party recognition software (if any), dedicated vs shared server, and relevant publications were considered recommended metadata. Optionally, teams could also provide details on server operating system, distributed processing, and hardware characteristics (i.e., number of processors and RAM information).

3.2 TIPS evaluation metrics

Traditional annotation quality metrics (e.g., precision, recall, and F-score) were not part of TIPS evaluation. Rather, this novel task only evaluated performance metrics, namely reliability indicators and performance indicators (Table 1).

The mean time between failures (MTBF) and the mean time to repair (MTTR) are the key reliability indicators. Conversely, the mean annotations per document (MAD), the mean time per document volume (MTDV), the mean time seek annotations (MTSA), and the average response time (ART) are the key performance indicators.
### Table 1. Description of TIPS evaluation metrics.

<table>
<thead>
<tr>
<th>Name</th>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTBF</td>
<td>$\frac{\sum(\text{start of downtime}(\text{failure n + 1}) - \text{start of uptime}(\text{failure n}))}{\text{number of failures}}$</td>
<td>Average elapsed time between failures of an annotation server.</td>
</tr>
<tr>
<td>MTTR</td>
<td>$\frac{(\sum(\text{end of downtime}(n) - \text{start of downtime}(n)))}{\text{number of failures}}$</td>
<td>Average time required to repair a failure in an annotation server, i.e. the necessary time to start the server again when a period of downtime occurs.</td>
</tr>
<tr>
<td>MAD</td>
<td>$\frac{\text{total number of annotations}}{\text{total number of responses}}$</td>
<td>Number of annotations divided by the total number of responses.</td>
</tr>
<tr>
<td>MTDV</td>
<td>$\frac{\sum\text{response time}}{\sum\text{document size}}$</td>
<td>Average time that the server takes to annotate a document (i.e. answer a request) based on the sum of the document sizes (in bytes) for all responses.</td>
</tr>
<tr>
<td>MTSA</td>
<td>$\frac{\sum\text{response time}}{\text{total number of annotations}}$</td>
<td>Sum of the response times divided by the total number of annotations produced.</td>
</tr>
<tr>
<td>ART</td>
<td>$\frac{\sum\text{response time}}{\text{total number of responses}}$</td>
<td>Average time to respond to a request.</td>
</tr>
</tbody>
</table>

### 4 Results

A total of 13 unique teams participated in TIPS. The annotation servers support a total of 12 unique annotation types. The chemical and disease types are the annotation types with greatest support (10 and 9 servers, respectively). The maximum number of types supported by a single server was 10 (server 120). Also, servers are implemented in various programming languages, namely Java (the most recurring), C#, C++, Node.JS, bash, Ruby, Python, Crystal.

The evaluation period started at February 5\textsuperscript{th} 2017 and ended March 30\textsuperscript{th} 2017. The aim was to perform a systematic and continuous evaluation of server response under a varied request workload. So, the scheduling of annotation requests accounted for periods of low activity and moderate to high activity as well as for the three document providers, including a mix of them (Figure 2).
Figure 2. Requests issued per document provider throughout the evaluation period. (A) The plot depicts request per competition weeks from February 2017 to March 2017. (B) Information about the number of requests issued in February and March (semicolon separated) per document provider and request type.

Final performance results are shown in Table 2.

Table 2. TIPS evaluation data. Bold data represents the top values for each metric.

<table>
<thead>
<tr>
<th>ID</th>
<th>#Requests</th>
<th>#Predictions</th>
<th>MTSA</th>
<th>MTDV</th>
<th>MAD</th>
<th>ART</th>
<th>MTBF</th>
<th>MTTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>3.19E+05</td>
<td>6.70E+05</td>
<td>7.58E-01</td>
<td>1.32E-03</td>
<td>2.13E+00</td>
<td>1.61E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>106</td>
<td>3.12E+05</td>
<td>4.07E+06</td>
<td>8.59E-02</td>
<td>9.42E-04</td>
<td>1.34E+01</td>
<td>1.15E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>107</td>
<td>2.95E+05</td>
<td>1.14E+06</td>
<td>2.85E-02</td>
<td>1.00E+00</td>
<td>4.27E+00</td>
<td>1.22E+03</td>
<td>4.62E+05</td>
<td>2.23E+05</td>
</tr>
<tr>
<td>108</td>
<td>1.23E+05</td>
<td>0.00E+00</td>
<td>*-</td>
<td>3.03E-02</td>
<td>0.00E+00</td>
<td>3.63E+01</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>111</td>
<td>3.11E+05</td>
<td>5.59E+05</td>
<td>3.55E+02</td>
<td>6.48E-01</td>
<td>2.27E+00</td>
<td>8.06E+02</td>
<td>5.19E+05</td>
<td>2.12E+04</td>
</tr>
<tr>
<td>114</td>
<td>3.19E+05</td>
<td>4.78E+06</td>
<td>1.21E+01</td>
<td>1.48E+03</td>
<td>1.51E+01</td>
<td>1.82E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>116</td>
<td>2.29E+05</td>
<td>2.31E+06</td>
<td>3.83E+02</td>
<td>7.55E+00</td>
<td>2.35E+01</td>
<td>9.01E+03</td>
<td>8.11E+04</td>
<td>4.65E+05</td>
</tr>
<tr>
<td>117</td>
<td>3.19E+05</td>
<td>7.13E+06</td>
<td>1.29E+01</td>
<td>2.38E+03</td>
<td>2.25E+01</td>
<td>2.90E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>120</td>
<td>2.91E+05</td>
<td>2.74E+07</td>
<td>1.37E-02</td>
<td>1.15E-03</td>
<td>1.01E+02</td>
<td>1.39E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>121</td>
<td>3.19E+05</td>
<td>3.30E+06</td>
<td>1.18E-01</td>
<td>9.96E-04</td>
<td>1.04E+01</td>
<td>1.22E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>122</td>
<td>3.16E+05</td>
<td>4.42E+06</td>
<td>7.23E-02</td>
<td>8.58E-04</td>
<td>1.48E+01</td>
<td>1.07E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>124</td>
<td>4.98E+04</td>
<td>2.98E+04</td>
<td>1.55E+01</td>
<td>4.49E-02</td>
<td>3.29E+00</td>
<td>5.14E+01</td>
<td>1.17E+06</td>
<td>6.09E+04</td>
</tr>
<tr>
<td>126</td>
<td>4.98E+04</td>
<td>3.22E+04</td>
<td>1.50E+01</td>
<td>5.00E-02</td>
<td>3.69E+00</td>
<td>5.58E+01</td>
<td>5.86E+05</td>
<td>8.98E+04</td>
</tr>
<tr>
<td>127</td>
<td>3.19E+05</td>
<td>2.79E+06</td>
<td>4.20E-01</td>
<td>3.07E-03</td>
<td>8.90E+00</td>
<td>3.74E+00</td>
<td>4.58E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>128</td>
<td>1.87E+05</td>
<td>8.57E+05</td>
<td>5.44E+02</td>
<td>6.35E+00</td>
<td>1.38E+01</td>
<td>7.52E+03</td>
<td>1.73E+05</td>
<td>1.47E+05</td>
</tr>
</tbody>
</table>

*This server provided empty prediction files for all requests.
Servers 103, 114, 117, 121 and 127 have processed the biggest number of requests (3.19E+05). Server 120 has generated the largest number of predictions (2.74E+07), with an average of 101 predictions per document (MAD). In average, each prediction for server 120 has been generated in 0.013 s (MTSA). The minimum processing time value (ART) was 1.07 s, and the minimum processing time per document volume (MTDV) was 8.58E-04 bytes/s (server 122). During the whole TIPS competition, 9 servers have operated uninterrupted. Among the rest, the server 111 had the smallest recovering score (MTTR) with a value of 5.8 h.

5 Discussion

Overall, server performance metrics are quite encouraging, for example, for a total of 4,092,502 requests, the median response time for most servers was below 3.74s with a median of 10 annotations per document. In terms of document provider, the median response time was 2.85s for the patent server, 3.01s for the abstract server and 3.48s for PubMed. PubMed slightly higher times are justified by the need of retrieving the abstracts at the time of the request, i.e. depending on PubMed service. Most of the servers showed great reliability and stability. Most of them were able to process 100,000 requests, for different providers, in five days. Considering that many participants have stated that their servers could perform batch processing, this figure is very promising, because the volume of processed documents could grow easily to one million documents.

Following this development path, the next TIPS evaluation phases will address multi-document requests, stress server tests and full-text annotation requests.

6 Acknowledgment

We acknowledge the OpenMinted (654021) and the ELIXIR-EXCELERATE (676559) H2020 projects, and the Encomienda MINETAD-CNIO as part of the Plan for the Advancement of Language Technology for funding. The Spanish National Bioinformatics Institute (INB) unit at the Spanish National Cancer Research Centre (CNIO) is a member of the INB, PRB2-ISCIII and is supported by grant PT13/0001/0030, of the PE I+D+i 2013-2016, funded by ISCIII and ERDF.
REFERENCES


7. Iglesias M (2011) CakePHP 1.3 application development cookbook: over 60 great recipes for developing, maintaining, and deploying web applications. Packt Pub
DUTIR at the BioCreative V.5.BeCalm Tasks: A BLSTM-CRF Approach for Biomedical Entity Recognition in Patents

Ling Luo, Pei Yang, Zhihao Yang*, Hongfei Lin, Jian Wang
College of Computer Science and Technology,
Dalian University of Technology, Dalian, China 116024
{lingluo, yangperasd}@mail.dlut.edu.cn;
{yangzh, hflin, wangjian}@dlut.edu.cn

Abstract. Patents contain the significant amount of information. Biomedical text mining has received much attention in patents recently, especially in the medicinal chemistry domain. The BioCreative V.5.BeCalm tasks focus on biomedical entities recognition in patents. This paper describes our method used to create our submissions to the Chemical Entity Mention recognition (CEMP) and Gene and Protein Related Object recognition (GPRO) subtasks. In our method, a bidirectional Long Short-Term Memory with a conditional random field layer (BLSTM-CRF) is employed to recognize biomedical entities from patents. Our best CEMP submission achieves an F-score of 90.42% and our best GPRO submission with type 1 achieves an F-score of 79.19%.

Keywords. Patents; Biomedical Entity Recognition; Deep Learning; Long Short-Term Memory; Conditional Random Field

1 Introduction

Biomedical named entity recognition (NER) is a fundamental step for the complex biomedical natural language processing (NLP) tasks (such as entity relation extraction). In the previous BioCreative tasks [1-3], various approaches have been proposed to recognize biomedical entities from the scientific literature. In addition to the scientific literature, patents are another important source, and they contain a wealth of useful biomedical information. Therefore, automatic extraction the information contained in patents has received much attention, especially in the medicinal chemistry domain. Among others, automatic biomed-
cal entity recognition from medicinal chemistry patents has become an important research task [4].

To promote the development of NER systems, the BioCreative V.5, a major challenge event in biomedical natural language processing, proposes the BeCalm tasks. This challenge included three individual subtasks. The first subtask is technical interoperability and performance of annotation servers (TIPS). This task focuses on the technical aspects of the evaluation of continuous text Annotation Servers for NER. The second subtask is chemical entity mention recognition (CEMP). This task requires the detection of chemical named entity mentions from patent titles and abstracts. The third subtask is gene and protein related object recognition (GPRO). The task requires the detection of gene and protein related objects mentions from patent titles and abstracts. We participated in the CEMP and GPRO subtasks and our submissions to the two subtasks are created by the deep learning model, which is a bidirectional Long Short-Term Memory with a conditional random field layer (BLSTM-CRF). Our methods and results are presented in the following sections.

2 Discussion

Similar to many NER tasks, we modeled the CEMP and GPRO tasks as a sequence labeling problem. We used the BIO (Begin, Inside, Outside) tagging scheme since it achieves better performance than BIOES tagging scheme in our experiments. For the challenge, we presented the neural network architecture, a bidirectional Long Short-Term Memory with a conditional random field layer (BLSTM-CRF), to recognize biomedical entities from patents. The processing flow of our method is shown in Figure 1. Firstly, some preprocessing steps including text cleaning, sentence splitting and tokenization are performed. Secondly, a word embedding is learned with large amounts of unlabeled data by the word2vec tool. Moreover, we induce the character embedding and linguistic feature embeddings. Then with the embeddings as input, a BLSTM-CRF model is trained by the annotated training set. Finally, some post-processing steps including tagging consistency, abbreviation resolution and bracket balance are employed. The process is described in details in the following sections.
2.1 Features

Word embedding and character embedding are widely used in the field of NLP, especially based on the deep learning methods. We used them as the features of our baseline. Moreover, to investigate the effects of traditional linguistic features (such as part of speech (POS) and chunking), these linguistic features are added into the baseline as different runs. All feature embeddings are parameters of the model, and they can be optimized when the model is trained. Details of each of features are presented as follows.

2.1.1 Word Embedding

Word embedding, also known as distributed word representation, can capture both the semantic and syntactic information of words from a large unlabeled corpus and has attracted considerable attention from many researchers [5]. Compared with the bag-of-words (BOW) representation, word embedding is low-dimensional and dense. In recent years, several models, such as word2vec [6] and GloVe [7], have been proposed and widely used in the field of NLP. To achieve a high-quality word embedding, we downloaded a total of 1,918,662 MEDLINE abstracts from the PubMed website with the query string “drug” as the unlabeled data. Then the data and all datasets (The training set comprises a total of 21,000 abstracts, and the test set comprises a total of 9,000 abstracts.) provided in the BeCalm tasks were used to train 50-dimensional word embedding by the word2vec tool as pre-trained word embedding.
2.1.2 Character Embedding

In addition to the word embedding, character-level features in a name contain rich structure information of the entity. These features (such as character n-grams, prefixed and suffixes) are commonly employed in the current NER methods [8]. Unlike the previous traditional methods in which character features are based on hand-engineering, character embedding can be learned while training. Character embedding has been found useful for many NLP tasks. They can not only learn interior representations of the names, but also alleviate the out-of-vocabulary problem. In our method, a character lookup table which contains a 25-dimensional embedding for every character is initialized randomly. Then the character embedding corresponding to every character in a word is given in both direct and reverse orders to a bidirectional LSTM. At last, the concatenation of the forward and backward representations from the bidirectional LSTM is used as the character-level feature of the word.

2.1.3 Linguistic Feature Embedding

Due to the complexity of the natural language, some linguistic features are often employed in traditional machine learning methods [9]. We also explored the effect of linguistic features (such as POS and chunking). The POS information and chunking information of each word were generated by the GENIA tagger [10]. In addition, named entity tags information generated by the GENIA tagger was also used as a feature. The dimensions of the POS, chunking and named entity tags embedding are 25, 10 and 5, respectively. They were initialized randomly.

2.2 BLSTM-CRF Model

The architecture of our model, a bidirectional Long Short-Term Memory with a conditional random field layer (BLSTM-CRF), is illustrated in Figure 2.

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data. Giving a sequence of vectors \((x_1, x_2, ..., x_t, ..., x_n)\) as input, they return another corresponding sequence \((h_1, h_2, ..., h_t, ..., h_n)\). The current state \(h_t\) is generated from the input \(x_t\) and the state \(h_{t-1}\) that is passed forward though time. However, traditional RNNs have the mathematical challenge of learning long-term dependencies. The main problem is that gradients propagated over
many stages tend to vanish. When the sequence is long, traditional RNNs are difficult to work well. To alleviate this problem, Long Short-Term Memory (LSTM) [11] is designed by incorporating a memory cell with the gating mechanism and has been shown to capture long-range dependencies. Therefore, LSTM is applied in our method. LSTM memory cell is implemented as the following:

\[
\begin{align*}
    i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
    c_t &= (1-i_t) \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
    o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
    h_t &= o_t \odot \tanh(c_t)
\end{align*}
\]

where \(\sigma\) is the element-wise sigmoid function, and \(\odot\) is the element-wise product. \(\{W_{xi}, W_{hi}, W_{ci}, W_{xc}, W_{hc}, W_{xo}, W_{ho}, W_{co}\}\) is the weight matrix set. \(\{b_i, b_c, b_o\}\) is the bias vector set.

However, the LSTM’s hidden state \(h_t\) only takes the information from the left context of the sequence at every time \(t\). An elegant solution is a bidirectional LSTM (BLSTM) [12]. In the BLSTM architecture, a forward LSTM computes a representation \(\overline{h_t}\) of the sequence

Figure 2. The architecture of BLSMT-CRF model
from left to right, and another backward LSTM computes a representation $\mathbf{h}_t$ of the same sequence in reverse. These two distinct networks use different parameters, and then the representation of a word is obtained by concatenating its left and right context representations, i.e. $\mathbf{h}_t = [\mathbf{h}^L_t; \mathbf{h}^R_t]$. The representation can make use of rich context information in predicting the current tag.

In the NER tasks, the output labels have strong dependencies. In addition to information of the word itself and the context, the entity tag of the word is also decided by the context tags information of the word. For example, in a reasonable entity tag sequence, the tag “I” generally appears after the tag “B”, but it does not appear after the tag “O”. However, the above-mentioned BLSTM model only uses the $\mathbf{h}_t$ to make independent tagging decisions for each output. Therefore, instead of modeling tagging decisions independently, the CRF layer are added after the BLSTM layer to decode the best tag path in all possible tag paths.

In our method, the NER task is to assign an entity tag to every word in a sentence. Firstly, the word embedding, character embedding and linguistic feature embedding are concatenated as input to feed the BLSTM layer. Then the output of BLSTM layer is fed into the CRF layer. At last, the CRF layer decodes the best tag path.

To be more specific, we introduce a tagging transition matrix $A$, where $A_{i,j}$ represents the score of transition from tag $i$ to tag $j$ in successive words. This transition matrix will be trained as the parameter of model. We define $\theta$ as the set of parameters for the original BLSTM, and $\theta’ = \theta \cup \{A_{i,j}\forall i, j\}$ as the set of all parameters for the BLSTM-CRF model. For an input sentence $[x]^T$ where $T$ is the length of the sentence, we consider $f_\theta([x]^T)$ to be the matrix of scores output by the BLSTM network. The element $[f_\theta]_{i,j}$ of the matrix is the score of the $i^{th}$ tag of the $j^{th}$ word in the sentence. The score of the sentence along with a tag path $[i]^T$ is then given by the sum of transition scores and network scores:

$$S([x]^T, [i]^T, \theta') = \sum_{i=1}^{T} (A_{[i]_{i+1}(i)} + [f_\theta]_{[i]_{i+1}})$$

(5)
Then we use a softmax function to yield the conditional probability of the path \([y]_1^T\) by normalizing the above score over all possible tag paths \([j]_1^T\):

\[
p([y]_1^T | [x]_1^T, \theta') = \frac{e^{S([x]_1^T, [y]_1^T, \theta')}}{\sum_j e^{S([x]_1^T, [j]_1^T, \theta')}}
\]  

(6)

During the training phase, we maximize the log-probability of the correct tag sequence:

\[
\log P([y]_1^T | [x]_1^T, \theta') = S([x]_1^T, [y]_1^T, \theta') - \log \sum_{\eta \in \Pi} e^{S([x]_1^T, [\eta]_1^T, \theta')}
\]  

(7)

Stochastic gradient descent (SGD) is used to optimize the model parameters. At inference time, we predict the best tag path that obtains the maximum score given by:

\[
\arg\max_{[j]_1^T} S([x]_1^T, [j]_1^T, \theta')
\]  

(8)

This can be computed using dynamic programming, and the Viterbi algorithm [13] is chose for this inference.

### 2.3 Post-Processing

In our method, we employed several common post-processing steps including tagging consistency, abbreviation resolution and bracket balance.

- If the number of a word sequence tagged by our model as a biomedical entity exceeds 50% of the total number of the sequence in a document (title and abstract), all instances of the word sequence will be tagged as an entity. For example, if our model found three chemical entities of “flunarizine hydrochloride” and missed out two other entities of “flunarizine hydrochloride” in a document, the missed entities would be retrieved.

- For abbreviation resolution, all local abbreviation definitions, such as “gamma-aminobutyric acid (GABA)”, will be found. If the abbreviation (i.e. GABA) in the long form was tagged by our model, then all instances of the abbreviation would be tagged in a document.

- While there are some entities with unbalanced brackets (such as parenthesis, square brackets and curly brackets), we attempted to balance the brackets by adding or removing characters to the right or left of the entity. For example, if “A-(X1-N)” (the next characters in the text
are “O2”) was tagged as an entity by our model, then the entity would be extended to include the right parenthesis (i.e. “A-(X1-NO2)”).

2.4 Training Procedure
In our method, the parameters of the model in the word embedding are initialized with pre-trained word embeddings and other parameters are initialized at random from a uniform distribution. Then all parameters are optimized using SGD to maximize the log-probability of the correct tag sequence. In addition, several hyper-parameters need to be determined in our model. We tuned the hyper-parameters on the validation set by random search. Our models are implemented using open-source deep learning library Theano and trained on a NVIDIA Tesla K40 GPU.

3 Results
The organizers of the BioCreative V.5.BeCalm tasks provided a corpus including the training and test sets. The training set comprises a total of 21,000 manually annotated documents (title and abstract), and test set comprises a total of 9,000 unannotated documents. Annotations for the CEMP task are provided in seven classes: systematic, identifiers, formula, trivial, abbreviation, family and multiple. In the case of the GPRO task, the annotations are divided in two groups: type 1, covering GPRO mentions that can be normalized to a database record (including the following classes: nested mentions, identifier, full name and abbreviation); and type 2, covering those GPRO mentions that in principle cannot be normalized to a unique bio-entity database record (including the following classes: no class, sequence, family and multiple).

3.1 CEMP Task Results
In the CEMP task, we randomly selected the 10% of the training set as the development set to tune the hyper-parameters. Although seven annotation classes for chemical entities are provided, discrimination between the classes is not an objective of the task. Therefore, in our method, all classes are grouped into a single one. The results of our submitted runs on our development set and official results of the runs on the test set are shown in Table 1.

Our submitted five runs are based on the following configurations.
Run 1: our BLSM-CRF model with word embedding and char embedding as inputs.

Run 2: using the same features and model as Run 1, but the development set was added into the training set (for the other runs, the development set is not used for training). And according to our experience, we stopped the model training at 20 iterations.

Run 3: like Run 1, but uses the additional chunking feature.

Run 4: like Run 1, but uses the additional POS feature.

Run 5: like Run 1, but all features were used.

The results show that there is no significant difference among F-scores of these runs except Run 2. The reason is that, since the development set was added into the training set, Run 2 was stopped training at 20 iterations according to our experience. Probably we stopped the model training prematurely so that the model parameters could not be optimized. In contrast, the other runs were trained by early stopping strategy [14] on the development set. Therefore, the performance of Run 2 is worse than those of the other runs.

In addition, on both the development and test sets, no significant improvement was observed with the additional features. The plausible reason is that the deep neural network itself has learned sufficient higher and abstract features automatically from the word and character embeddings.

### Table 1. CEMP task results

<table>
<thead>
<tr>
<th>Runs</th>
<th>Development set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>1</td>
<td>87.30</td>
<td>92.28</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>87.21</td>
<td>92.58</td>
</tr>
<tr>
<td>4</td>
<td><strong>88.12</strong></td>
<td>91.7</td>
</tr>
<tr>
<td>5</td>
<td>87.54</td>
<td>91.23</td>
</tr>
</tbody>
</table>

In the GPRO task, only 5,795 documents in the training set contain annotated gene entities and the rest 15,205 documents do not contain
gene entities. In our experiments, to explore the effectiveness of the documents without annotated gene entities, two corpora were used to train the different runs. One consists of the documents data in the original training set (Data\textsubscript{GPRO\_ori}); the other consists of all documents with annotated gene entities and the documents of the same number (5,795) without gene entities in the original training set (Data\textsubscript{GPRO\_balance}). We randomly selected the 10% of the two corpora as the corresponding development set, respectively and the similar models in the CEMP task were employed for the GPRO task. However, we mistakenly thought only entities that can be mapped to an identifier (type 1) are evaluated like the GPRO subtask in the BioCreative V does [4]. Therefore, our final results only provide the type 1 entities, and the type 2 entities are ignored. Finally, five runs based on the following configurations were chosen to submit.

- Run 1: our BLSM-CRF model with word embedding and char embedding as inputs. Type 1 and 2 entities were treated as two distinct classes, and the model was trained with the Data\textsubscript{GPRO\_balance}.
- Run 2: the result was produced by the ensemble of two models. Firstly, model 1 was trained with the Data\textsubscript{GPRO\_ori} using the same features and model as Run 1, but type 1 and 2 entities were treated as a single class. Then the type 2 entities recognized by the model 2 (i.e. the model for Run 3) were removed from all entities recognized by the model 1.
- Run 3: like Run 1, but uses the additional chunking feature.
- Run 4: the same features and model as Run 1, but the model was trained with the Data\textsubscript{GPRO\_ori} and the development set was added. According to our experience, we stopped the model training at 20 iterations.

<table>
<thead>
<tr>
<th>Runs</th>
<th>Results of GPRO type 1 on the development set</th>
<th>Results of GPRO type 1 on the test set</th>
<th>Results of GPRO type 1 and 2 on the test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>67.97</td>
<td>86.06</td>
<td>75.95</td>
</tr>
<tr>
<td>2</td>
<td>64.26</td>
<td>84.65</td>
<td>73.06</td>
</tr>
<tr>
<td>3</td>
<td>63.16</td>
<td>85.69</td>
<td>72.72</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>70.84</td>
<td>83.76</td>
<td>76.76</td>
</tr>
</tbody>
</table>

P denotes precision, R denotes recall and F denotes F-score.
Run 5: like Run 1, but all features were used.

The results of different runs on our development set and official results of all runs on the test set are shown in Table 2. Among others, Run 5 with all features achieves the best performance for GPRO type 1 on the test set. The reason is that the additional linguistic features can help model distinguish type 1 genes from type 2 genes. Unfortunately, because our final results only provide the recognitions of the type 1 entities, the recalls significantly decrease on the test set when both type 1 and type 2 entities are evaluated.

4 Acknowledgment

This work is supported by the grants from the Natural Science Foundation of China (No. 61272373, 61572102 and 61572098), Trans-Century Training Program Foundation for the Talents by the Ministry of Education of China (NCET-13-0084), the Fundamental Research Funds for the Central Universities (No. DUT14YQ213) and the Major State Research Development Program of China (No. 2016YFC0901902).

REFERENCES


HITextracter System for Chemical and Gene/Protein Entity Mention Recognition in Patents

Zengjian Liu, Xiaolong Wang, Buzhou Tang*, Qingcai Chen, Xue Shi, Jiankang Hou

Key Laboratory of Network Oriented Intelligent Computation, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

liuzengjian.hit@gmail.com; wangxl@insun.hit.edu.cn; tangbuzhou@gmail.com; qingcai.chen@gmail.com; 603272564@qq.com; 1207005877@qq.com

Abstract. In this paper, a hybrid system was proposed for chemical entity mention recognition (CEMP) and gene/protein related object recognition (GPRO) in BeCalm challenge. Firstly, five individual machine learning-based subsystems were developed to identify chemical and gene/protein related entity mentions, that is, a bidirectional LSTM (long-short term memory, a variant of recurrent neural network)-based subsystem without any manually-crafted feature, a bidirectional LSTM-based subsystem with some manually-crafted features, a bidirectional LSTM-based subsystem with orthographic features learning, a CRF (conditional random field)-based subsystem and a SSVM (structured support vector machine)-based subsystem. Then, an ensemble learning-based classifier was deployed to combine all the results predicted by above individual subsystems. Evaluation on the official test set showed that the best F1-scores achieved by our system are 90.37% on CEMP, 76.34% on CPRO type 1 respectively.

Keywords. Chemical entity mention recognition; gene and protein related object recognition; sequence labeling problem; conditional random fields; recurrent neural network; ensemble learning

1 Introduction

Chemical patents contain a wealth of chemical and biochemical knowledge, such as chemical compounds, genes and proteins. The BioCreative V challenge [1] has aimed to evaluate and encourage the development of tools to extract these information from patents. To enable a more robust evaluation platform (Biomedical Annotation Metaserver, BeCalm), the BioCreative V.5. BeCalm. challenge also was organized

* * Corresponding author
with three tasks: CEMP (Chemical Entity Mention recognition), GPRO (Gene and Protein Related Object recognition), and TIPS (Technical interoperability and performance of annotation servers). We participated in the CEMP and GPRO tasks, and developed a hybrid system based on five individual entity recognition methods.

2 Methods

Dataset

The BeCalm challenge organizers provided total 30,000 manually annotated patents for the CEMP and GPRO tasks, 21,000 out of which are used as a training set and the remaining 9,000 as a test set. In the training set, 99,632 chemical entity mentions and 17,751 gene/protein related objects were annotated. Table 1 shows the numbers of instances of each type in both CEMP and GPRO tasks.

Table 1. Numbers of instances of each type in CEMP and GPRO tasks.

<table>
<thead>
<tr>
<th>Type</th>
<th>CEMP Number</th>
<th>Type 1</th>
<th>GPRO Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAMILY</td>
<td>36,238</td>
<td>ABBREVIATION</td>
<td>7,543</td>
</tr>
<tr>
<td>SYSTEMATIC</td>
<td>28,580</td>
<td>FAMILY</td>
<td>5,030</td>
</tr>
<tr>
<td>TRIVIAL</td>
<td>25,927</td>
<td>FULL_NAME</td>
<td>4,842</td>
</tr>
<tr>
<td>FORMULA</td>
<td>6,818</td>
<td>MULTIPLE</td>
<td>178</td>
</tr>
<tr>
<td>ABBREVIATION</td>
<td>1,373</td>
<td>NESTED</td>
<td>89</td>
</tr>
<tr>
<td>MULTIPLE</td>
<td>418</td>
<td>NO_CLASS</td>
<td>45</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>278</td>
<td>SEQUENCE</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>99,632</td>
<td>IDENTIFIER</td>
<td>1</td>
</tr>
</tbody>
</table>

Overview of system

Our system, as shown in Figure 1, consists of seven components: a tokenization module, five individual modules for chemical and gene/protein entity mention recognition respectively, and an ensemble module to combine all results of above individual modules. Given a record with title and abstract, the tokenization module first split each
sentence into tokens. Then, five individual methods were used to identify the chemical and gene/protein entity mentions. Subsequently, the ensemble module used a stacked ensemble learning-based classifier to combine the results of all above individual modules. We will introduce these core modules of our system in the next few sections.

Figure 1. Overview architecture of our system.

**Tokenization**

After the analysis of raw text, we found that there are lots of unexpected phrases in patents for the existing tokenization tools (e.g. MedEx), for example, the phrase "N-(4-hydrosulfonimidoylphenyl)-5-(trifluorometh yl)pyrimidin-2-amine" is hard to be tokenized. Therefore, in our system for the BeCalm challenge, we employed a new tokenization module. Firstly, we split sentences into tokens by blank spaces, then further separate consecutive numbers, consecutive letters and other characters. For example, above phrase was tokenized into "N - ( 4 - hy drosulfonimidoylphenyl ) - 5 - ( trifluorometh yl ) pyrimidin - 2 - amine". This method can effectively avoid the boundary errors between predicted instances and gold instances caused by improper tokenization module.
CRF&SSVM-based methods

As our previous work [2], we proposed a CRF-based and a SSVM-based methods for the recognition of chemical and gene/protein entity mentions with above tokenization module. The same set of features were used in both these methods, which are listed in Table 2. We use “CRFsuite” [3] as the implementation of CRFs, and “hmm-svm” [4] as the implementation of SSVM.

Table 2. The features used in our CRF&SSVM-based methods.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag-of-words</td>
<td>The unigrams, bigrams and trigrams of words within a window of [-2, 2].</td>
</tr>
<tr>
<td>Part-of-speech (POS) tags</td>
<td>The POS unigrams, bigrams and trigrams within a window of [-2, 2]. We use GENIA tagger for the POS tagging.</td>
</tr>
<tr>
<td>Combinations of words and POSs</td>
<td>{w_0p_{-1}, w_0p_0, w_0p_1, w_0p_{-1}p_0, w_0p_{-1}p_1, w_0p_0p_1}, where w_0 denotes the current word, and p_{-1}, p_0 and p_1 denote the last, current and next POS tags respectively.</td>
</tr>
<tr>
<td>Sentence features</td>
<td>The number of words in the sentence, whether there is an end mark at the end of the sentence such as '.', '?' and '!', whether there is any bracket unmatched in the sentence.</td>
</tr>
<tr>
<td>Semantic features</td>
<td>Whether the current token contains alkane stems (e.g. “meth,” “eth”, “prop” and “tetra”), trivial rings (e.g. “benzene”, “pyridine” and “toluene”), and simple multipliers (“di”, “tri” and “tetra”), as mentioned in [1].</td>
</tr>
<tr>
<td>Affixes</td>
<td>All prefixes and suffixes of length from 1 to 5.</td>
</tr>
<tr>
<td>Section features</td>
<td>Which section the token belongs to, title or abstract?</td>
</tr>
<tr>
<td>Word Shapes</td>
<td>Any or consecutive uppercase character(s), lowercase character(s), digit(s) and other character(s) in the current word is/are replaced by ‘A’, ‘a’, ‘#’ and ‘-’ respectively.</td>
</tr>
<tr>
<td>Orthographical features</td>
<td>Whether the word is upper case, has uppercase characters inside, has punctuation marks inside, has digit inside, the word is Roman or Arabic number, etc.</td>
</tr>
<tr>
<td>Domain knowledge</td>
<td>Whether the current token contains any prefix/suffix of chemical compounds, drugs, proteins, etc.</td>
</tr>
<tr>
<td>Character features</td>
<td>Number of characters, number of digits, number of uppercase and lowercase letters and number of lowercase letters.</td>
</tr>
<tr>
<td>Character n-grams</td>
<td>Character n-grams of length from 2 to 4.</td>
</tr>
</tbody>
</table>
Bidirectional LSTM method

A bidirectional LSTM, which has been successfully applied in several sequence labeling tasks [5-7], was deployed for the CEMP and GPRO tasks. It contains three main layers: 1) input layer, which generates the representation of each word in a sentence, and contains two parts: character-level representation and token-level representation; 2) LSTM layer, which includes a forward LSTM and a backward LSTM, takes the word representation sequence of a sentence as input, and outputs a new word representation sequence that captures the context information of each word in this sentence; 3) inference layer, which captures the dependencies between successive labels by keeping a label transition matrix, and predicts the best label sequences with correct structures. The architecture of our BI-LSTM is same as Lample’s (2016) [6] for name entity recognition.

Bidirectional LSTM with Feature

In order to incorporate some significant features, we extended the above BI-LSTM model by adding a hidden layer after the LSTM layer [7, 8], which concatenates the word representations generated by LSTM layer and the feature representations together. The features used in the BI-LSTM with feature model are: POS tags, sentence features, semantic features, section information, and domain knowledge features, which are same as the features used in the CRF-based method.

Bidirectional LSTM with Orthographic features learning

To further capture the orthographic information of tokens, we extended the inputs of our BI-LSTM model refer to [9]. Firstly, the orthographic feature of each token was generated by mapping any uppercase character, lowercase character, digit and other character in the word to ‘A’, ‘a’, ‘#’ and ‘-’ respectively. For example, the orthographic feature of “1-6C-alkyl” is “#-#A-aaaaa”. Then, as the word representations, we also generated the token-level and character-level representations of orthographic features, and concatenated them with the word representations together as the inputs of our BI-LSTM model.

Ensemble Learning Method

To take full advantages of above individual methods, we used an ensemble learning method [10], support vector machine, to merge all results of them. The goal of the ensemble learning method is to determine
whether a predicted CEMP/GPRO instance is a true instance, and the features used in this method includes

- Whether the text spans of a instance exactly match with others?
- Whether the text spans of a instance exactly match with others of the same type?
- Whether the text spans of a instance partially match with others?
- Whether the text spans of a instance partially match with others of the same type?
- Whether a instance contains a conjunction or preposition?
- Which methods have predicted current instance?
- How many times a instance was predicted?
- How many times the span of a instance was predicted?
- The number of tokens in a instances.
- How many times a instance was predicted in same patent?

3 Results

In the BeCalm challenge, we were allowed to submit five runs for CEMP and GPRO tasks respectively. The results of these different runs we submit are listed in Table 3, where “Ensemble-RNN” means combining the results of three RNN-based methods, and “Ensemble-ALL” combines the results of all five methods. The best micro F1-scores achieved by our system are 90.37% on CEMP, 76.34% on CPRO type 1 respectively.

Table 3. The results of our systems for both CEMP and GPRO tasks.

<table>
<thead>
<tr>
<th>Run</th>
<th>CEMP</th>
<th>GPRO type 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre.</td>
<td>Rec.</td>
</tr>
<tr>
<td>BI-LSTM</td>
<td>88.97</td>
<td>91.82</td>
</tr>
<tr>
<td>BI-LSTM with feature</td>
<td>88.70</td>
<td>91.28</td>
</tr>
<tr>
<td>BI-LSTM with orthographic</td>
<td>88.91</td>
<td>91.28</td>
</tr>
<tr>
<td>Ensemble-RNN</td>
<td>91.25</td>
<td>88.02</td>
</tr>
<tr>
<td>Ensemble-ALL</td>
<td>91.42</td>
<td>88.56</td>
</tr>
</tbody>
</table>
4 Acknowledgment

This paper is supported in part by grants: National 863 Program of China (2015AAA015405), NSFCs (National Natural Science Foundations of China) (61573118, 61402128, 61473101, 61472428), Special Foundation for Technology Research Program of Guangdong Province (2015B010131010), Strategic Emerging Industry Development Special Funds of Shenzhen (JCYJ20140508161040764, 20151013161937, JCYJ20140417172417105, JCYJ20140627163809422, JSGG20151015161015297 and JCYJ20160531192358466), Innovation Fund of Harbin Institute of Technology (HIT.NSRIF.2017052), Program from the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (93K172016K12) and CCF-Tencent Open Research Fund (RAGR20160102).

REFERENCES


CRFVoter: Chemical Entity Mention, Gene and Protein Related Object recognition using a conglomerate of CRF based tools

Wahed Hemati, Alexander Mehler, and Tolga Uslu

Text Technology Lab,
Goethe Universitt Frankfurt am Main, Germany
{hemati,amehler,uslu}@em.uni-frankfurt.de
http://www.hucompute.org/

Abstract. This paper relates to the two offline BioCreative V.5 Becalm tasks. The first challenge is CEMP, the recognition of chemical named entity mentions. The second challenge is GPRO, the recognition of gene and protein related objects in running text. We focus on training and optimizing state-of-the-art solutions for named entity tagging for CEMP and GPRO. Finally, we present CRFVoter, a two staged application of CRF.

Key words: Biocreative, BeCalm, Chemical named entity recognition, Named Entity Recognition, CRF

1 Introduction

BioCreative V.5 consists of two offline tasks, namely CEMP (Chemical Entity Mention Recognition) and GPRO (Gene and Protein Related Object Recognition). CEMP requires the detection of chemical named entity mentions. The task requires detecting correctly the start and end indices corresponding to chemical entities. GPRO task requires identifying mentions of gene and protein related objects mentioned in patent titles and abstracts. In this work, we survey Named Entity Recognition techniques, which is an abstraction of the CEMP and GPRO task. Our survey includes 5 state-of-the-art NER systems and two combination techniques for these systems, namely Majority vote and CRFVoter.

2 Corpus

The organizers of BioCreative V.5 provided a corpus of 21000 patent abstracts (titles and abstracts in English) from patents published between 2005 and 2014. The corpus is manually annotated for the CEMP and GPRO tasks. For our experiments we divided the corpus in 60% training set, 25% development set and 15% test set by means of random sampling. We applied multiple preprocessing steps on each set including sentence splitting, tokenization, lemmatization and fine-grained morphological tagging. These and related information units were used as features in our experiments.
Table 1. Differences of labeled output between each pair of NER system.

<table>
<thead>
<tr>
<th></th>
<th>Stanford</th>
<th>MarMoT</th>
<th>CRF++</th>
<th>MITIE</th>
<th>Glample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>0</td>
<td>2.29%</td>
<td>2.12%</td>
<td>2.44%</td>
<td>2.50%</td>
</tr>
<tr>
<td>MarMoT</td>
<td>0</td>
<td>2.56%</td>
<td>2.61%</td>
<td>2.43%</td>
<td></td>
</tr>
<tr>
<td>CRF++</td>
<td>0</td>
<td>0</td>
<td>2.91%</td>
<td>2.47%</td>
<td></td>
</tr>
<tr>
<td>MITIE</td>
<td></td>
<td></td>
<td>2.51%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glample</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 System Description

In this section, we present a survey of Named Entity Recognizer trained for the CEMP and GPRO tasks. For each NER we optimized the hyperparameter settings. Hyperparameter tuning is a challenging topic in Machine Learning (ML). The optimal set of hyperparameters depends on the model, dataset and the domain. To this end, we focused in our experiments on optimizing hyperparameter, which lead to a noticeable increase of F-score compared to default settings. For each NER, we used grid search on a set of configurations of hyperparameter and trained them accordingly, choosing the hyperparameter configuration that gives the best performance. The big downside of optimizing hyperparameter is to overfit the model on training data. Each NER classifies a different subset correctly. Table 1 shows the pairwise differences between NER systems. Therefore, a combination of these NER was seemingly promising in order to increase precision and recall, due to possible orthogonal output labels. To this end, we experimented with a simple majority vote. Further more, we developed a two-stage application of CRF for combinations of sequence labeling tools, called CRFVoter. We trained each NER system on the training set and tested against the test set (see Section 2). In this work, we consider the NER systems as enumerated in Table 1 and described in the following subsections.

3.1 Stanford Named Entity Recognizer

Stanford Named Entity Recognizer\(^1\) is a Java implementation of a CRF based Named Entity Recognizer [1]. Table 2 shows the hyperparameter space used in our experiments. The combination of parameters results in 432 model files. The best performing set of features for GPRO, marked with ♦, leads to an F-score of 0.82. The worst setting results in an F-score of 0.73. The best performing feature set for CEMP is marked with ♣ and produces an F-score of 0.825; the worst setting results in 0.74.

\(^1\) [http://nlp.stanford.edu/software/CRF-NER.shtml](http://nlp.stanford.edu/software/CRF-NER.shtml)
Table 2. Parameter Space of Stanford Named Entity Recognizer.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use n-Grams</td>
<td>[true, false]</td>
</tr>
<tr>
<td>No mid-n-Grams</td>
<td>[true, false]</td>
</tr>
<tr>
<td>Use Disjunctive</td>
<td>[true, false]</td>
</tr>
<tr>
<td>Use Type Sequences</td>
<td>[true, false]</td>
</tr>
<tr>
<td>Max Left</td>
<td>[1, 2, 3]</td>
</tr>
<tr>
<td>Max Right</td>
<td>[1, 2, 3]</td>
</tr>
<tr>
<td>Max N-Gram Length</td>
<td>[2, 4, 6]</td>
</tr>
</tbody>
</table>

Combination count: 432

3.2 MarMoT

MarMoT is a generic CRF framework [2]. MarMoT implements a higher order CRF with approximations such that it can deal with large output spaces. Additionally it can be trained to fire on the predictions of lexical resources (gazette files) and on word embeddings [2]. Table 3 shows the hyperparameter space used in our experiments for MarMoT. The combination of parameters results in 3888 model files. The best performing set of features for GPRO is marked with ♦ and produces an F-score of 0.72. The worst set results in an F-score of 0.59. The best performing set of features for CEMP is marked with ♣ and generates an F-score of 0.85. The worst set results in a F-score of 0.61.

Table 3. Parameter Space of MarMoT.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num iterations</td>
<td>[10, 20]</td>
</tr>
<tr>
<td>Penalty</td>
<td>[0, 1, 2]</td>
</tr>
<tr>
<td>Beam size</td>
<td>[1, 2, 5]</td>
</tr>
<tr>
<td>Quadratic penalty</td>
<td>[0, 1, 2]</td>
</tr>
<tr>
<td>Order</td>
<td>[1, 2, 3, 4]</td>
</tr>
<tr>
<td>Prob threshold</td>
<td>[0.01, 0.001]</td>
</tr>
<tr>
<td>Effective order</td>
<td>[1, 2, 3]</td>
</tr>
<tr>
<td>Num chunks</td>
<td>[2, 5, 10]</td>
</tr>
</tbody>
</table>

Combination count: 3888

3.3 CRF++

CRF++ is a customizable open source implementation of CRF [3]. In our experiments we used unigram and bigram features, containing the current, previous

\footnote{http://cistern.cis.lmu.de/marmot/}

\footnote{http://taku910.github.io/crfpp/}
and the next word. Table 4 shows the hyperparameter space used in our experiments for CRF++. The combination of parameters results in 20 model files. The best performing set of parameters for GPRO is marked with ♣ and generates an F-score of 0.69. The worst set results in an F-score of 0.04. The best performing set of parameters for CEMP is marked with ♦ producing an F-score of 0.73, while the worst setting results in an F-score of 0.42.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>[0.6, 1, 1.6, 3, 5, 7, 15♣, 50, 100, 1000]</td>
</tr>
<tr>
<td>a</td>
<td>[CRF-L1, CRF-L2♣]</td>
</tr>
</tbody>
</table>

Combination count: 20

### 3.4 MITIE

MITIE is an open source information extraction tool. MITIE can be trained using techniques like distributional word embeddings and *Structural Support Vector Machines* [4]. Due to the lack of documentation, we did not optimize MITIE. The default configuration for named entity recognition produces an F-score of 0.65 for GPRO and 0.62 for CEMP.

### 3.5 Glample NER Tagger

Glample NER Tagger is a neural-network-based named entity recognizer. It is based on Bidirectional LSTM and CRF[5]. Due to the long-lasting training time, only the default parameter settings were considered. This resulted in an F-score of 0.75 for GPRO and 0.77 for CEMP.

### 3.6 Majority Vote

By means of majority voting, we combined the best performing outputs of each of the NER systems considered so far. We selected the label that was most frequently output by the different NER systems. Majority voting reaches an F-score of 0.71 for GPRO, which is below the best performing system considered so far. For CEMP majority voting results in an F-score of 0.78. Facing these results we can state that a simple majority vote brings no gain in precision and recall.
3.7 CRFVoter

Since majority voting did not better F-score, we developed the so-called CRFVoter, that is, a two-stage CRF-system for combining different sequence labeling systems. In the first stage each NER is optimized independently (see Section 3) on the trainings set. In the second stage, the development set (see Section 2) is tagged by each NER independently. The output label of each NER system is taken as individual feature for CRFVoter. Figure 1 exemplifies CRFVoter on the input stream

“Inhibitors of D-amino acid oxidase . . .”

For each token of this stream, the corresponding labels are calculated by the NER systems of Section 3. In the second stage, the output labels of each NER system are taken as individual features for a CRF operating on the latter system’s output. The CRFVoter trains a model based on these features. For tagging, CRFVoter takes again the output of each NER system as features and labels the sequence by means of the 2nd-stage CRF. In the example of Figure 1, majority voting would tag the sequence wrongly. On the other hand, CRFVoter learned the correct sequence of labels. CRFVoter achieved an F-score of 0.84 on GPRO and 0.88 on CEMP – both outcomes are better than any of the best performing NER systems documented in Section 3.

4 Results

Table 5 shows the comparison of annotators trained for GPRO and Table 6 considers the corresponding results with respect to CEMP. The best performing annotator is CRFVoter in both tasks when being tested on the test set described in Section 2. On the blinded test set for CEMP provided by the Biocreative team, the best performing system, which is again CRFVoter, reaches an F-score of 0.87. For the blinded test set provided for GPRO, it reaches an F-score of 0.75.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford NER</td>
<td>0.83</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>MarMoT</td>
<td>0.76</td>
<td>0.69</td>
<td>0.72</td>
</tr>
<tr>
<td>CRF++</td>
<td>0.75</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>MITIE</td>
<td>0.74</td>
<td>0.58</td>
<td>0.65</td>
</tr>
<tr>
<td>Glample</td>
<td>0.79</td>
<td>0.72</td>
<td>0.75</td>
</tr>
<tr>
<td>Majority Vote</td>
<td>0.72</td>
<td>0.71</td>
<td>0.72</td>
</tr>
<tr>
<td>CRFVoter</td>
<td>0.85</td>
<td>0.84</td>
<td>0.84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford NER</td>
<td>0.85</td>
<td>0.80</td>
<td>0.82</td>
</tr>
<tr>
<td>MarMoT</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>CRF++</td>
<td>0.77</td>
<td>0.73</td>
<td>0.73</td>
</tr>
<tr>
<td>MITIE</td>
<td>0.62</td>
<td>0.61</td>
<td>0.62</td>
</tr>
<tr>
<td>Glample</td>
<td>0.76</td>
<td>0.79</td>
<td>0.77</td>
</tr>
<tr>
<td>Majority Vote</td>
<td>0.78</td>
<td>0.79</td>
<td>0.78</td>
</tr>
<tr>
<td>CRFVoter</td>
<td>0.88</td>
<td>0.87</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Fig. 1. Architecture of CRFVoter exemplified by means of a single sentence.

Inhibitors of D- amino acid oxidase ...
5 Discussion and future work

In this work, we compared a set of NER systems. We trained and optimized every NER system for GPRO and CEMP by means of hyperparameter optimization. We showed that optimizing hyperparameter can be crucial. One NER system in our experiments gained an improvement of more than 60%. In future work, a hyperparameter search algorithm, which is less time-consuming than grid search, will be implemented, for instance, random search or Bayesian optimization. A bigger hyperparameter space can then be searched and also non-trivial values for continuous variables can be optimized. We additionally introduced and evaluated the so-called CRFVoter, a two-stage CRF tool for combining underlying sequence modeling tools (as given by the NER of our comparative study). CRFVoter gained 2% improvement compared to the best performing reference systems being examined in our study. Thus, CRFVoter may be further-developed by feeding it with the output of additional sequence labeling systems. This will be the second part of our future work.

6 Acknowledgment

We gratefully acknowledge support by the Deutsche Forschungsgemeinschaft via the Specialised Information Services Biodiversity Research.

References

Neji: Recognition of Chemical and Gene Mentions in Patent Texts

André Santos and Sérgio Matos

DETI/IEETA, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{andre.jeronimo, aleixomatos}@ua.pt
http://bioinformatics.ua.pt/

Abstract. The BioCreative V.5 challenge focused on the recognition of chemicals and gene mentions in medicinal chemistry patents. For participation in the chemical entity (CEMP) and gene and protein (GPRO) recognition tasks, we used the concept recognition framework Neji and applied a machine-learning strategy using an optimized feature set. Our best submissions achieved an F-score of 86.6% for the identification of chemicals and 71.3% for the identification of gene names.

Key words: Text mining, Patents, Named entity recognition, Chemicals, Genes

1 Introduction

The BioCreative V.5 text mining challenge\(^1\) focused on the development and evaluation of information extraction systems for recognition of chemical and gene entity mentions in chemistry patents [1]. Two offline sub-tasks were considered: recognition of chemical entity mentions (CEMP) and recognition of gene and protein related objects (GPRO). For each sub-task, systems should identify all mentions of entities of the corresponding types in free-text and return the start and end indices of the text span.

We followed a machine-learning approach, using conditional random fields (CRF) models. We took advantage of the provided training and development corpora for performing feature selection and for identifying white and black lists to use in post-processing steps.

2 System description

We used Neji\(^2\), a flexible and extensible concept recognition framework specially optimized for biomedical text [2]. Neji’s architecture is illustrated in Figure 1,

\(^1\) http://www.becalm.eu/pages/biocreative
\(^2\) Available from https://github.com/BMDSoftware/neji

---

\(^*\) Corresponding author
Fig. 1. Neji processing pipeline and architecture.

showing the various processing stages and input (Reader) and output (Writer) modules that provide support for a variety of formats, as well as allowing easy customization for new formats. The processing modules are managed through an efficient pipeline with multi-threading support. Neji includes natural language processing modules, based on GDep [14] and Apache OpenNLP 3, concept recognition modules based on dictionary matching and machine learning, and post-processing modules, including parentheses correction and abbreviation resolution. The machine learning component is based on Gimli [3], and makes use of the MALLET [12] implementation of Conditional Random Fields (CRFs) models [10]. This module provides simple methods for feature extraction and for training and applying CRF models for entity recognition.

We applied a machine learning (ML) approach, combined with dictionary-matching for obtaining lexicon features, as described in [4].

2.1 Corpora

The BioCreative V.5 corpus is composed of 21 thousand manually annotated patents (title and abstract), available for training the systems, and a further nine thousand patents used for evaluation. As with the previous BioCreative V CHEMDNER Patents task [8], chemical entities are annotated in seven classes: systematic, identifiers, formula, trivial abbreviation, family and multiple. We however grouped all mentions into a single class. Gene and protein related object (GPROs) annotations were divided into type 1, covering GPRO mentions that can be normalized to a database record; and type 2, including mentions that can not be normalized. We considered only identification of type 1 mentions.

3 https://opennlp.apache.org/
The training set contains a total of 99634 chemical mentions and 17751 gene mentions (12422 of which of type 1).

In order to augment the available training data, we included the BioCreative IV CHEMDNER corpus, containing 10000 PubMed abstracts and a total of 84355 chemical entity mentions [9], which we used for training the recognition model for the CEMP sub-task, and the BioCreative II gene mention corpus, containing 20000 sentences from Pubmed abstracts with around 44500 gene mention annotations [16], which we used for training the GPRO task model.

Furthermore, to account for the expected differences between the patent and literature documents, as well as within corpus heterogeneity, we clustered the documents and created different recognition models for each cluster. For this, we applied bi-clustering, as provided in the scikit-learn machine learning library for Python [13]. For the CEMP task, we combined 14 thousand patent documents from the BioCreative V.5 corpus, corresponding to the training and development sets of the BioCreative V task, with the 10 thousand documents from the BioCreative IV corpus, and obtained three clusters with 9032, 4622 and 10346 documents each. After the internal evaluation stage, we included the remaining 7000 documents from BioCreative V.5 and re-created the clusters, obtaining 10758, 6670 and 13572 for each cluster. For the GPRO task, we combined 14 thousand patents with the 20000 sentence from BioCreative II and generated clusters containing 18616, 4839 and 10545 documents. Similarly, after the internal evaluation phase we included the remaining documents and obtained clusters with 22165, 5771 and 13064 documents. Table 1 shows the distribution of patent and literature documents across the clusters.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>CEMP test set removed patents</th>
<th>CEMP test set included patents</th>
<th>GPRO test set removed patents</th>
<th>GPRO test set included patents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1776</td>
<td>726</td>
<td>5021</td>
<td>5737</td>
</tr>
<tr>
<td>2</td>
<td>4516</td>
<td>106</td>
<td>6540</td>
<td>130</td>
</tr>
<tr>
<td>3</td>
<td>7708</td>
<td>2638</td>
<td>9439</td>
<td>4133</td>
</tr>
</tbody>
</table>

2.2 Pre-processing

Sentence splitting was performed with Lingpipe\(^4\). Tokenization, lemmatization, part-of-speech (POS) tagging, chunking and dependency parsing were performed using Neji’s custom version of GDep [3]. The BIO scheme was used for encoding the annotations.

\(^4\)http://alias-i.com/lingpipe
2.3 Model and feature selection

We performed recursive feature elimination by training on the 7000 documents that compose the training set of BioCreative V, and testing on the development set. Together with this feature selection step, we tested CRF models with orders 1 and 2 and with forward (from left to right) and backward (from right to left) parsing directions.

2.4 Post-processing

Exclusion and inclusion lists were generated by analyzing the false-positive and false-negative mentions, respectively, obtained on the 7000 documents used for internal evaluation.

2.5 Ranking

To score and rank the annotations, we used the confidence scores provided by the CRF models, which is a value between 0 and 1 that reflects the certainty of the model generating each annotation.

3 Results and Discussion

Table 2 shows the feature sets that originated the best results for each task, based on cross-validation over the 14 thousand documents used for development. Interestingly, the results for the CEMP task improved with the inclusion of gene lexicon features, in addition to the chemical lexicon. The reverse was also true for the GPRO task. The best results were obtained using a first order CRF with backward parsing for the CEMP task, and a second order CRF with forward parsing for the GPRO task.

Tables 3 and 4 describe the submitted runs for participation in the CEMP and GPRO tasks, respectively, the internal evaluation results obtained on the BioCreative V CHEMDNER Patents test set, and official results on the BioCreative V.5 test set.

The results show that the post-processing stage improved considerably the results during internal evaluation, but this improvement was not replicated on the final test set. The use of distinct models trained on clustered documents originated slight improvements in recall in the CEMP task, and a large improvement also in recall in the GPRO task. These improvements were however balanced by significant reductions in precision, leading to worst overall results for CEMP and slightly better result for the GPRO task.

Following the challenge, we performed feature selection separately for each cluster of the CEMP task. For this, we divided the documents in each cluster in two groups and evaluated the impact of each feature by cross-testing with these two groups and taking the average f-score. With this strategy, we obtained three models with different feature sets which were then applied to the 7000
Table 2. Features used for training recognition models for each task.

<table>
<thead>
<tr>
<th>Group</th>
<th>Feature</th>
<th>CEMP</th>
<th>GPRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>Token</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Stem</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lemma</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>POS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Chunk tags</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Dependency parsing</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Orthographic</td>
<td>Capitalization (e.g., “InitCap”, “AllCaps”)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Digits and capitalized characters counting</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>(e.g., “TwoDigit”, “TwoCap”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Symbols (e.g., “Dash”, “Dot”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greek letters (e.g., “α”)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Roman digits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morphological</td>
<td>Prefixes and suffixes</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Character n-grams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexicons</td>
<td>Word shape features to reflect how letters,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>digits and symbols are organized in the token</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(e.g., the structure of “Abc:1234” is expressed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>as “Aaa#1111”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chemical entity names from Jochem [7], ChEBI [6]</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>and CTD [5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gene names from BioThesaurus [11]</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Trigger words from BioLexicon [15]</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Local context</td>
<td>Conjunctions of lemma and POS features of the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>windows {-1, 0}, {-2, -1}, {0, 1}, {-1, 1} and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{-3, -1}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Runs submitted to the CEMP sub-task.

<table>
<thead>
<tr>
<th>run</th>
<th>Description</th>
<th>BC V test set</th>
<th>BC V.5 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 model trained with 21k BioCreative V.5 training data</td>
<td>86.2 86.2 86.2</td>
<td>89.0 84.3 86.6</td>
</tr>
<tr>
<td>2</td>
<td>as above, plus FP filtering</td>
<td>89.2 86.2 87.7</td>
<td>89.3 84.0 86.6</td>
</tr>
<tr>
<td>3</td>
<td>clusters 1, 2, 3</td>
<td>83.4 79.6 81.4</td>
<td>84.6 69.4 85.6</td>
</tr>
<tr>
<td>4</td>
<td>BC V + BC IV</td>
<td>82.4 89.2 85.7</td>
<td>85.1 86.6 85.8</td>
</tr>
<tr>
<td>5</td>
<td>clusters 2, 3</td>
<td>81.9 88.7 85.2</td>
<td>85.7 85.8 85.8</td>
</tr>
</tbody>
</table>

Table 4. Runs submitted to the GPRO sub-task.

<table>
<thead>
<tr>
<th>run</th>
<th>Description</th>
<th>BC V test set</th>
<th>BC V.5 test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 model trained with 21k BioCreative V.5 training data</td>
<td>71.2 62.9 66.8</td>
<td>78.8 62.0 69.4</td>
</tr>
<tr>
<td>2</td>
<td>as above, plus FN/FP filtering</td>
<td>77.3 77.0 77.1</td>
<td>77.4 61.9 68.9</td>
</tr>
<tr>
<td>3</td>
<td>clusters 1, 2, 3</td>
<td>47.2 73.3 57.4</td>
<td>68.8 65.8 67.3</td>
</tr>
<tr>
<td>4</td>
<td>BC V + BC II</td>
<td>34.1 76.6 47.2</td>
<td>61.6 71.0 66.0</td>
</tr>
<tr>
<td>5</td>
<td>clusters 1, 3</td>
<td>50.6 73.0 59.8</td>
<td>71.5 71.1 71.3</td>
</tr>
</tbody>
</table>
documents in the BioCreative V CHEMDNER Patents test set. Using intersection, we achieved an f-score of 86.1, a precision of 77.0 and a recall of 97.8. This was improved to an f-score of 86.9, precision of 78.8 and recall of 96.9 by using false-positives filtering as in the submitted run 2.

Acknowledgments This work was supported by Portuguese National Funds through FCT - Foundation for Science and Technology, in the context of the project IF/01694/2013. Sérgio Matos is funded under the FCT Investigator programme.

References


Chemlistem - chemical named entity recognition using recurrent neural networks

P.Corbett¹, J.Boyle²

Data Science Group, Technology Dept, The Royal Society of Chemistry.

¹pcorbett@rsc.org; ²boylej@rsc.org

Abstract. Chemical named entity recognition has traditionally been dominated by CRF (Conditional Random Fields)-based approaches but given the success of the artificial neural network techniques known as “deep learning” we decided to examine them as an alternative to CRFs. We present here three systems. The first system translates the traditional CRF-based idioms into a deep learning framework, using rich per-token features and neural word embeddings, and producing a sequence of tags using bidirectional Long Short Term Memory (LSTM) networks – a type of recurrent neural net. The second system eschews the rich feature set – and even tokenisation – in favour of character labelling using neural character embeddings and multiple LSTM layers. The third system is an ensemble that combines the results of the first two systems, achieving an F score of 0.9032 on the test data (precision 0.9002, recall 0.9062).

Keywords. Chemicals, Named Entity Recognition, Deep Learning.

1 Introduction

At the Royal Society of Chemistry the data science group undertakes a variety of text mining data to enrich both our data offerings and our corpus. One common task is chemical named entity recognition, and the group has spent considerable time applying different machine learning algorithms to extract such information. This paper discusses one of these approaches, which uses structured deep learning.

The CEMP (Chemical Entity Mention recognition) task of BioCreative V.5 [1] addresses recognition of chemical named entities in patent text, using a training set of 21,000 patent abstracts and a test set of 9,000 patent abstracts. The corresponding task in BioCreative V [2] was dominated by systems employing Conditional Random Fields (CRF) – there were two rule-based non-CRF systems but no other
methods employing non-CRF machine learning approaches to the sequence labeling problem. CRF-based systems, such as the highly successful tmChem system [3] treat a sentence or paragraph as a sequence of tokens, and assign a tag to each token to indicate whether it is part of a chemical name, and its position in the name.

The recent resurgence of artificial neural network techniques known as "deep learning" [4] suggest that these may provide an alternative or a complement to CRFs. Recurrent neural networks offer an approach to sequence labeling, a common approach to natural language processing (NLP) tasks such as part-of-speech (POS) tagging and named entity recognition. One type of network – a variety of Long Short-Term Memory (LSTM) known as a Bidirectional LSTM has achieved state-of-the-art performance on common NLP tasks [5]. In this paper we demonstrate how Bidirectional LSTMs, implemented using the Keras toolkit [6], can be applied to chemical named entity recognition.

Here we discuss two different approaches to LSTM-based chemical named entity recognition, and an ensemble system that combines both. The first system – the "traditional" system - works similarly to traditional CRF-based systems, in that it assigns tags to a sequence of tokens, each token bearing features from a rich feature set. Our “traditional” system differs from those that are CRF-based in a number of ways – for example, our traditional system supplements the feature set with neural word embeddings, and (with a few minor exceptions) does not include information about neighboring tokens in the feature set, instead relying on the neural network structure to carry the information from neighboring tokens to the right place.

The second system – the "minimalist" system - labels a sequence of characters, rather than words (i.e. it does not use a tokeniser), and does not use a rich feature set, instead using character embeddings and multiple LSTM layers in order to induce the equivalent of a feature set internally. In related work, character embeddings have been used in domains where word segmentation is difficult, for example Chinese NLP [7] and text containing programming language snippets [8] – suggesting that this may be particularly suitable for chemical text, where tokenization presents particular difficulties. Finally, the ensemble system examines to what extent the two approaches are complementary.
2 System description and methods

For each of our approaches there was a three step process, involving pre-processing, a neural network step, and finally post-processing.

2.1 Pre-processing

Tokenisation in the traditional system was performed using a modified version of a Python translation of the Oscar4 tokeniser [9]. On the training data only, when an entity boundary was in the middle of a token, the token was split in two. The minimalist system does not use tokenization as such – however it is equivalent to ‘tokenizing’ the data to ‘tokens’ a single character long (including whitespace characters). Tokens (or in the minimalist system, characters) in the training data were assigned SOBIE (sometimes known as BIOES) tags – "O" marking a token not part of an entity, "S" marking a token that is the whole of an entity (a "singleton"), "B" marking a token at the beginning of an entity, "I" marking one inside an entity, and "E" marking one at the end.

For both systems the data was split 80:20 for training and testing.

The traditional system starts with finding those tokens in the corpus that occur more than two times, and assigning initial embedding vectors based on GloVe (a set of pre-trained word vectors based on Wikipedia) [10] – tokens not found in GloVe are given initial embedding vectors full of zeros. Tokens that occur two times or less are all given a single "unknown token" vector, again initialized to zeros.

The traditional system uses a "preclassifier" [11] to judge how likely a token is to be chemical – i.e. assigned an S, B, I or E tag as opposed to O. To train this, the system finds tokens only ever tagged O or only ever tagged SBIE, generates binary features for each of these, selects the 1000 binary features with highest mutual information with O-only vs SBIE-only, and uses those to train a random forest (using scikit-learn [12]) with 100 trees. This is the "preclassifier", and is used for producing scores (probability predictions) for tokens it was not trained on. The system trains an additional 5 preclassifiers each using four fifths of the available tokens, and uses each to produce a score for the tokens in was not trained
The features for the preclassifier are: word shape, character 4-, 3-, 2- and 1-grams (including start and end markers, so this gets prefixes and suffixes), tests against various regular expressions, and tests to see if the token is in various lexicons (a list of chemicals derived from ChEBI [13], another list from ChemSpider [14], and a standard English word list).

Additionally, there are two sets of features that are sent directly to the neural network. One set includes length-based measures (including the number of all non-lowercase characters, the number of all non-letter characters and the number of digit characters) as numerical features, and binary features for the lexicons and regular expressions above. This set is passed to the network in its entirety. The second set of features consists of the 100 most common binary features selected from 2- and 3-character suffixes and word shapes. Note that none of these features look at neighbouring tokens; unlike in a CRF where features for neighboring tokens are included explicitly, in this system we rely on the neural network to combine features from different tokens.

The features for each token in a sentence (excluding the embeddings) consist of the score from the preclassifier and the three sets of features from the paragraph above.

The minimalist system uses only character embeddings – a set of 90 characters (letters, digits, common punctuation) is used, and unknown characters acting as the 91st character.

### 2.2 Neural Network

The traditional system has two inputs that merge together. One branch is an embedding layer, with 300 units per token. The other branch takes the various features mentioned in the section above, passes them to a 1D convolution layer, with a window size of 3 (i.e. for each token the layer takes inputs from the previous token, the token itself and the next token), and 256 outputs per token, with a ReLu activation function. These two branches are merged, and the 300+256 outputs per token are used as inputs for a bidirectional LSTM layer, with 64 outputs per token per direction. These outputs are inputs for the final layer – a time-distributed dense layer, with 5 outputs per token (corresponding to S, O, B, I and E
tags), with a softmax activation function – this ensures that the outputs for each token sum to 1.

The system was trained for 20 epochs, with the model being saved after each epoch, and evaluated against the remaining 20% of the data. Each epoch was trained in mini-batches, drawn from batches of sentences all the same length. The model from epoch that gave the best F score – the 16th epoch – was selected.

The minimalist system has a single input layer – an embedding layer with 200 outputs per character, followed by three bidirectional LSTM layers – the first with 128 outputs per character per direction, the second and third with 64 outputs per character per direction. The final layer was identical to the final layer in the traditional system.

This system was trained for 30 epochs. As before, the model from the highest-scoring epoch – the 27th – was selected. The same mini-batch training procedure was used, except that for the first four epochs, the system was trained in order of sequence length, with the shortest sequences first.

Both networks were trained with the RMSProp optimizer, using the categorical cross-entropy loss function. The LSTMs in both systems, and the convolutional layer in the first system, were trained using dropout [15] with the dropout probability set to 0.5.

### 2.3 Post-processing

The neural network assigns five scores to each token (for the minimalist system, for "token" read "character") – one for each of the S, O, B, I and E tags. To convert this to a list of entities, the system scans for possible entities, looking up the value for each tag in each possible entity in each position, taking the minimum value, and, if this is above a threshold, accepting the entity and assigning it that value as a score. The thresholds were 0.5 for the traditional system, 0.6 for the minimalist system, and were chosen by investigating several thresholds and selecting those that maximized the F scores.
We experimented with two variants of the minimalist system – a high-recall run, with the threshold adjusted to 0.05 (the threshold that gave a recall around 0.95 when overlapping entities are counted), and a run with different thresholds depending on whether the possible entities appeared in various lists. The results of these experiments are given in the table below, and are labeled Min+hr and Min+pp respectively.

The ensemble system works by running both systems with a low threshold, and generating two lists of entities. If an entity appears in only one list, its score is the score from that list, otherwise it is the sum of the scores from the two lists. The possibly combined score is then divided by 2, and a threshold of 0.475 is applied – this threshold is just below 0.5, so entities that get a high score from one system but no score from the other may nevertheless be recognized.

This challenge does not allow overlapping entities to be submitted, so in runs where this is a danger, checks are done and the lower-scoring entities are discarded.

### 3 Results and Discussion

The raw results are as below:

<table>
<thead>
<tr>
<th>System</th>
<th>Official Test</th>
<th>Internal Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>Precision</td>
</tr>
<tr>
<td>Traditional</td>
<td>.8919</td>
<td>.8867</td>
</tr>
<tr>
<td>Minimalist</td>
<td>.8901</td>
<td>.8865</td>
</tr>
<tr>
<td>Ensemble</td>
<td>.9032</td>
<td>.9002</td>
</tr>
<tr>
<td>Min+pp</td>
<td>.8899</td>
<td>.8960</td>
</tr>
<tr>
<td>Min+hr</td>
<td>.8398</td>
<td>.7777</td>
</tr>
</tbody>
</table>

Internal evaluations were performed on the 1/5 of the training data not used for training. It is notable that the official test tended to give better results – presumably due the quality of the test data being higher than that of the training data.

Despite the different methods involved, the traditional and the ensemble system performed similarly. It is possible that the minimalist system
could be improved by adding additional layers, adding more outputs per layer, or otherwise increasing the amount of computing power required. The system is slow to train and the parameters were not fully optimized. Due to lack of time, we have not investigated the inner workings of the minimalist system – however we think that the memory in the LSTMs should make them useful for spotting character n-grams.

It is possible to include CRF layers as the final layer of an LSTM-based neural network [5] – however this was not investigated as the toolkit that we were using [6] did not support it.

During development, the postprocessing system had looked promising – however this did not survive the final evaluation. The high recall run was also a disappointment – the BeCalm evaluation script does not allow the submission of overlapping entities, and this puts a cap on the level of recall available. Without this constraint, our internal evaluation gets a recall of 0.9551 (precision 0.6900, F 0.8012). One feature of the minimalist system is that there is no hard limit on recall imposed by the tokeniser – with the traditional system there are some entities with boundaries in the middle of tokens, impossible for the system to get even with the lowest possible threshold, setting a hard limit on recall.

The ensemble system has achieved an F of 0.9032 – above the symbolic "90% barrier". This is not yet human-level performance – for example an inter-annotator agreement study of chemical named entity annotation found that an F of 0.93 is possible [16]. This score was achieved without extensive training or post processing on a relatively simple model. For this reason we feel that this approach has demonstrated considerable promise, and we will continue to investigate. Our systems and their source code available on-line [17].

4 Acknowledgment

We would like to thank Adam Bernard for the initial Python translation of the Oscar4 tokeniser, and Colin Batchelor, Nicholas Bailey and Jeff White for valuable discussions.
REFERENCES

17. https://bitbucket.org/rscapplications/chemlistem
Recognition of Chemical Entity Mention in Patents Using Feature-rich CRF

Yuankai Guo\textsuperscript{1}, Shiyi Zhao\textsuperscript{2}, Chen Qu\textsuperscript{3}, Lishuang Li\textsuperscript{*4}

School of Computer Science and Technology, Dalian University of Technology, China
\textsuperscript{1}guoyuankai@mail.dlut.edu.cn; 
\textsuperscript{2}zhaoshiyi2013@mail.dlut.edu.cn; 
\textsuperscript{3}quchen0502@gmail.com; 
\textsuperscript{*4}lilishuang314@163.com

Abstract. Chemical named entity recognition is the preliminary groundwork for scientific research and biomedical application. For Chemical Entity Mention in Patents (CEMP), a subtask of the BioCreative V.5, we implement a CRF++ template trained with a set of features including general linguistic features and chemical characteristics. Our system performs with an F-score of 82.45% on test dataset.

Keywords. CEMP, CRF, Named Entity Recognition, Machine Learning

1 Introduction

Chemical entity mention recognition is a fundamental step in biomedical research, playing a critical role in the related biomedical researches. For this reason, the BioCreative V.5 Challenge sets Chemical Entity Mention Recognition subtask, which requires the recognition of chemical named entity mentions in text, with a training set of 21,000 patent abstracts and a test set size of 9,000. For this subtask, three main methods are often utilized, which are dictionary-based methods, rule-based methods and statistical machine learning methods. The machine learning methods are increasingly used in Named Entity Recognition (NER) for the good performance and robustness. Combined with the characteristics of Maximum Entropy Model (MaxEnt) and Hidden Markov Model (HMM), conditional random field algorithm (CRF) has achieved good performance on sequence labeling problem. So we treat this subtask as a problem of

*Corresponding author.
sequence labeling and build a CRF based system with rich features to solve it, achieving an F-score of 82.45% on test dataset.

2 System Description

2.1 System Architecture

Our system consists of four components as Fig.1: preprocessing feature extraction, train & test, and post processing. The preprocessing module utilizes tokenization to produce the labeled train set. The second module extracts the feature. The third module is a training and prediction process using CRF++. At last the post processing module is utilized to refine the results.

![Fig.1 System Architecture](image)

2.2 System Features Extraction

The features in our approach are described as Table 1. In feature extraction module, we mainly extract two category features, general linguistic features, shown as the first seven lines, and chemical characteristics, as the rest of the table.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENIA features</td>
<td>The original word and stems along with Part-of-speech tag provided by GENIA tagger</td>
</tr>
<tr>
<td>Affix</td>
<td>Prefixes and suffixes (length: 2 to 4) are extracted as features.</td>
</tr>
<tr>
<td>Word Shape[^1]</td>
<td>Pattern of the word and its brief version.</td>
</tr>
<tr>
<td>Morphological feature[^1]</td>
<td>Number of specific characters: total characters, lower case ones, upper case ones and digits.</td>
</tr>
<tr>
<td>Word Length</td>
<td>The length of the word (lens:1, 70)</td>
</tr>
<tr>
<td>feature</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vowels</td>
<td>The distribution of vowels. For example, “carbon” is extracted to “-a-o-”</td>
</tr>
</tbody>
</table>
| Orthographical feature[1]| The classification of the token consists of 31 categories.  
Word Clustering: Brown Clustering and its prefixes (length: 6 to 8) |
| Element Symbols          | We create a lexicon of element symbols for symbol recognition.  
Chemical Elements: Whether current token is a chemical element |
| Semantic feature[2]      | Characteristics specific to chemicals, including suffixes (e.g. “-yl”), alkane stems (e.g. “meth”) and trivial rings (e.g. “benzene”) |

2.3 Post Processing

- We tag all occurrences of a specific sequence as chemicals if the sequence is tagged by the CRF model at least twice.
- We balance each mention in terms of parentheses and brackets.
- Two mentions will be merged together if they are connected by a single hyphen or chemical bonds in the original text.
- We build a dictionary of chemical identifiers by extracting vocabulary matching specific patterns from CTD (Comparative Toxicogenomics Database). A token will be recognized as a chemical entity if it can be found in the lexicon.

3 Result and Discussion

Our system reports an F-score of 82.45% on test dataset with 83.10% precision and 81.81% recall. Additionally, we also have explored word clusters as a part of features.

4 Prospect

Deep learning has been widely applied to tackle NLP related tasks in recent years and has achieved a good performance on various types of tasks. Long-Short Term Memory (LSTM), as a deep learning method, has been widely applied to tackle NLP related tasks in recent years for its excellent ability of learning long-term dependencies. As our earlier work, we adopted the bidirectional recurrent neural network with LSTM unit to identify biomedical entities, achieving an F-score of 88.61% on the BioCreative GM corpus[4]. But for LSTM’s plenty of
parameters which are difficult to tune, we haven’t get a better result than the feature-rich CRF solution in CEMP subtask this year. Further experiments will be carried out to achieve satisfactory results.

5 Acknowledgments
This work is supported by grant from the National Natural Science Foundation of China (no. 61672126).

REFERENCES


Towards Robust Chemical Recognition with TaggerOne at the BioCreative V.5 CEMP Task

Robert Leaman, Zhiyong Lu*

National Center for Biotechnology Information, Bethesda, Maryland, USA
robert.leaman@nih.gov; zhiyong.lu@nih.gov

Abstract. We describe our submissions to the BioCreative V.5 CEMP task for chemical named entity recognition in patents. We experimented with improving the robustness of the predictions made by TaggerOne – a biomedical named entity recognition system intended to be generic to any entity type – through three methods. First, we improve the feature representation for out-of-vocabulary words with Brown clusters. Second, we improved the generalization of the model under cross-domain shifts with adversarial training. Third, we apply an ensemble approach. We find all three approaches to improve performance. Our highest performance was 0.8847 F-score. TaggerOne is publicly available at https://www.ncbi.nlm.nih.gov/bionlp/tools/taggerone/

Keywords. Chemical named entity recognition; adversarial training; ensemble methods

1 Introduction

Chemical patents are an attractive target for text mining due to their importance as a primary source for medicinal chemistry. However patents are less formal documents than published articles, and therefore more likely to contain noise – mistakes or even intentional obfuscations – in addition to jargon specific to biomedical chemistry. The recent series of shared tasks in chemical text mining at the BioCreative workshops have focused on chemical named entity recognition (NER) in both PubMed abstracts [1, 2] and chemical patents [3].

NCBI developed a pair of machine learning based systems for the CHEMDNER chemical named entity recognition task in PubMed abstracts. These systems, tmChem model 1 and model 2 [4], are both based on conditional random fields [5] and use a rich feature approach [6, 7]. NCBI created an ensemble system for the subsequent CEMP task for
chemical NER recognition in patents, taking advantage of the numerous open source chemical NER systems created for the previous CHEMDNER task [8]. The resulting ensemble had very high performance but was of limited practical use due to the significant computational overhead of obtaining predictions from multiple models and the difficulty of simultaneously deploying the various systems.

Previous work by Sutton, Sindelar and McCallum [9] shows that the performance improvements achieved when combining classifiers are due, at least in part, to a reduction in weight undertraining. When training a single model, the presence of one or more strong features during training can “drown out” the contribution of weaker features, causing their weights to be too low when the strong feature is not present at test time. Ensemble methods address this by emphasizing different subsets of the feature space, thus reducing the availability of the highly predictive features and making the average of the model predictions more generalizable. Neural networks address this issue with dropout: a percentage of inputs to each layer are randomly dropped during training [10]. In structured machine learning methods, however, Søgaard [11] suggests training with an antagonistic adversary: rather than removing features at random, remove a randomly selected subset of those that are highly predictive. Søgaard shows that training with an antagonistic adversary is particularly effective for cross-domain shifts, where the distribution of the test data does not match that of the training distribution.

In NER a primary source of error is vocabulary that was not observed during training. Our experiments therefore attempted to address this source of error in two primary ways. First, we improve the feature representation for out-of-vocabulary words by learning word representations from a large amount of unlabeled data. Second, since out-of-vocabulary effects are a form of cross-domain shift, we experiment with training using an antagonistic adversary. We also create an ensemble system as a benchmark for the upper limit of the performance that can be expected. We perform our experiments using TaggerOne, a recently released system for joint named entity recognition and normalization for various biomedical entities [12]. The highly flexible online training algorithm used by TaggerOne makes it ideal for experimentation.

2 Methods

TaggerOne is a machine learning based system for joint named entity
recognition and normalization [12]. Joint training and inference allows the model to use the normalization information to inform the NER component, resulting in increased performance for both subtasks. The model consists of a semi-Markov [13] structured linear classifier [14] using a rich feature approach for NER [6, 7], a supervised semantic indexing approach for normalization [15, 16]. The model is trained with the online training algorithm MIRA (the margin-infused relaxed algorithm) [17], and requires the specification of two hyperparameters: the regularization, which controls the size of the updates, and the maximum step size, which sets an upper bound on the update size. As a semi-Markov model, it performs segmentation and classification simultaneously, allowing one state per entity type instead of two states (as in the BIO scheme) or four states (as in the BIOEW scheme). Our adaptation of TaggerOne in this manuscript does not make use of its normalization capability.

The original TaggerOne feature set includes a wide variety of features. At the token level, these features include the token text, stem, part of speech, character n-grams, and patterns. Features at the segment level include surrounding characters, tokens and whether the segment contains unbalanced parenthesis. Some models in this work include a dictionary feature containing the chemicals lexicon provided by the Comparative Toxicogenomics Database (CTD, http://ctdbase.org/), which is derived from the chemical branch of MeSH (https://www.nlm.nih.gov/mesh). We slightly augmented this list to include the names of all of the chemical elements.

We improved the feature representation for out of vocabulary words by leveraging the availability of a large unlabeled text dataset from a similar domain, namely, PubMed. Previous work has shown Brown clusters [18] to be useful in NER [19]. In addition, more recent work has improved performance by learning a word representation from a large amount of unlabeled data [20]. Our experiments employed the Brown clusters and the clustered word representation vectors distributed by the banner-chemdner tool [21]. Our preliminary experiments showed an improvement with Brown clusters for lengths 4, 6, 10 and 20 (data not shown), which we adopted for the final experiment. Our preliminary ex-
periments did not show an improvement for word vector clusters, however (data not shown), and word vector clusters were therefore not considered further.

![Diagram](image)

Figure 1. Description of the individual models used to train the ensemble and the flow of data through the system. Diagram adapted from [8].

Our implementation of antagonistic adversaries [11] selects a subset of features at random for each training instance – the percentage of features to select being an additional hyperparameter – then drops those features whose weight is greater than one standard deviation of the mean weight for all features. We consider the weight of the feature to be the Euclidean length of its weights across all states. Features that are dropped have their values set to zero for all feature vectors in an instance and across all states. For efficiency, features are selected by first sampling from a binomial distribution to determine the number of features that should be dropped, then the features themselves are selected randomly.

We included one ensemble run, using a configuration similar to our ensemble for the previous CEMP task but using TaggerOne to combine the individual predictions. The ensemble used four systems: tmChem model 1 and tmChem model 2 [4], the Wuhan University CHEMDNER tagger [22] and banner-chemdner [21]. All systems are open source, based on conditional random fields [5] and a rich feature approach. The systems in the ensemble were trained using combinations of patent and PubMed training data, as described in Figure 1. The output of each individual system was then input into TaggerOne as a binary feature, which
were the only features used, and TaggerOne was trained using the remaining training data. We also included the output of tmChem model 1 as one submission.

The initial implementation of TaggerOne instantiated features for all instances prior to training, making the memory requirement scale roughly linearly with the number of instances. This becomes unacceptable for very large datasets, and is unnecessary since TaggerOne uses online training. Merely instantiating features prior to training, however, would cause an unacceptable increase in training time. Instead we used separate concurrent processes to perform feature extraction and training. This allows TaggerOne to scale to arbitrarily large datasets without increasing the training time or the memory requirement.

3 Results

Our five submitted runs consisted of three with TaggerOne alone, one with TaggerOne as an ensemble, and one with tmChem model 1. We separated two thousand patents from the initial training set as a holdout set, and designated the remaining documents as the training set. The three runs with TaggerOne alone also included the PubMed abstracts from the CHEMDNER task as training data [2]. The TaggerOne ensemble was trained as described in Figure 1. tmChem was trained by combining all available patent data with the PubMed abstracts from the CHEMDNER task, as for the runs with TaggerOne alone, but also included the chemical annotations from the BC5CDR corpus [23]. The value of all TaggerOne hyperparameters, when used, was set by cross-validation on the holdout set. The four configurations of TaggerOne are described in Table 1.

Table 1. Configuration of the four variations of TaggerOne submitted. The Enhanced feature set consists of the Initial feature set plus the dictionary feature from the CTD chemical vocabulary and Brown clusters.

<table>
<thead>
<tr>
<th>Run</th>
<th>Regularization</th>
<th>Maximum step size</th>
<th>Adversary</th>
<th>Feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggerOne-Raw</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Initial</td>
</tr>
<tr>
<td>TaggerOne-Brown</td>
<td>10.0</td>
<td>0.001</td>
<td>0.00</td>
<td>Enhanced</td>
</tr>
<tr>
<td>TaggerOne-Adversary</td>
<td>10.0</td>
<td>0.001</td>
<td>0.03</td>
<td>Enhanced</td>
</tr>
<tr>
<td>TaggerOne-Ensemble</td>
<td>0.1</td>
<td>0.001</td>
<td>0.10</td>
<td>Ensemble</td>
</tr>
</tbody>
</table>
The performance of the five models submitted to the task on our internal holdout set are described in Table 2.

Table 2. Results for the five submitted runs on our internal holdout set. The highest value is shown in bold.

<table>
<thead>
<tr>
<th>Run</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggerOne-Raw</td>
<td>0.8405</td>
<td>0.8630</td>
<td>0.8516</td>
</tr>
<tr>
<td>TaggerOne-Brown</td>
<td>0.8383</td>
<td>0.8739</td>
<td>0.8558</td>
</tr>
<tr>
<td>TaggerOne-Adversary</td>
<td>0.8424</td>
<td>0.8746</td>
<td>0.8582</td>
</tr>
<tr>
<td>TaggerOne-Ensemble</td>
<td>0.8532</td>
<td><strong>0.9150</strong></td>
<td><strong>0.8830</strong></td>
</tr>
<tr>
<td>tmChem model 1</td>
<td><strong>0.8799</strong></td>
<td>0.8623</td>
<td>0.8710</td>
</tr>
</tbody>
</table>

The official performance of the five models submitted to the task are described in Table 3.

Table 3. Official results for the five submitted runs. The highest value is shown in bold.

<table>
<thead>
<tr>
<th>Run</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggerOne-Raw</td>
<td>0.8639</td>
<td>0.8733</td>
<td>0.8686</td>
</tr>
<tr>
<td>TaggerOne-Brown</td>
<td>0.8641</td>
<td>0.8807</td>
<td>0.8723</td>
</tr>
<tr>
<td>TaggerOne-Adversary</td>
<td>0.8635</td>
<td>0.8795</td>
<td>0.8715</td>
</tr>
<tr>
<td>TaggerOne-Ensemble</td>
<td>0.8439</td>
<td><strong>0.9297</strong></td>
<td><strong>0.8847</strong></td>
</tr>
<tr>
<td>tmChem model 1</td>
<td><strong>0.8731</strong></td>
<td>0.8765</td>
<td>0.8748</td>
</tr>
</tbody>
</table>

4 Discussion

We first note that the official results are generally higher than the results on our internal holdout set. We note that while TaggerOne is intended to work well for any biomedical entity type, its performance is nearly as strong as tmChem, which is specifically dedicated to chemical NER. We see that using TaggerOne “out of the box” – without setting or optimizing hyperparameters – results in performance that approaches the optimal configuration. Alternately, adding Brown clusters improved performance for both the holdout and test sets. Adversarial training helped in the holdout set, but slightly hurt in the test set, possibly due to the difference in the holdout and test sets causing the adversarial training hyperparameter to be set to a suboptimal value. The ensemble provided the highest performance. We found adversarial training to help significantly with the ensemble configuration in our preliminary experiments (data not
shown); achieving this performance required the adversarial training hyperparameter to be set to a relatively high value. The strong performance by tmChem is primarily due to high precision.

5 Conclusion

We have explored several methods of improving the robustness of predictions for chemical named entity recognition in patents. We have shown that improving the feature representation for out-of-vocabulary words (via Brown clusters) improves performance. Adversarial training improved performance on the holdout set and may be worth exploring further. The highest performances were obtained by the dedicated tool for chemical NER, tmChem, and the ensemble approach with TaggerOne.

6 Acknowledgment

The authors thank the organizers of the BioCreative V.5 CHEMDNER task and the BeCalm team. We also thank the authors of the individual open source systems used in this work. This research is funded by the National Institutes of Health Intramural Research Program, National Library of Medicine.

REFERENCES

A hybrid text mining system for chemical entity recognition and classification using dictionary look-up and pattern matching @ BeCalm challenge evaluation workshop

Kalpana Raja1*, Sabenabanu Abdulkadhar2, Lam C Tsoi1,3,4
Jeyakumar Natarajan2*

1Department of Dermatology, University of Michigan Medical School, Ann Arbor, MI, USA
2Data Mining and Text Mining Laboratory, Department of Bioinformatics, Bharathiar University, Coimbatore, Tamil Nadu, India
3Department of Computational Medicine & Bioinformatics, University of Michigan Medical School, Ann Arbor, MI, USA
4Department of Biostatistics, University of Michigan, Ann Arbor, MI, USA

*1rkalpana@med.umich.edu; sabenabanu.a@hotmail.com; alextsoi@med.umich.edu; 2n.jeyakumar@yahoo.co.in

Abstract. Chemicals as therapeutics and investigational agents receive much attention in clinical research and applications recently. However, automated approaches to recognize and categorize the chemical entities in biomedical text are challenging because of the wide varieties of morphologies and nomenclature. We present here a hybrid text mining system that combines chemical lexicon and patterns for recognition/categorization. We applied this approach to identify chemical entities from the patent abstracts of BioCreative V.5 Chemical Entity Mention Recognition (CEMP) corpus. We also compared the hybrid approach with the “traditional” lexicon-based method, and illustrated that the hybrid approach can achieve enhanced performance (i.e. precision, recall, and F-score) than the lexion-based method.

Keywords. Chemical entity recognition; Chemical categorization; Text mining; Pattern matching; Chemical lexicon.

1 Introduction

The advances in data revolution reveal valuable information on the new roles of chemicals in disease treatment and adverse reaction. The effect of chemicals on the biological systems as therapeutic agents (i.e. drugs), investigational agents in drug discovery and unintentional agents to understand the adverse effects make them an important class of biomedical entities in clinical research and applications [1]. The scientific findings on chemicals are commonly available in published bi-
Medical literature, and automated approaches using text mining techniques have proven to be effective for entity extraction. Nevertheless, the task is challenging due to the different morphologies and nomenclatures used for representing chemical entities [2].

Different text-mining approaches have been developed utilizing techniques such as rule-based [3], dictionary based [4], machine learning [1], and hybrid approaches [5]. In particular, the Conditional Random Fields [6,1,5] and Support Vector Machines [7] algorithms with rule-based or dictionary-based approaches are widely used in chemical entity recognition. In spite of several existing approaches, the challenge is still open and leaves a space for improvement. BioCreative V.5 Chemical Entity Mention Recognition (CEMP) task invited the text mining community to develop novel and robust approaches for recognizing and categorizing the chemical entities in a set of patent abstracts. We presented a hybrid approach that combines a chemical lexicon and a pattern matching module for recognizing and categorizing chemicals from the patent abstracts.

![Figure 1: Workflow of the proposed system](image-url)
2 Systems description and methods

Overview

Our approach for recognizing and categorizing the chemicals consists of two parts: (1) building a chemical lexicon; (2) extracting and categorizing the chemicals from the patent abstracts using the lexicon and a pattern matching module. Figure 1 presents the workflow of the proposed system.

Chemicals lexicon

Processing of UMLS Metathesaurus

The chemical lexicon was compiled from three resources: UMLS Metathesaurus [8], DrugBank [9] and PharmGKB [10]. The 2015AB version of UMLS Metathesaurus was downloaded from the UMLS Terminology Services (UTS) and customized to Rich Release Format using MetamorphoSys, an in-build UMLS installation wizard and Metathesaurus customization tool [11]. The resource contains more than 3.2 million medical concepts (e.g. chemicals, drugs, diseases) and 12.8 million synonyms from over 190 vocabularies including SNOMED, and ICD 9/10 diagnostic codes. We filtered 4,652,003 medical concepts and synonyms that are in English, and selected only the concepts belonging to “Chemicals and Drugs” semantic type. However, the concepts from other semantic groups (e.g. disease, living organisms) were found to overlap with the chemicals synonyms. We removed the common concepts or synonyms between the semantic groups, common English terms (e.g. link, conduct, aim), semantic types (e.g. amino acid, hormone) and abbreviation overlap (e.g. the abbreviation C maps to Catechin, Cocaine, Carbon and Blood group antigen C). We used Attempto, a resource for controlled natural language and a rich subset of Standard English [12] to identify common English terms as chemicals. The latest version is released in 2013 and contains 97,526 English terms. The other three error types were identified through simple overlap. The process yielded 461,379 chemical concepts and 929,747 synonyms.

Processing of DrugBank and PharmGKB

The latest version of DrugBank database [13] was downloaded and parsed with UTF-8 encoding to handle chemicals with Greek alphabets
and special characters (e.g. \(\alpha\)-methylthiofentanyl). We extracted 8,203 drugs, 1,201 salts, and synonyms. We downloaded the drugs.zip file from PharmGKB and extracted 3,175 chemical names, 6,763 generic names and 18,309 trade names [10]. The generic names and trade names are synonyms. We compiled the chemical entities and synonyms from the three resources and assigned a customized ID which is unique for a chemical.

**Recognition of chemical entity mention**

We combined the chemical lexicon with MedTagger [14] for application. MedTagger is an Open Health Natural Language Processing (OHNLP) tagger that uses a lexicon for entity extraction. It uses a pipeline of text mining approaches such as tokenization, lexical normalization, dictionary look-up using the well-known Aho-corasick approach and concept screening [15]. The CEMP task defines seven chemical classes namely systematic, identifier, formula, trivial, abbreviation, family and multiple classes for categorization. The systemic class defines the IUPAC and IUPAC-like chemical nomenclature (e.g. 2-acetoxy-benzoic-acid). The identifiers are the database identifiers from various chemical databases (e.g. 2244, a PubChem ID). The formula includes molecular formula (e.g. \(\text{CH}_3\text{COOC}_6\text{H}_4\text{COOH}\)), canonical and isomeric SMILES (e.g. \(\text{CC(=O)OC1=CC=CC=C1C(=O)O}\), InChI (e.g. \(\text{InChI=1S/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h2-5H,1H3,(H,11,12)}\)), and InChIKey (e.g. \(\text{BSYNRYMUTXBSQ-UHFFFAOYSA-N}\)). The trivial names are the trade name / brand name / common name / generic name of a marketed drug (e.g. acetylsalicylic acid, aspirin). The abbreviations are the standard acronyms (e.g. GABA for gamma-aminobutyric acid). The family class is associated with the chemical structure (e.g. diphenols, terphenoids). The multiple class includes the chemical names that are not described in a continuous string of characters (e.g. thieno3,2-d fused oxazin-4-ones). We developed a pattern matching module using Java regex to categorize the chemicals into seven different classes as shown in Table 1. Among the seven classes, formula and identifiers are not available in the chemical lexicon and the pattern matching module alone was applied for their recognition. In addition, we used a lexicon compiled from PubChem [16], DrugBank [13] and KEGG DRUG [17] for classifying trivial names. The family
class consists of sub-classes and we created a family lexicon to distinguish the family sub-classes (e.g. systematic) from the main systematic class.

**Table 1:** Patterns to classify chemical Entity classes

<table>
<thead>
<tr>
<th>Chemical class</th>
<th>Regex pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEMATIC</td>
<td>\b[^\d.$]\b</td>
<td>beta.-alethine</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>\w+ [A-Za-z]\w+ [0-9]\w+</td>
<td>KMD-3213</td>
</tr>
<tr>
<td>FORMULA</td>
<td>\b<a href="%5Ba-z%5D?!d*">A-Z</a>?[A-Z]?!d*\b</td>
<td>CH2 COOH</td>
</tr>
<tr>
<td>TRIVIAL</td>
<td>[A-Z][a-z]\w+</td>
<td>matrinetannate</td>
</tr>
<tr>
<td>ABBREVIATION</td>
<td>[A-Z]{4}</td>
<td>EDDA</td>
</tr>
<tr>
<td>FAMILY</td>
<td>[a-z][A-Z][a-z]</td>
<td>Alkaloids</td>
</tr>
<tr>
<td>MULTIPLE</td>
<td>\b(and</td>
<td>or)\b</td>
</tr>
</tbody>
</table>

**Dataset and Evaluation**

The BioCreative V.5 CEMP task consists of 21,000 patent abstracts in the training data and 9,000 patent abstracts in the test data. While the training data consists of 99,632 annotations (Table 2), the annotations for test data are yet to be released. The standard evaluation metrics such as Precision, Recall and F-score were used to evaluate the performance of the proposed system [18].

**Table 2:** Chemicals annotation in the training data

<table>
<thead>
<tr>
<th>Chemical class</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEMATIC</td>
<td>28,580</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>278</td>
</tr>
<tr>
<td>FORMULA</td>
<td>6,818</td>
</tr>
<tr>
<td>TRIVIAL</td>
<td>25,927</td>
</tr>
<tr>
<td>ABBREVIATION</td>
<td>1,373</td>
</tr>
<tr>
<td>FAMILY</td>
<td>36,238</td>
</tr>
<tr>
<td>MULTIPLE</td>
<td>418</td>
</tr>
</tbody>
</table>
3 Results and Discussion

The traditional lexicon-based approach achieved 0.532 precision, 0.651 recall and 0.586 F-score on the training data and 0.472 precision, 0.515 recall and 0.492 F-score on the test data (Table 3). The hybrid approach that combines the chemical lexicon and patterns recognition/categorization achieved an enhanced performance of 0.601 precision, 0.651 recall and 0.625 F-score. We also report the performance of the system on each component i.e. entity recognition and classification (Table 4). The chemical dictionary is the main component for identifying five types of classes excluding identifiers and formula. Though the dictionary contains the systematic and trivial names, we observed that it does not contain IUPAC like names and all trivial names. By incorporating a pattern matching approach for IUPAC like names and including a lexicon for trivial names from resources that are not in chemical lexicon, we show an enhanced performance of more than 10% on precision, recall, and F-score when compared to the traditional lexicon-based approach.

Table 3: System performance reported for CEMP task

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data</td>
<td>0.532</td>
<td>0.587</td>
<td>0.558</td>
</tr>
<tr>
<td>Test data</td>
<td>0.472</td>
<td>0.515</td>
<td>0.493</td>
</tr>
</tbody>
</table>

Table 4: System performance after CEMP task on training data

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity recognition</td>
<td>0.582</td>
<td>0.651</td>
<td>0.614</td>
</tr>
<tr>
<td>Classification</td>
<td>0.570</td>
<td>0.773</td>
<td>0.658</td>
</tr>
<tr>
<td>Entity recognition+ Classification</td>
<td>0.601</td>
<td>0.651</td>
<td>0.625</td>
</tr>
</tbody>
</table>

Limitations and Future work

Though the patterns perform well on single term entities (e.g. SMILES), we observed pattern overlap on entities with multiple terms (e.g. IUPAC like names) that resulted in partial identification of chemical entities. The chemical annotations in the training data mainly be-
long to systematic, trivial and family class (i.e. ~89%), among which the recognition of systematic and family classes are more challenging. As a future work, we will be replacing the patterns with a machine learning approach using CRF, an established approach for entity recognition.

4 Conclusion

We present a hybrid approach that combines a chemical lexicon compiled from three resources namely UMLS Metathesaurus, DrugBank and PharmGKB, and a pattern matching approach for chemicals entity recognition and classification into seven different classes defined in BioCreative V.5 CEMP task. We report the performance of the proposed system only on entity recognition and classification task, and as a system with both the modules. In the current study, we take the advantage of many available resources (e.g. PubChem, KEGG Drug) and a pattern matching approach for entity recognition and classification.

Acknowledgement

The research has received funding from Dermatology Foundation USA, the Arthritis National Research Foundation USA and the National Psoriasis Foundation USA, and the University Grants Commission Maulana Azad National Fellowship for Minority students (UGC-MANF), Government of India, Grant No: F1-17.1/2015/MANF-2015-17-TAM-54928. The authors acknowledge all the funding received.

REFERENCES

4. Hettne, Kristina M., Rob H. Stierum, Martijn J. Schuemie, Peter JM Hendriksen, Bob JA Schijvenaars, Erik M. Van Mulligen, Jos Kleinjans, and Jan


IBEnt: Chemical Entity Mentions in Patents using ChEBI

Andre Lamurias*, Luis F. Campos, and Francisco M. Couto
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. This article presents our approach to the CEMP task of BioCreative V.5, which consisted in using our system, IBEnt, to identify chemical entity mentions in patents through machine learning and semantic similarity techniques. The features used combine the results of a CRF classifier, two lexical matching methods (FiGO and MER) and semantic similarity measures on ChEBI ontology. We also tested the usage of MER by itself, without the machine learning approach. Combining these techniques, we submitted 5 runs for evaluation. We obtained better results using the machine learning approach with lexical and semantic similarity features. The best F-score obtained was 0.8541, while the MER system obtained 0.5967.

Key words: Named-Entity Recognition, Machine Learning, Semantic Similarity, Conditional Random Fields

1 Introduction

This paper presents our approach to the BioCreative V.5 CEMP task (Chemical Entity Mention in Patents) [13]. The objective of this task was to develop a system for detecting chemical entities in patent documents. A gold standard of 21,000 patents with chemical annotations was provided to the participants. For each chemical entity mentioned in a document, the start and end offsets were provided, along with the original text. We divided the gold standard into two partitions of equal size, which we refer to as training and development sets. The test set provided to the participants consisted of 9,000 patents. The participating systems were evaluated by the quality of their annotations in this test set.

Our approach used IBEnt [9], a framework to identify biomedical entities based on machine learning and semantic similarity techniques. We trained one classifier using Conditional Random Fields (CRF) and combined the results of that classifier with semantic techniques and a lexicon-based system to train a Random Forests classifier. The code used to generate our results is available at our GitHub repository1. The remainder of this article describes the features and resources used for this task, presents our results and discusses the performance of each approach.

* Corresponding author: alamurias@lasige.di.fc.ul.pt
1 https://github.com/lasigeBioTM/IBEnt
2 Systems description and methods

We trained a CRF classifier with CRFsuite [11] on the training set annotations. The features used consisted in the linguistic, orthographic, morphological and contextual properties of the tokens, as well as domain-specific features. For most features, we considered a contextual window of size one, i.e., the value of the same feature for the previous and next token. Lemma and Part-of-Speech tags were obtained using Stanford CoreNLP [10]. Furthermore, we used three domain-specific features that have been also been used previously in similar tasks. These domain features checked if the token had a greek letter, a dash, or a periodic table element. A more detailed description of these features can be found in [9].

In the last decades the biomedical community has been developing and using ontologies to represent entities [2]. For example, in the case of chemical compounds we have Chemical Entities of Biological Interest (ChEBI) [6]. Each entity identified with the CRF classifier was matched to ChEBI, using a lexical similarity method, FiGO [4]. This method assigns a confidence score to each mapping, based on the information content of each word of the expression. Words that are more common have lower information content, contributing to lower confidence scores, while more informative words contribute to higher confidence scores. We refer to this confidence score as FiGO score.

We then computed the semantic similarity between every entity in the same sentence. Our assumption is that entities mentioned in a limited text window are more similar than entities mentioned across larger text windows. This assumption can be used to filter false positives made by the CRF classifier and FiGO[1, 9]. For example, if “2,3-bisphosphoglyceric acid”, “cyclic 2,3-bisphospho-D-glyceric acid” and “2,3-bisphosphoglycerate” were recognized in the same sentence, and assuming that the first two are semantically related, and the latter entity has a low semantic similarity to the first two, then we would have less confidence on the latter entity being correctly recognized. The semantic similarity score of an entity consisted in the maximum similarity to other entities identified in the same sentence.

We used five semantic similarity measures (SSM) (Resnik [15], simUI [5], simGIC [14], h-simUI, h-simGIC [9]), therefore obtaining five semantic similarity scores for each entity.

We then used the CRF, lexical and semantic similarity scores as features for a Random Forests classifier. The Random Forests implementation used was from scikit-learn [12]. The objective of this classifier was to exclude false positives from the CRF results. As such, the training data consisted of one instance for each entity identified by the CRF classifier. Since the CRF classifier was trained on the training set, we trained the Random Forests classifier on the development set.

As a comparison to the machine learning approach, we also used a lexicon-based system - MER [3]. We constructed four lexicons using chemical entities
datasets freely available online (ChEBI\textsuperscript{2}, ChEMBL\textsuperscript{3}, DrugBank\textsuperscript{4} and HMDB\textsuperscript{5}). Using the training data, we tested which combination of these would give rise to the best performance. We found out that using a lexicon consisting of the terms included in ChEBI, ChEMBL and HMDB achieved the highest F-Score. One of the runs consisted in the results of using MER with this lexicon. We also constructed a lexicon consisting of all the terms annotated in the training set. We knew beforehand that using MER would return low performance scores, but we thought that would be interesting to study how a fast and simple lexicon-based system as MER would compare with more complex systems that use machine learning.

Figure 1 provides an overview of our system. We obtained scores from CRF, FiGO and semantic similarity measures. These scores were used to train a Random Forests classifier. We also incorporated three features based on the three lexicons used with MER. An additional feature was added to each entity for each lexicon used, which had the value 1 if that entity was found on that lexicon, and 0 otherwise. After training the Random Forests classifier, we applied to the test set documents the same process that was applied to development set.

2.1 Runs

We combined the techniques previously described into 5 results submissions (runs) (Table 1). Our intention was to test a machine learning system (IBEnt), a rule-based system (MER) and a combination of both. Therefore, run 1 consisted in using IBEnt with the Random Forests classifier previously described. On run 2, we added features to this classifier based on the MER results. Run 3 consisted in using MER with a lexicon composed by the terms from ChEBI, ChEMBL and HMDB, while run 4 used MER with the terms found on the training set. Finally, run 5 combined the lexicons used in run 3 and 4 with a lexicons composed by the terms found on the test set by IBEnt (run 1). While run 2 represents how the results of a rule-based system can be used in the context of machine learning, the idea of run 5 was to show that machine learning can be used to generate lexicons that can then be used by more efficient lexicon-based systems.

3 Results and Discussion

After processing the documents of the test set, we submitted the results of each run to the BeCalm platform. The precision, recall and F-score scores obtained are shown in Table 2.

The highest F-score obtained was with run 1, which used CRF, FiGO and semantic similarity features. Adding features based on lexicon matching (run 2) did not improve recall nor precision.

\textsuperscript{2} https://www.ebi.ac.uk/chebi/
\textsuperscript{3} https://www.ebi.ac.uk/chembl/
\textsuperscript{4} https://www.drugbank.ca/
\textsuperscript{5} http://www.hmdb.ca/
When comparing the runs that use a lexicon-based system (3, 4 and 5), the best F-score was 0.5967 (run 4). This run used only the entities found in the training data as lexicon. This lexicon performed better than combining the ChEBI, ChEMBL and HDMB vocabularies (run 3).

The script used to generate run 5 had an error that eliminated the entities detected on multiple lexicons. The minimum expected recall would be the same recall as run 4 (0.5747), since run 5 includes the lexicon used on run 4. However, when merging lexicons, repeated entities were accidentally excluded. Since we did not have access to the test set annotations, it is not possible to test again with the bugfix.

We measured the time necessary to process the test set using the Random Forests approach compared to the lexicon-based approach. While it took an average of 5.19 seconds to process each document using Random Forests, the
Table 2. Results obtained on the test set during the competition.

<table>
<thead>
<tr>
<th>Run</th>
<th>Approach</th>
<th>Features</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Random Forests</td>
<td>CRF, FiGO, SSM</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Random Forests</td>
<td>CRF, FiGO, SSM, MER</td>
<td>ChEBI, ChEMBL, HDMB</td>
</tr>
<tr>
<td>3</td>
<td>MER</td>
<td>-</td>
<td>ChEBI, ChEMBL, HDMB</td>
</tr>
<tr>
<td>4</td>
<td>MER</td>
<td>-</td>
<td>Training set</td>
</tr>
<tr>
<td>5</td>
<td>MER</td>
<td>-</td>
<td>ChEBI, ChEMBL, HDMB, Training set, Run 1 output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8732</td>
<td>0.8338</td>
<td>0.8531</td>
</tr>
<tr>
<td>2</td>
<td>0.6705</td>
<td>0.7452</td>
<td>0.7059</td>
</tr>
<tr>
<td>3</td>
<td>0.5367</td>
<td>0.3794</td>
<td>0.4445</td>
</tr>
<tr>
<td>4</td>
<td>0.6205</td>
<td>0.5747</td>
<td>0.5967</td>
</tr>
<tr>
<td>5</td>
<td>0.5476</td>
<td>0.3042</td>
<td>0.3911</td>
</tr>
</tbody>
</table>

lexicon-based approach took 0.37 seconds per document using the same hardware. This difference in processing time was the reason we tested this approach on some runs. Although we were not able to obtain F-scores as high as with machine learning, the lexicon-based approach was able to process the documents much faster.

On the previous edition of this task [7], the highest F-score obtained was 0.8937, which is 0.0406 higher than our best F-score on this edition. The highest precision was 0.8971, which is closer to the precision we obtained this year.

4 Conclusion

We present our open-source system, IBEnt, that participated in the CEMP task of BioCreative V.5. IBEnt is mainly based on machine learning and semantic similarity techniques. Semantic similarity is calculated using the ChEBI ontology. This system obtained an F-score of 0.8531, using CRF, FiGO and semantic similarity features. Furthermore, we combined this approach with MER, a lexicon-based system, to study how these two approaches can be used together.
Using a lexicon-based system, we obtained a best F-score of 0.5967. This type of approach can be used in cases where the response time is the priority, instead of the quality of the results. However the quality of the results obtained with this approach may be improved by incorporating a more comprehensive lexicon, by adding abbreviations, synonyms and other types of chemical descriptors. To accomplish this, we will have to carefully analyze all the descriptors of chemical compounds found by IBEnt and not found by MER.

In the future we could improve our results by testing different proportions between the training and development set. We divided the gold standard in two sets of the same size, one to train a CRF classifier and the other one to train a Random Forests classifier based on results from the CRF classifier. We may also apply our distant supervision methods to improve the classifiers\[8\]. However, a different partition of the gold standard could lead to better results, for example, using 70% of the documents to train CRF and the rest to train Random Forests.

Acknowledgments. This work was supported by FCT through funding of the PhD Grant ref. PD/BD/106083/2015 and of the LaSIGE Research Unit, ref. UID/CEC/00408/2013.

References
ChemGrab: Identification of Chemical Names Using a Combined Negative-Dictionary and Rule-Based Approach

Vivekanand Sharma, PhD and Indra Neil Sarkar, PhD, MLIS

Center for Biomedical Informatics, Brown University, Providence, RI USA

vivekanand.sharma@brown.edu; neil.sarkar@brown.edu

Abstract. The growing volume of electronically available text provides the opportunity to extract potentially relevant information that may offer valuable insights. To this end, the cataloguing of documents based on named entity mentions is an essential task. Development of text mining approaches for extracting entities and relationships may enable efficient management and retrieval of relevant information within specific contexts. The system described here, ChemGrab, focused on the BioCreative V.5 CEMP Challenge that aims to identify mentions of chemical entities from within patent text. The approach used in this study to identify chemical mentions used a combination of a negative-dictionary and rules based on word-level features. The system performance on the test set achieved a micro precision, recall, and F-score of 0.53, 0.67, and 0.59, respectively.

Keywords. Named Entity Recognition; Chemical Entity Mention in Patents; Natural Language Processing

1 Introduction

The identification and organization of information within patent texts can be a crucial step [1, 2] for facilitating novelty checking, validation, and identification of starting points for knowledge discovery [3]. However, the extraction of chemical and biological entities from patent text is a challenging task due to significant differences in linguistic structures [4]. Additional challenges arise due to complexities in chemical names, term ambiguity, complex syntactic structures, and Optical Character Recognition errors [5]. Nonetheless, cataloging of patent information can be crucial in accelerating research, devising Intellectual Property management strategies, and promoting technology transfer [2].
In light of the increasing amount of digitally available text data, there is opportunity for the development of automated methods for mining key entities and relationships. Such mining can support and enhance retrieval and curation of relevant documents. To develop natural language processing (NLP) strategies for extracting named entities from patent texts, the availability of annotated corpora is an essential step. Significant effort has been invested in creation of annotated corpora (gene, protein, chemicals) in the biomedical domain (such as from scientific literature indexed in MEDLINE) [6–8]. With the recognition of the importance of mining patent data, there has been some recent effort in this direction [1, 5]. Towards encouraging the development of tools and methods for automated recognition of chemical and biological entities from medicinal chemistry patents, the BioCreative-related initiatives were organized to provide manually annotated patent text corpora for supporting the development of NLP tools that could be effectively benchmarked [4].

Named entity recognition techniques often use dictionaries containing domain specific vocabularies [9]. Systems such as Peregrine [10], TaxonGrab [11], and LINNAEUS [12] have demonstrated the utility of dictionary matching approaches to identify disease and organism names. In context of chemical entities, dictionary-matching approaches can be challenging and require post-processing rules [13]. Furthermore, high coverage of chemical concepts with dictionaries alone can be difficult due to the range and volume of novel compound names [14]. To address this challenge, rule-based and machine learning techniques have been shown to improve performance when used in combination with dictionary-based approaches [13, 15]. Finally, ensemble approaches, which combine multiple machine learning models have also shown promising results [16].

Towards achieving the goal of recognition of chemical name entity mentions for the Chemical Entity Mention in Patents (CEMP) task, this study focused on leveraging a combination of dictionary-matching and rule-based entity recognition approaches. The results from evaluation on a training and test set highlight the efficacy of approach and identify several areas for improvement.
2 Methods

ChemGrab is a system that was developed in this study, which relies on dictionaries for matching and identification of feature rules for identification of chemical tokens. The identified chemical name tokens are expanded to chemical entities, and include the start and end indices. The implementation of ChemGrab was done using Julia (v.0.5).

2.1 Dictionary of chemicals

The Jochem dictionary was used as the look-up dictionary for this study [17]. This dictionary is based on combination of information from Unified Medical Language System (UMLS), Medical Subject Headings (MeSH), Chemical Entities of Biological Interest (ChEBI), DrugBank, Kyoto Encyclopedia for Genes and Genomes (KEGG), Human Metabolome Database (HMDB), and ChemIDPlus.

2.2 Negative Chemical Dictionary

A combination of word lists from WordNet [18] and SPECIALIST Lexicon [19] was used to develop a negative dictionary of non-chemical entities. WordNet is a lexical database of English words, synonyms, and their variations. The SPECIALIST Lexicon, a UMLS knowledge source, consists of common English words and biomedical vocabulary. The lexicon includes words as well as their spelling and grammatical variants. From the combined list of these two sources, chemicals were excluded by comparison with Jochem based on an exclusion criteria of pair-distance between two strings (s1 and s2) calculated using similarity metric with a cut-off of 0.90:

\[
pair\ distance\ score = \frac{2 \times |\text{pairs}(s1) \cap \text{pairs}(s2)|}{|\text{pairs}(s1)| + |\text{pairs}(s2)|}
\]

Using the pair distance score, the chemical names from the negative dictionary were eliminated.
2.3 Word level features

Word level feature identification for chemical entity recognition was done based on a comparison of the negative and chemical dictionaries. The characters for each word were divided into three groups: (1) Alphabet (a-z); (2) Number (0-9); and (3) Other characters (excluding whitespace). From each respective dictionary, the occurrence frequency of characters with a specified separation distance within a given word (chemical or non-chemical) entity (referred to as “elements” hereafter) was recorded. A separation distance used was in the range of 1 to 5. A tf-idf inspired scoring method was used to downweight trivial elements and upweight rare ones.

\[
\text{score} = \frac{(xy)_i^t}{(xy)_t} \times \log \frac{\sum_t (x \ast)_t}{\sum_t (x \ast)_t^i}
\]

where, \( t \in \{\text{chem dataset, non-chem dataset}\} \), \( i \) is the character separation distance, \( x \) and \( y \in \{a-z, 0-9, \text{other characters (excluding whitespace)}\} \), \((xy)_i^t\) is frequency of co-occurrence of character \( x \) and \( y \) at a given separation distance \( i \) within the chemical dictionary or the word dictionary. \((x \ast)_i^t\) is the frequency of co-occurrence of character \( x \) with any other character at a given separation distance \( i \) within the chemical dictionary or the word dictionary. When \( i \) is not specified, the term indicates co-occurrence irrespective of separation distance.

Each element received a chemical score and a non-chemical score as obtained from their respective dictionary. For each set of two-character combinations (e.g., a-a) at a given separation distance (e.g., 1-5) the difference of scores were normalized using a z-score calculation and standard normal curve area, which was used for calculating the final score (described in section 2.5). Critical elements were identified using a significance level of 0.05 for non-chemicals and 0.95 for chemicals.

2.4 Tokenization

The goal of this step was to identify chemical entity names that contained whitespace. From the dictionary of chemical names containing whitespaces, a list was generated that contained three characters prior to whitespace and all of the following characters. The list was manually
evaluated to remove entries that did not seemed relevant (e.g., “ium ion” or “hyl group”). Matching and replacement of whitespaces contained within entities was done using the above described list as a scaffold. The text segments were tokenized at ‘whitespace’, ‘/’, ‘-’ and ‘;’. Additional processing steps involved removal of non-alphabet and non-digit characters from the start and end positions of tokens. Non-chemical tokens were excluded based on comparison with the negative word list described in section 2.1.

2.5 Recognition of chemical entities

After tokenization, chemical entities were identified from those strings that were not identified in the negative dictionary using following three successive steps: (1) Direct Lookup: tokens were compared to the Jochem dictionary to identify direct matches; (2) Rule-based identification: word-level features as described in section 2.3 were used to determine whether a token was a chemical. The scores of critical elements were used to calculate a final score:

\[
token\ score = 1 - \prod_{i=1}^{m} (1 - E)
\]

where, \( m \) is the total number of critical elements (2) and \( E \) is the element score. A token score of one reflected a perfect chemical token. Thresholds of 0.97 and 1.00 were tested for identification of chemical tokens; (3) Approximate matching: each chemical entity from Jochem was indexed according to those that occurred five or less times and were using lowest scoring elements (e.g., ‘o’, ‘l’, ‘1’). Using these features, potentially matching candidates were queried for further scoring. The similarity scoring was based on the Levenshtein metric, using a threshold of 0.7 (1 being exact match and 0 indicating no match). Following identification of all chemical tokens from the above four steps, the start and end indices of neighboring tokens were used to expand as a single chemical entity giving due consideration to the punctuations that occurred between them.
3 Results and Discussion

The evaluation was performed using the evaluation functionality within Biomedical Annotation Metaserver (BeCalm) participant account. This evaluation was quantified based on calculation of micro precision, recall and F-score. Two thresholds of token scores were tested during the training set submission and the best scoring system was used for final test set submission (Table 1).

Table 1: Results from evaluation on training and test data submissions

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Micro precision Training</th>
<th>Micro precision Test</th>
<th>Micro recall Training</th>
<th>Micro recall Test</th>
<th>Micro F-score Training</th>
<th>Micro F-score Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>0.5379</td>
<td>0.5293</td>
<td>0.6783</td>
<td>0.6726</td>
<td>0.6000</td>
<td>0.5924</td>
</tr>
<tr>
<td>1.00</td>
<td>0.5835</td>
<td>-</td>
<td>0.5851</td>
<td>-</td>
<td>0.5843</td>
<td>-</td>
</tr>
</tbody>
</table>

The system described here for the CEMP task relied heavily on dictionary for matching, tokenization as well as rule identification. Although preliminary in nature, the performance of ChemGrab highlights several areas for improvement. The tokenization step implemented in this study relied on direct matches to characters surrounding whitespace, which may be a limiting factor. This step could be generalized by identifying patterns instead from the token windows. Future work will aim at identifying corpus specific rules to address the problem of whitespace separation.

Improvement in performance may also be achieved by using the training corpus to learn chemical-specific tokenization rules in conjunction with negative dictionary. The scoring of word level features described here may possibly be enhanced in future by incorporating the chemical ontology structure. Such a scoring system may result in imparting higher weights to specific elements from specific chemical groups.

Additional work is required in post-processing step of combining chemical tokens to expand over multi-token mention of chemical entities. This step would involve checking balanced parenthesis, square brackets, curly brackets, punctuations, and other non-word characters, which are commonly found in chemical names. The system developed
for this study does not include tagging of acronyms, addition of which may improve the recall. Future work is expected to include using corpus specific contextual features. The system is available as a REST-compliant web service (http://bcbi.brown.edu/chemgrab).

4 Conclusion

The identification of chemical entities from biomedical literature and patents can help guide effective management and retrieval of relevant information that offer potential to guide future investigation. Here, a chemical named entity recognition approach was developed, ChemGrab, which relies on dictionary and language lexicon for look-up, matching and feature identification. The promising evaluation results of ChemGrab, relative to the CEMP reference data set from BioCreative V.5, suggest that it may serve as a foundation for automating identification of chemical mentions in text.

5 Acknowledgement

This work was supported by the United States National Library of Medicine of the United States National Institutes of Health under Award Number R01LM011963.

REFERENCES

Combining the BANNER tool with the DINTO ontology for the CEMP task of BioCreative V.5

Cristóbal Colón-Ruiz, Isabel Segura-Bedmar, and Paloma Martínez

Computer Science, University Carlos III of Madrid, 28911 Madrid, Spain, 
{ccolon, isegura, pmf}@inf.uc3m.es
http://labda.inf.uc3m.es/

Abstract. This paper describes our system for the Chemical Entity Mention in Patents (CEMP) task of BioCreative V.5. The system consists of an adaptation of the BANNER tool, which is based on Conditional Random Fields (CRF) and has provided satisfactory results in the biomedical domain. In addition to the features provided by the tool for the recognition of entities in biomedical texts, a lexical feature is added using the DINTO ontology which will be combined with other ontologies such as ChEMBL and DrugBank.

Key words: BANNER, Conditional Random Fields, DINTO ontology, Chemical entity recognition

1 Introduction

Information related to drugs and chemical compounds constitutes an important pillar for research in the area of biology and biomedical sciences as well as for chemical experts. The enormous number of topics in which the chemical entities are present increases the interest for an efficient access to this information.

Natural language processing (NLP) and text mining technologies are one of the keys to improving access to this type of information from unstructured data such as patents, therefore, the CEMP task of BioCreative V.5 aims to detect chemical entities in medical chemistry patent abstracts automatically.

After an analysis with reference to previous editions of the task [3], it was verified that a large number of participants used supervised learning systems, being the conditional random fields one of the most representative techniques.

Our work seeks to verify the effectiveness of BANNER tool [1] for the recognition of chemical entities and consequently the effectiveness of the features provided by the tool. In addition, we analyze the contribution of the DINTO ontology [2], which contains a large number of chemical entities that can provide more information to the system.
2 Systems description and methods

Taking into account the results provided by CRF-based systems in previous editions, we propose the use of BANNER tool, an entity recognition system, based on conditional random fields and designed to increase domain independence.

Banner has a 3-stage pipeline, where the input is a sentence. During the first process, the sentences are tokenized. Then each token is represented by a series of features which are described below:

- The part of speech which the token forms in the sentence, that is, the process of assigning to each token its grammatical category within the text in relation to the adjacent words.
- The lemmatization of tokens, where each lemma represents the accepted form for all variations of words.
- Prefixes and suffixes up to 2, 3 and 4 characters for each token.
- A subsequence of tokens within the sentence, for this case, bigrams and trigrams are considered in the system.
- The normalization of tokens in word classes, replacing uppercase letters with 'A', lowercase ones with 'a', numbers with '0' and all other characters with 'x'. For example, the mention "C1-alkyl" is a word class of type "A0x0aaaaa".

As a new feature, in addition to using the dictionaries provided by BANNER, we use the DINTO ontology to generate a dictionary of 13294 mentions. This feature indicates whether or not the token is found within the given dictionary. However, in addition to using the resources mentioned, other dictionaries are also used as those provided by the ChEMBL and DrugBank ontologies, as well as combinations of all of them.

For the labeling of the tokens, several IOB tagging schemes have been used (I = inside of a entity, O = outside of a entity, B = beginning of a entity), and finally, the CRF models are trained using the features of all tokens of the sentences provided by the training set.

3 Discussion

The main hypothesis of this work is that the incorporation of a dictionary provided by the DINTO ontology could help to identify and recognize mentions in the set of patents. Another of the main reasons is to adapt the tool of BANNER to the domain of the recognition of chemical entities to verify the independence of the tool based on the results obtained.

To test the tool, as well as the incorporation of features, we have performed a series of experiments with our development set (See Table 1). Because no development set was provided for the current CEMP task, we split the training set provided with 21000 instances into a new training set of 14000 instances and another set of 7000 instances for testing.
In the early experiments, which did not include dictionaries, we can see how BANNER provides, regardless of the scheme used, very similar results apparently without a significant statistical difference. However, the IO scheme seems to provide a slight improvement, moderately increasing the number of TP and decreasing the number of FP and FN.

Next, we train the models using the different dictionaries with the IO tagging scheme. As we can see in the table, the changes also do not show a significant variation, contributing a slight increase of the F1 when we used the combination of the dictionary of DINTO with the one of ChEMBL. Taking into account that this last configuration was done only with the IO tagging scheme, we proceed to check the results with the other schemas, giving as shown in the Table 1, results slightly lower than the initial configuration.

Based on the observed results, we decided to select the following configurations for the runs whose outputs were submitted to the CEMP task:

- Run1: IO schema with DINTO and ChEMBL dictionaries.
- Run2: IO schema with ChEMBL and DrugBank dictionaries.
- Run3: IO schema with only DINTO dictionary.

Table 2 shows the results obtained in test dataset by the runs mentioned above. Our best run has achieved a precision of 88.42%, a recall of 82.64% and an F1 of 85.44%. As can be seen, in general the results achieved are very close between them, but we can also see how the BANNER tool is able to provide good results in the task of recognition of chemical entities, as well as the increase of the results achieved due to the use of two complementary dictionaries.
<table>
<thead>
<tr>
<th>Run</th>
<th>IO 8</th>
<th>dictionaries</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td>IO</td>
<td>DINTO+ChEMBL</td>
<td>0.8842</td>
<td>0.8264</td>
<td>0.8544</td>
</tr>
<tr>
<td>Run 2</td>
<td>IO</td>
<td>ChEMBL+DrugBank</td>
<td>0.8831</td>
<td>0.8251</td>
<td>0.8531</td>
</tr>
<tr>
<td>Run 3</td>
<td>IO</td>
<td>DINTO</td>
<td>0.8799</td>
<td>0.8240</td>
<td>0.8510</td>
</tr>
</tbody>
</table>

**Table 2.** CEMP results on the test dataset. DINTO means that the dictionary used comes from the DINTO ontology; ChEMBL means that the dictionary used comes from the ChEMBL ontology; DrugBank means that the dictionary used comes from the DrugBank ontology.

**Acknowledgments.** This work was supported by the Research Program of the Ministry of Economy and Competitiveness - Government of Spain, (eGovernAbility-Access project TIN2014-52665-C2-2-R).

**References**

An Ensemble Algorithm for Sequential Labelling: A Case Study in Chemical Named Entity Recognition

Chen-Kai Wang¹, Hong-Jie Dai*²,³, Jitendra Jonnagaddala⁴,⁵, Emily Chia-Yu Su*¹

¹Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan, R.O.C., ²Department of Computer Science and Information Engineering, National Taitung University, Taitung, Taiwan, R.O.C., ³Interdisciplinary Program of Green and Information Technology, National Taitung University, Taitung, Taiwan, R.O.C., ⁴Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan, R.O.C., ⁵School of Public Health and Community Medicine, UNSW Sydney, Australia

¹emilysu@tmu.edu.tw; *hjdai@nttu.edu.tw; z3339253@unsw.edu.au

dennisckwang@gmail.com

Abstract. Ensemble methods are learning algorithms that classify new data points by synthesizing the predictions of a set of classifiers. Many methods for constructing ensembles have been proposed such as weighted voting, manipulations of training samples, features, or labels. The paper proposes a novel ensemble algorithm which constructs ensembles by manipulating the label set given to the learning algorithm and then classifies a new dataset by a voting algorithm specifically designed for sequential labelling task. The dataset released in the BioCreative V.5 CEMP (Chemical Entity Mention recognition) task was used to evaluate the performance of proposed algorithm. The results revealed that the proposed algorithm can improve the precision and F-score.

Keywords. Chemical named entity recognition; ensemble method; sequential labelling problem

1 Introduction

In standard supervised sequential labelling task problems, a machine learning algorithm is given with a sequence of training examples of the form \( t_i = \{( o_1, y_1), \ldots, ( o_n, y_n) \} \) and the goal of the learning algorithm is to find a function \( f \) so that \( f( o) = y \). In named entity recognition (NER)
like the BioCreative V.5 CEMP (Chemical Entity Mention recognition) task [1], $o_i$ is typically a vector consisted of features such as the current word, the part-of-speech of the current word, etc., and $|t|$ represents the length of a sentence or the number of tokens. The $y$ values are drawn from a discrete set of labels such as B-Chemical, I-Chemical, and O if the IOB2 scheme [2] is used. Given a set of training examples $t$, a learning algorithm outputs a classifier $c_i$, which can predict the corresponding $y$ values of new $x$ values. An ensemble of classifiers is a set of classifiers $c = \{c_1, c_2, \ldots, c_m\}$ whose individual decisions are combined in a way to classify new examples [3].

Several studies have shown that ensembles are often perform better than the individual classifiers that make them up. Ensembles can be created by several ways. For example, for one training dataset, we can apply different machine learning algorithms or even the same machine learning algorithm with different feature sets to create a set of classifiers. The labels in a dataset can also be manipulated to create different datasets for creating ensemble. For example, TG Dietterich and G Bakiri [4] randomly partitioned the classes appeared in a training set into two subsets and then relabeled the data according to the new tag set to create their ensemble. In this work we proposed an ensemble algorithm which constructs ensembles by manipulating the label set given to the learning algorithm and then classifies a new data by a voting algorithm specifically designed for sequential labelling task.

2 Method

In the BioCreative V.5 CEMP task, the goal is to recognize chemical entity mentions and classify them into seven entity categories including (1) SYSTEMATIC: the systematic names; (2) IDENTIFIERS: database IDs; (3) FORMULA: molecular formula; (4) TRIVAL: trivial, brand, common or generic names of compounds; (5) FAMILY: chemical families that can be associated to chemical structures; (6) MULTIPLE: mentions that correspond to chemicals that are not described by a continuous string of characters; (7) ABBREVIATION: abbreviations and acronyms. In our implementation, the BIESO tag set with fine-grained tokenization [5] were used. The tags, B, I, E, S and O, stand for “Begin”, “Inside”,

“End”, “Single-word”, and “Outside” of a particular category of chemical entities, which results in $7 \times 3 + 6 + 1 = 28$ tags.

**Ensemble Generation**

In this work, an ensemble of nine classifiers were generated. For all the created classifiers, they were based on the same feature sets as described in our previous work [5]. The first two classifiers were trained by using the conditional random fields (CRFs) [6] and maximum entropy (ME) [7] with the original training dataset released by the CEMP task. The other seven classifiers were created by using CRFs with seven relabeled datasets. The relabeled datasets were compiled by duplicating the original dataset into seven copies. For a copy, we kept one entity type and merged the other six categories into one label. The above process repeated seven times to create seven individual dataset; each of which contains only two entity types: one is the original entity type and another represents the other six categories.

**Ensemble Algorithm**

The above ensemble generation step created nine classifiers. We denote the created classifiers as $c$. Assume that the given $x$ is a sequence of observations $\{o_1, \ldots, o_n\}$. For each of the observation $o_i$, a classifier $c_k$ can output the scores for all possible categories. For instance, each of the first two models, $c_1$ and $c_2$, trained with the original dataset by CRF and ME can output 28 scores for each $o$. Given a sequence $x$ of $|t_i|$ tokens, both output a $|t_i| \times 28$ matrix $m^k$ and each entry of the matrix (denoted as $m^k_{(i,j)}$) represents the score given by $c_k$ for the $i$th token in case that it labeled with the $j$th class. However, for the classifier trained with the relabeled dataset, only the columns corresponding to the unmerged label and O tags in the dataset can be filled in. In our implementation, the uniform score was assigned for those columns; the score was evenly distributed among all merged categories.

Now we can average all entries of the $|c|$ score matrices to get an averaged matrix $m$ and apply the maximum function of each row to determine the best outputs of our ensemble, which is similar to the idea of voting-based ensemble learning. Unfortunately, we cannot do this in se-

---

1 After fine-grained tokenization, all MULTIPLE entities consist more than one token.
sequential labelling tasks, because we need considering the labelling sequence to avoid illegal label combinations, such as \{B- SYSTEMATIC, I-FORMULA\}. Instead of applying an additional machine-learning model to fuse the variety of label combination [8, 9], we propose the algorithm shown in Figure 1 to deal with the problem.

The algorithm starts by pruning the possible starting points in the first row. We ignore all classes that cannot be the initial seed for the following calculation. The step removes elements such as E-IDENTIFIER, I-FAMILY, from the first row of the given \(m\) to generate the seed vector \(s\). For each seed label in \(s\), we apply depth-first search (DFS) algorithm to traverse \(m\) to generate all possible sequence combinations from the initial seed label \(s\) and their aggregated scores. The results are returned by the DFS function and the maximum score of all sequences is compared with the best score to determine whether or not to update the current best ensemble result. After finishing the calculation of all seeds, the mapped label sequence of the best ensemble result is returned.

**INPUT:**
A score matrix \(m\): Each entry \(m_{i,j}\) represents the score for the \(i\)th token labeled with the \(j\)th class.
An array \(a\) of \(c\) classes: Each element of the array \(a\) represents the exact class label for the \(j\)th column of \(m\).
An array \(b\) consists of all possible begin labels: The array \(b\) contains all possible begin labels, such as B-ABBREVIATION, S-FORMULA.
**OUTPUT:** An array represents the final predicted labels.

SequentialLabellingEnsemble\((m, a, b)\)
1. \(s \leftarrow \text{Pre-pruning}(m, a, b)\)
2. \(best \leftarrow \text{nil}\)
3. \(\text{for } k \leftarrow 1 \text{ to } \text{len}(s)\)
4. \(dfs_p \leftarrow \text{DFS}(s_k, m, a, b)\)
5. \(\text{max}_dfs \leftarrow \text{max(score}(dfs_p_1), \text{score}(dfs_p_2), \ldots, \text{score}(dfs_p_n))\)
6. \(\text{if } \text{max}_dfs > \text{score}(best)\)
7. \(best \leftarrow \text{max}_dfs\)
8. \(\text{return map(best, a)}\)

Figure 1. The proposed ensemble algorithm.
The computation complexity of the algorithm shown in Figure 1 is high, because of the possible label combinations are huge. Therefore, in our implementation, we set two parameters, $w$ and $nb$, to control the complexity. $w$ controls the depth of the DFS algorithm to traverse and $nb$ controls the number of best sequence hold for calculation.

3 Results and Discussion

We submitted five runs in the CEMP task. Table 1 shows the results. The first run is an ensemble with seven classifiers trained with CRFs and the dataset modified from the training set of the BioCreative V.5 CEMP task. The CRF toolkit we used is CRF++\(^2\). Unfortunately, we failed to use the toolkit with the entire training set of BioCreative V.5 CEMP task to train the model with the seven tag set. Therefore, we used the dataset of BioCreative V CHEMDNER-patents track [10], which is smaller than the dataset released in the CEMP task, to build our classifier for the second run. The third run combined the first run with the classifier trained with ME on the BioCreative V.5 CEMP task. The fourth run combined the second and third run with the classifier trained with CRF on another relabeled dataset of the CEMP corpus. In the dataset, we used one label to represent all seven categories. The final run was an ensemble of the second and third runs.

Table 1. The official evaluation results on the CEMP test test.

<table>
<thead>
<tr>
<th>Run</th>
<th>F-score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ensemble (7 classifiers)</td>
<td>0.8377</td>
<td>0.8567</td>
<td>0.8194</td>
</tr>
<tr>
<td>2. BC V</td>
<td>0.8387</td>
<td>0.8479</td>
<td><strong>0.8296</strong></td>
</tr>
<tr>
<td>3. 1+ME</td>
<td>0.8387</td>
<td>0.8575</td>
<td>0.8207</td>
</tr>
<tr>
<td>4. 2+3+All Merged</td>
<td>0.8389</td>
<td><strong>0.8583</strong></td>
<td>0.8203</td>
</tr>
<tr>
<td>5. 2+3</td>
<td><strong>0.8395</strong></td>
<td>0.8568</td>
<td>0.8228</td>
</tr>
</tbody>
</table>

Although it cannot be directly comparable, we can observe that the proposed algorithm can improve the precision and F-score comparing with the baseline configuration (Run-2). In addition, we observed some bugs in our algorithm implementation after the evaluation phase. We believe

\(^2\) https://taku910.github.io/crfpp/
that the performance of the ensemble results can be much better than the official results as shown in Table 1.

4 Conclusion

In the paper, we give a briefly introduction of our ensemble algorithm specifically designed for the sequential labelling task. Unlike other previous works which employs additional machine-learning models or post-processing rules to combine the results of all classifiers for the sequential label problem, our algorithm considers the confidence scores of each individual classifiers and the possibility of label transition. The results on the CEMP task demonstrates the proposed ensemble algorithm can improve the performance of the individual classifier. In the future, we will compare the performance of the proposed algorithm with that of other traditional ensemble methods. We will also apply different relabeling technique to generate the ensembles and study their performance with the proposed algorithm.

5 Acknowledgment

The study was supported by Ministry of Science and Technology (MOST) 105-2221-E-143-003 and MOST-106-2922-I-038-014.

REFERENCES


Statistical Principle-based Approach for Gene and Protein Related Object Recognition

Po-Ting Lai\textsuperscript{1}, Ming-Siang Huang\textsuperscript{2,3}, Chu-Hsien Su\textsuperscript{4}, Richard Tzong-Han Tsai\textsuperscript{5}, Wen-Lian Hsu\textsuperscript{1}

\textsuperscript{1} Department of Computer Science, National Tsing-Hua University, Taiwan, R.O.C
\textsuperscript{2} Taiwan International Graduate Program Bioinformatics, Institute of Information Science, Academia Sinica, Taiwan, R.O.C
\textsuperscript{3} Institute of Biomedical Informatics, National Yang Ming University, Taiwan, R.O.C
\textsuperscript{4} Institute of Information Science, Academia Sinica, Taiwan, R.O.C
\textsuperscript{5} Department of Computer Science and Information Engineering, National Central University, Taiwan, R.O.C

potinglai@gmail.com
elephant52381@iis.sinica.edu.tw
jason@iis.sinica.edu.tw
thtsai@csie.ncu.edu.tw
hsu@iis.sinica.edu.tw
\textsuperscript{*}Corresponding Author

Abstract. We introduce a Statistical Principle-based Approach (SPBA) for named entity recognition (NER). SPBA is a pattern-based approach. It uses patterns to represent protein names, and uses the semantic labels to map sentence into labeled sentence. NER is then formulated as aligning labeled sentence with patterns. The weights of insertion/deletion/match are learned through logistic regression model in our refactored JNLPBA corpus. We participated in BioCreative V.5 Gene and Protein Related Object (GPRO) task to evaluate the ability of SPBA in processing patent abstracts. Since the NE types and boundaries are slightly different in two corpora. We adjusted SPBA’s NER results by using a linear chain Conditional Random Fields (CRFs) model. In BioCreative V.5 GPRO task, our best configuration achieved an F-score of 73.73% on GPRO type 1; an F-score of 78.66% on combining GPRO type 1 and 2.

Keywords. Named Entity Recognition

1 Introduction

The goal of BioCreative V.5 Gene and Protein Related Object (GPRO) task [3] is that given a patent abstract, a text mining system should (1) identify the boundaries of GPRO (2) and determine whether GPROs can be normalized to database ID or not.
To tackle this task, we developed a pipeline named entity recognition (NER) system that cascaded two NER components, an SPBA-based NER and a CRF-based NER[4].

SPBA is a pattern-based approach for named entity recognition (NER), and it was developed on our refactored JNLPBA corpus [6]. First, our domain experts constructed an entity knowledge base (EKB) by collecting public available dictionaries, like MeSH and UniProt. EKB will be used to map word or phrase into one or more semantic label (called concept). Through EKB, we can generate NE patterns. For instance, “GATA1 erythroid transcription factor” can be labeled as “GATA1GeneSymbol erythroidBiologicalProcess transcription factorProteinEnd”, thus be represented as a pattern [GeneSymbol] [BiologicalProcess] [ProteinEnd]. Using more general concept allows a pattern to identify more NEs. We employed an alignment mechanism in our pattern matching to allow flexible matching, and scored an alignment properly through the logistic regression model to improve accuracy. Different annotation criteria between refactored JNLPBA and GPRO task caused a lower performance in the strict evaluation metric. Thus, we developed a CRFs-based component trained on GPRO training set to adjust the output NE type and boundary. Finally, a GPRO normalization component was developed to map NE into its database ID.

2 Method

Here we describe SPBA in detail, and how we adjust the NE’s type and boundary. First, we construct an entity knowledge base (EKB) consisting of concepts and patterns. Then our pattern matching approach is illustrated for identifying NEs. Furthermore, a logistic regression-based approach is employed to learn the weights of patterns for scoring a matched NE. Moreover, a linear chained CRF-based approach is proposed to adjusting the boundaries of NEs according to the annotation criteria of GPRO task. Finally, a normalization component is introduced for mapping GPRO into database ID.

2.1 Statistical Principle-based Approach
Entity Knowledge Base: A NE is composed of one or more words. Some of these words could be generalized to concepts. For example, “liver cancer” could be generalized to the “Cancer” concept. If we express a NE as a set of sequence of concepts (called pattern), these patterns are likely to match unseen instances of that NE type. Therefore, EKB is constructed by collecting the concept set from publicly available biological databases shown in Table 1.

Pattern Generation: To generate pattern, we first employ prefix-tree matching to label all NEs in the training data by using the EKB. Then, unlabeled words are removed, and the remaining label sequence is called a pattern. For example, given a NE “USF-related transcription factor” and two EKB concepts, GeneSymbol and ProteinEnd. It will be labeled as: “USF\text{GeneSymbol} -related \text{transcription factor}\text{proteinEnd}”. Then [GeneSymbol] [ProteinEnd] will be the pattern. Since a NE may be labeled in more than one way, generating more than one pattern, we only keep the pattern with the highest ratio of labeled words to total words.

Pattern Matching: After pattern generation, the patterns will be used to recognize candidate NEs by an local alignment algorithm.

Logistic Regression: We use the logistic regression (LR) model [7] to learn weights for insertion, match and deletion. The score of an alignment of a pattern $p$ and a labeled sentence $l$, $\text{Similarity}(p,l)$ is

<table>
<thead>
<tr>
<th>Concept</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiologicalProcess</td>
<td>MeSH term</td>
</tr>
<tr>
<td>CellLineSymbol</td>
<td>CLDB [1]</td>
</tr>
<tr>
<td>CellTypeSymbol</td>
<td>ExPAsy [2]</td>
</tr>
<tr>
<td>Chemical</td>
<td>ChEBI [5]</td>
</tr>
<tr>
<td>Chromosome</td>
<td>Regular expression</td>
</tr>
<tr>
<td>Disease</td>
<td>MeSH term</td>
</tr>
<tr>
<td>DNASymbol</td>
<td>Entrez</td>
</tr>
<tr>
<td>Morphology</td>
<td>Manual</td>
</tr>
<tr>
<td>OrganTissue</td>
<td>SWISS-PROT</td>
</tr>
<tr>
<td>ProteinSymbol</td>
<td>Entrez and UniProt</td>
</tr>
<tr>
<td>RNAsymbol</td>
<td>Entrez</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>MeSH term</td>
</tr>
<tr>
<td>Structure</td>
<td>ExPAsy</td>
</tr>
</tbody>
</table>
calculated by the following formula:

\[
\text{similarity}(p, l) = \sum_{i} \lambda_M (\text{the } i\text{th matched word}) + \sum_{j} \lambda_D (\text{the } j\text{th deleted word}) \\
+ \sum_{k} \lambda_I (\text{the } k\text{th inserted word})
\]

where \( \lambda_M (w) \) is the weight for the feature in which the matched word is \( w \);
\( \lambda_D (w) \) is the weight for the feature in which the deleted word is \( w \);
\( \lambda_I (w) \) is the weight for the feature in which the insertion word is \( w \);

Then, \( \text{Similarity}(p, l) \) is transformed into a real number ranging from 0 to 1 by the sigmoid function:

\[
h(p, l) = \frac{1}{1 + e^{-\text{Similarity}(p, l)}}
\]

\( h \) is considered as the alignment score of \( p \) and \( l \).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
<td>NULL</td>
<td>NULL</td>
<td>DT</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>promoter</td>
<td>0</td>
<td>a</td>
<td>pr</td>
<td>er</td>
<td>NN</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>sequence</td>
<td>0</td>
<td>a</td>
<td>se</td>
<td>ce</td>
<td>NN</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>of</td>
<td>0</td>
<td>a</td>
<td>of</td>
<td>of</td>
<td>IN</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
<td>a</td>
<td>th</td>
<td>he</td>
<td>DT</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>human</td>
<td>0</td>
<td>a</td>
<td>hu</td>
<td>an</td>
<td>JJ</td>
<td>B</td>
<td>GPRO_TYPE_1</td>
</tr>
<tr>
<td>p1</td>
<td>0</td>
<td>a1</td>
<td>p1</td>
<td>p1</td>
<td>NN</td>
<td>I</td>
<td>E-GPRO_TYPE_1</td>
</tr>
<tr>
<td>TNF</td>
<td>1</td>
<td>A</td>
<td>TN</td>
<td>NF</td>
<td>NN</td>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>-</td>
<td>NULL</td>
<td>NULL</td>
<td>HYPH</td>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>A</td>
<td>NULL</td>
<td>NULL</td>
<td>HYPH</td>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>gene</td>
<td>0</td>
<td>a</td>
<td>ge</td>
<td>ne</td>
<td>NN</td>
<td>E</td>
<td>O</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>a</td>
<td>is</td>
<td>is</td>
<td>VBZ</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>provided</td>
<td>0</td>
<td>a</td>
<td>pr</td>
<td>ed</td>
<td>VBN</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>.</td>
<td>NULL</td>
<td>NULL</td>
<td>.</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**Fig. 1.** An example of CRF features.
LR is applied as follows. Initially, all training sentences are labeled by using the EKB. The set of labeled training sentences is referred to as $L$. The size of $L$ is greater or equal to the original set of training sentences because one sentence may have more than one ways of labeling. Then, patterns of each NE type are used to identify candidate NEs for each training sentence through the alignment. Moreover we collect the sets of true positive (TP) and false positive (FP) labeling results, referred to as $E_{TP}$ and $E_{FP}$, respectively. Finally, we employ logistic regression model to learn feature weights.

2.2 Adjusting and Normalizing NE

**NE Adjustment:** There are two differences between SPBA’s and GPRO’s annotations. The first one is NE types. SPBA was trained on the refactored JNLPBA whose NE types are cell line, cell type, DNA, protein and RNA. However, GPRO task used two NE types, GPRO type 1 and GPRO type 2. GPRO type 1 denotes that NE can be normalized into database ID; GPRO type 2 denotes that NE cannot be normalized. The second one is NE boundaries. The curators of the refactored JNLPBA preferred to annotate longer phrase/chunk as NEs, but GPRO seems prefer to annotate the phrase/chunk which can exactly match the database’s official name. Thus, we found that GPRO NEs were usually substrings of SPBA’s NEs. Therefore, we used a linear chained CRF model to tackle this problem. The adjustment of NE types and boundaries are formulated as a NER problem. Given a sentence and SPBA’s NE as feature, to predict the GPRO’s NE. A subset of NERBio’s features [8] including word, POS, affix, orthographical, word shape and POS features are used. Fig. 1 shows an example of our features.

**GPRO Normalization:** Another way to determine GPRO type of NE can be done by checking whether NE can be mapped into database ID. Therefore, in another configuration, we used some normalization rules [9] to retrieve the ID of a NE. For examples, expanding both dictionary names and NEs like converting to lower cases; removing the symbols; removing the named entity suffix “s”.
3 Experiment Results and Future Work

We participated in BioCreative V.5 GPRO Task [3]. The official performances of our submissions on GPRO test set are shown in Table 2. The task uses a strict F1-measure evaluation metric. The column of “Use GPRO dict” means that GPROs of the training set are used to extend our normalization dictionary. The column of “Use normalization” means that we use the normalization component. Our best configuration achieves an F-score of 73.73% on GPRO type 1; an F-score of 78.66% on combining GPRO type 1 and 2. To our surprise, the results shows that the Config. 2 slightly outperform the Config. 1 which uses the normalization component. In the future, we would like to release a restful web service of our pipeline system.

<table>
<thead>
<tr>
<th>Config</th>
<th>Use GPRO dict</th>
<th>Use normalization</th>
<th>GPRO Type 1</th>
<th>Merge GPRO Type 1 and 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F-score</td>
</tr>
<tr>
<td>1</td>
<td>O</td>
<td>68.69%</td>
<td>78.24%</td>
<td>73.15%</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>66.53%</td>
<td>82.68%</td>
<td>73.73%</td>
</tr>
</tbody>
</table>

Table 2. The performances of our submissions

REFERENCES

Tagger: BeCalm API for rapid named entity recognition

Lars Juhl Jensen*

Disease Systems Biology Program, Novo Nordisk Foundation Center for Protein Research, Faculty of Health and Medical Sciences, University of Copenhagen, Blegdamsvej 3B, DK-2200 Copenhagen N, Denmark

lars.juhl.jensen@cpr.ku.dk

Abstract. Most BioCreative tasks to date have focused on assessing the quality of text-mining annotations in terms of precision of recall. Interoperability, speed, and stability are, however, other important factors to consider for practical applications of text mining. The new BioCreative/BeCalm TIPS task focuses purely on these. To participate in this task, I implemented a BeCalm API within the real-time tagging server also used by the Reflect and EXTRACT tools. In addition to retrieval of patent abstracts, PubMed abstracts, and PubMed Central open-access articles as required in the TIPS task, the BeCalm API implementation facilitates retrieval of documents from other sources specified as custom request parameters. As in earlier tests, the tagger proved to be both highly efficient and stable, being able to consistently process requests of 5000 abstracts in less than half a minute including retrieval of the document text.

Keywords. Named Entity Recognition; Application Programming Interface; Web Service; Performance Evaluation

1 Introduction

BioCreative and other shared tasks in the biomedical text-mining community have over the years played a key role in progressively improving text-mining methods, in particular for named entity recognition (NER). Most BioCreative tasks have focused purely on evaluating the precision and recall (1,2), with the BioC interoperability task (3) and the interactive annotation task (IAT) (4) being notable exceptions. However, as illustrated by the latter two tasks, whereas precision and recall are obviously important factors, they are far from the only factors that matter when using text mining in practice. Interoperability, speed, and stability are other very important factors; the new Technical In-
I participated in the BioCreative V IAT (4) with the interactive annotation tool, EXTRACT, which helps curators find and extract standard-compliant terms for annotation of metagenomic records and other samples (5). Behind its web-based user interface, the system makes use of the same real-time tagger for NER as the augmented browsing tool Reflect (6). The core NER engine was designed from the ground up with speed in mind and is capable of tagging thousands of PubMed abstracts per second per CPU core (7). This makes it ideally suited for large-scale and real-time applications, such as the TIPS task.

Here, I present a BeCalm API for the NER tagger underlying the EXTRACT (5) and Reflect (6). The system delivered a total turnaround time of about 1 second for small requests, and was able to process approximately 5,000–10,000 abstracts per minute for larger batch requests. Notably, the vast majority of this time was spent on retrieving the document text rather than actual processing of it; to make the server faster, it would thus be necessary to locally cache the documents, which was explicitly not permitted in the TIPS task.

2 Materials and Methods

Dictionaries used for NER and normalization

The server uses a combination of previously published dictionaries to recognize six of the types of entities accepted by the BeCalm server and normalize them to identifiers from databases and ontologies. These are a subset of the entity types used in EXTRACT v2 (5).

For annotation of gene/protein names, the tagger uses a dictionary covering the 9.6 million protein-coding genes from 2031 organisms included in STRING v10.5 (8) as well as ncRNAs from the RAIN database (9). Unlike many NER systems, the BeCalm API makes a distinction between genes and their protein products. Because the STRING database is locus-based, i.e. it does not distinguish between splice isoforms, and because ncRNAs are also included, I chose to use the type GENE for these annotations and to not support the PROTEIN annotation type. All recognized names are disambiguated to their respective STRING or RAIN identifiers, which are derived from the Ensembl (10), RefSeq (11), and miRBase (12) databases.
Annotations of the type CHEMICAL are made using a dictionary comprised of small-molecule compounds from the PubChem database (13), which was developed and used for recognition of chemical names in STITCH v5 (14). All annotations of chemicals are normalized to PubChem compound identifiers.

The tagger makes annotations of the type ORGANISM using an updated version of the dictionary of the SPECIES/ORGANISMS tagger (7). The dictionary was constructed based on NCBI Taxonomy (10), and all annotations are thus normalized to NCBI taxon identifiers.

For SUBCELLULAR_STRUCTURE, TISSUE_AND_ORGAN, and DISEASE the tagger uses the dictionaries created as part of the COMPARTMENTS (15), TISSUES (16), and DISEASES (17) database, respectively. These were constructed from Gene Ontology (18), Brenda Tissue Ontology (19), and Disease Ontology (20), identifiers from which are used for normalization of the annotations.

The version of the dictionary used by Tagger for the TIPS task has been deposited on FigShare (doi:10.6084/m9.figshare.4578292). The reduced dictionary used by PiTagger has also been deposited on Figshare (doi:10.6084/m9.figshare.4635175). The latest version of the dictionary, which is used by the production server, is available for download at http://download.jensenlab.org/tagger_dictionary.tar.gz.

**Named entity recognition software**

The core of the NER system is a highly optimized dictionary-based tagger engine, implemented in C++. It is able to perform flexible matching of a dictionary with millions of names against thousands of abstracts per second per CPU core (7). The tagger is furthermore inherently thread safe, for which reason a single instance of the tagger can easily handle many parallel requests. These properties make it an excellent starting point for building a real-time service that can handle large requests as required for TIPS task.

Although the TIPS task does not assess the quality of the annotations, it is worth noting that the speed of the tagger was not achieved by sacrificing quality. The quality of the tagging results for organism names was previously evaluated on gold-standard corpora and found to be comparable to the best methods (7,21). The NER quality has not been benchmarked directly for chemicals, genes, tissues, and diseases has not been benchmarked directly; however, these NER components
have shown to give good results when used as the basis for association extraction (8,9,13,15–17).

The tagger software is open source and available at https://bitbucket.org/larsjuhljensen/tagger/. It can be used either as a command-line tool or as a Python module. It is also distributed as a Docker container at https://hub.docker.com/r/larsjuhljensen/tagger/.

**BeCalm API implementation and hosting**

I implemented the BeCalm API itself in Python and runs as a module under an in-house web service framework. The framework uses multiple queues and thread pools to simultaneously run several compute-intensive requests in parallel (e.g. getAnnotations requests) and be responsive to smaller requests (e.g. getStatus). The API code accesses a single instance of the tagger engine through its Python module, which has the complete dictionary preloaded in memory.

The main tagger runs on a single server with one Intel Xeon E5-2620 2.4 GHz CPU and 256 GB of RAM. This server also runs many other resources and databases related to text mining, including EXTRACT (5), SPECIES/ORGANISMS (7), COMPARTMENTS (15), TISSUES (16), and DISEASES (17). This server is physically hosted at the high-performance computing facility Computerome and is from hereon referred to as Tagger.

To test the influence of the performance of actual document tagging vs. overhead associated with fetching of document texts, I ran a second instance of the tagger on a Raspberry Pi 3 with a 1.2 GHz quad-core ARM Cortex-A53 and 1 GB and RAM. Due to the limited memory, this instance runs with a reduced dictionary; however, it should be noted that tagging speed is largely independent of dictionary size because the tagging algorithm is based on hash lookups (7). This instance was hosted over my home internet connection (60 Mbit/s download, 25 Mbit/s upload) and is in the following referred to as PiTagger.

3 Results and Discussion

**Rapid annotation of biomedical entities**

To test the speed of Tagger and PiTagger when accessed through the BeCalm API, I submitted private requests for tagging of 1, 10, 100, 1000, and 5000 abstracts from the abstract and patent servers via the BeCalm web interface.
Table 1. **Performance of the taggers.** For small requests the total turnaround time is ~1 second. Larger requests take an extra 5–10 seconds per 1000 abstracts to be processed on Tagger. Notably, most of this time is spent on retrieving the document texts from document identifiers, whereas the actual NER step takes only about 20% of the total time. This is reflected in the fact that the PiTagger, which runs on a Raspberry Pi 3, takes only about 50% longer to process large requests.

<table>
<thead>
<tr>
<th># Documents</th>
<th>Tagger: total time (seconds)</th>
<th>PiTagger: total time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abstract server</td>
<td>Patent server</td>
</tr>
<tr>
<td>1</td>
<td>0.87±0.32</td>
<td>0.84±0.32</td>
</tr>
<tr>
<td>10</td>
<td>0.98±0.30</td>
<td>0.83±0.26</td>
</tr>
<tr>
<td>100</td>
<td>1.89±0.29</td>
<td>1.48±0.27</td>
</tr>
<tr>
<td>1000</td>
<td>11.31±0.68</td>
<td>6.02±0.37</td>
</tr>
<tr>
<td>5000</td>
<td>52.18±2.76</td>
<td>26.67±1.16</td>
</tr>
</tbody>
</table>

All settings except from the number of documents to tag were left at their default values. Each of the five sizes of tagging requests was repeated five times at four different timepoints, giving a total of 20 observations of the total time required for tagging for each size of request from each document source on each of the two tagger servers. These results are summarized as means and standard deviations in Table 1.

Neither Tagger nor PiTagger suffered any errors or slowdowns during these tests, despite the Tagger server hosting multiple other resources and the PiTagger running on minimal hardware. This shows that the software is not only fast but also stable. This is unsurprising since all parts except the BeCalm API-specific code have been used in a production setting for several years.

In summary, the Tagger speed tests showed that there is a constant overhead of about 1 second on all tagging requests, which dominates the picture up to tagging of about 100 patent abstracts. For larger re-
quests, the service takes ~5 and ~10 seconds more per 1000 patent abstracts and PubMed abstracts, respectively. This difference is presumably explained by PubMed abstracts being, on average, about twice as long as patent abstracts. Notably, the vast majority of the time is spent on fetching the document texts, with only about ~20% of time being spent on actual processing. Although explicitly not permitted in the TIPS task, local storage or caching of documents on the annotation server would thus be an attractive future feature.

To further test and illustrate that retrieval of document texts is the main bottleneck, I configured a second copy of the tagger code, PiTagger, to run on a Raspberry Pi 3. For small requests, the total time is indistinguishable between Tagger and PiTagger, and even for large requests PiTagger takes only about 50% longer than Tagger (Table 1). This is the case despite the service running only one thread per request, thus utilizes only a quarter of the compute power of a Raspberry Pi 3 in these tests. The PiTagger did not participate in the full official TIPS evaluation.

The total tagging time for the official TIPS requests was in the beginning consistently longer than for the private requests reported in Table 1, which were submitted during the same weeks. Monitoring the tagging services during TIPS requests revealed that actual document processing was as fast as always. In light of the results above, I assume that this slowdown was due to the fetching of documents taking longer in the official tests, because all participants simultaneously send requests to the central document servers.

**Extending the BeCalm API**

The BeCalm API in its current form has certain design constraints that limit from the flexibility and thereby usefulness of the annotation servers. Firstly, document text is not submitted as part of the request, but must instead be fetched from designated sources based on the submitted document identifiers. Secondly, the results cannot be returned directly to the end user, but must be returned to the central BeCalm server. Through creative use of the `custom_parameters` part of the request, I have circumvented both of these constraints.

Instead of hardwiring the annotation server to use only the abstract and patent servers provided by BeCalm, the relationships between `source` and server URL are specified within a `servers` subsection of `custom_parameters`. This enables end users to obtain the tagging results
for any desired documents, provided they make the documents avail-
able through an API compatible with the one used by the BeCalm doc-
ument servers.

Similarly, the annotation server is not hardwired to return the annotation
results to the BeCalm server. Instead, the `saveAnnotations` request
will be made to the URL specified in as `apiurl` in the `custom_parameters`
section. This allows end users to set up their own server to receive the results directly, if they so wish.

4 Acknowledgment

This work was supported by the Novo Nordisk Foundation [NNF14CC0001]. Thanks to Helen V. Cook for improvements to the source code and documentation of the tagger, Sune Pletscher-Frankild for developing the Python framework under which the BeCalm API runs, and the organizers of the 3rd Biomedical Linked Annotation Hackathon (BLAH3), where the BeCalm API was developed.

REFERENCES

MER: a Minimal Named-Entity Recognition Tagger and Annotation Server

Francisco M. Couto*, Luis F. Campos, and Andre Lamurias
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. Named-Entity Recognition (NER) aims at identifying the fragments of a given text that mention a given entity of interest. This manuscript presents our Minimal named-Entity Recognizer (MER), designed with flexibility, autonomy and efficiency in mind. To annotate a given text, MER only requires a lexicon (text file) with the list of terms representing the entities of interest; and a GNU Bash grep and awk tools.

MER was deployed in a cloud infrastructure using multiple Virtual Machines to work as an annotation server and participate in the Technical Interoperability and Performance of annotation Servers (TIPS) task of BioCreative V.5. Preliminary results show that our solution processed each document (text retrieval and annotation) in less than 3 seconds on average without using any type of cache. MER is publicly available in a GitHub repository (https://github.com/lasigeBioTM/MER) and through a RESTful Web service (http://labs.fc.ul.pt/mer/).

Key words: Named-Entity Recognition, Annotation Server, Text Mining, Biomedical Ontologies, Lexicon

1 Introduction

Named-Entity Recognition (NER) aims at identifying mentions of entities in a given text. To define which type of entities to recognize, state-of-the-art tools usually require as input a training corpus. Their performance depends on the availability of a large corpus with an accurate and comprehensive set of annotations, which is usually arduous to create and maintain. Many other tools require only as input a lexicon consisting in a list of terms within some domain, for example a medical lexicon. A lexicon is normally much easier to find or to create. The input text is then matched against the terms in the lexicon.

We propose a Minimal named-Entity Recognizer (MER) designed with flexibility, autonomy and efficiency in mind. MER only requires a lexicon as input, in the form of a text file, in which each line contains a term representing a named-entity to recognize. MER makes really easy the addition of new lexicons. MER is not only minimal in terms of the input but also in its implementation, which was reduced to a minimal set of components and software dependencies. MER

* Corresponding author: fcouto@di.fc.ul.pt
is then composed of just two components, one to process the lexicon (offline) and another to produce the annotations (on-the-fly). Both were implemented as a GNU Bash shell script¹, mainly for two reasons: i) efficiency, due to its direct access to high performance text and file processing tools, such as grep and awk; and ii) portability, since the scripts can be executed in any Unix-like operating systems (common in most servers) without requiring any additional software or modifications.

MER was deployed in a cloud infrastructure to work as an annotation server and participate in the Technical Interoperability and Performance of annotation Servers (TIPS) task of BioCreative V.5 [2]. This participation allowed us to assess the flexibility, autonomy and efficiency of MER in a realistic scenario. The preliminary results show that our annotation server achieved good reliability and performance indicators.

MER is publicly available in a GitHub repository². The repository contains a small tutorial to help the user start using the program and test it. The remainder of this article will detail the MER tool, and how it was incorporated in our annotation server to participate in TIPS. We end by analyzing and discuss the preliminary results and present future directions.

## 2 MER

### 2.1 Input

Before being able to annotate any text, MER requires a lexicon with the list of terms to match. So, if for example we want to recognize terms that are present in ChEBI³, the user just has to collect all ChEBI terms and store them in a text file, containing in each line one term representing a ChEBI entity. Figure 1 presents an example where four ChEBI compounds are represented by a list of terms based on their ChEBI’s name. Currently MER is only performing NER, in the future we may allow the option to include an accession number to each line in the file in order to also perform entity linking.

---

¹ [https://www.gnu.org/software/bash/](https://www.gnu.org/software/bash/)
² [https://github.com/lasigeBioTM/MER](https://github.com/lasigeBioTM/MER)
³ [https://www.ebi.ac.uk/chebi/](https://www.ebi.ac.uk/chebi/)
Fig. 2. Each block represents the content of each of the four files created after pre-processing the input file shown in Figure 1.

== one-word (. . . word1.txt) ==
α. maltose

== two-word (. . . word2.txt) ==
nicotinic acid

== more-words (. . . words.txt) ==
nicotinic acid d. ribonucleotide

== first-two-words (. . . words2.txt) ==
nicotinic acid

Fig. 3. Example of a given sentence to be annotated (first line), and its one-word and two-word patterns created by MER.

α–maltose and nicotinic acid was found, but not
nicotinic acid D-ribonucleotide

α. maltose | nicotinic | acid | d. ribonucleotide | found | nicotinic | acid

α. maltose nicotinic | acid | d. ribonucleotide | found | nicotinic
| nicotinic acid | d. ribonucleotide | found | nicotinic | acid

Fig. 4. Output example of MER for the sentence in Figure 3 and the lexicon in Figure 1

0 9 α–maltose
14 28 nicotinic acid
65 79 nicotinic acid
14 45 nicotinic acid D-ribonucleotide

2.2 Pre-Processing

Each lexicon has to go through two pre-processing steps. The first step splits the lexicon in three files containing the terms composed by one (one-word), two (two-word) and three or more words (more-words). The second step creates a fourth file containing the first two words (first-two-words) of all the terms in the more-words file. During the above steps, MER makes the following minor modifications to the terms: convert all text to lowercase; contiguous white spaces are replaced by one white space; full stops are removed; leading and trailing white spaces are removed; and all special characters are replaced by a full stop. Since some special characters may cause matching problems, MER assumes that all the special characters (characters that are not alphanumeric or a whitespace, for example, hyphens) can be matched by any other character, so these characters are replaced by a full stop, like in regular expressions. Figure 2 presents the contents of each of the four files created using the terms shown in Figure 1. Note that the word acid-adenine was replaced by acid.adenine, and the last file presents the first two words of each entry in the third file.
2.3 Recognition

Given an input text, and the name of the lexicon already pre-processed, the goal is to identify which terms of the lexicon are mentioned in the text. The first step of MER is to apply the same minor modifications to the input text as described in the Pre-Processing section, but also remove stop words, and words with less than 3 characters. This will result in a processed input text derived from the original one. Note that MER only recognizes direct matches, if lexical variations of the terms are needed they have to be added in the lexicon, for example by using a stemming algorithm.

A common solution would be to apply grep directly to the input text. However, the execution time is proportional to the size of the lexicon, since each term of the lexicon will correspond to an independent pattern to match. To optimize the execution time we inverted this solution, i.e. we use the words in the processed input text as patterns to be matched against the lexicon file. Since the number of words in the input text is much smaller than the number of terms in the lexicon, grep has much less patterns to match. For example, finding the pattern `nicotinic acid` in the two-word chemical file created for TIPS is more than 100 times faster than using the common solution. This requires the creation of two alternation patterns: i) one-word pattern, with all the words in the input text; and ii) two-word pattern, with all the consecutive pairs of words in the input text. Figure 3 shows an example of these two patterns.

Next, MER creates three background jobs to match the terms composed of: i) one word, ii) two words, and iii) three or more words. The one-word job uses the one-word pattern to find matching terms in the one-word file. Similarly for the two-word job, that uses the two-word pattern and file. The last job uses the two-word pattern to find matches in the two-first-word file, and the resulting matches are then used as a pattern to find terms in the more-words file. The last job is less efficient since it executes grep twice, however the resulting list of matches with the two-first-word file is usually small, so the second execution is negligible. In the end, each job will create a list of matching terms that are mentioned in the input text.

Since the processed input text cannot be used to find the exact position of the term, MER uses the list of matching terms to find their exact position in the original input text. MER uses awk to find the multiple instances of each term in the original input text. The awk tool has the advantage of working well with UTF-8 characters that use more than one byte, in opposition to grep that just counts the bytes to find the position of a match. MER provides partial overlaps, i.e. a shorter term may occur at the same position as a longer one, but not full overlapping matches (same term in the same position). We also developed a test suite to refactor the algorithm with more confidence that nothing is being done incorrectly. The test suite is available in the GitHub repository branch dedicated to development.

Figure 4 shows the output of MER when using as input text the sentence in Figure 3, and the lexicon of Figure 1. Note that `nicotinic acid` appears twice at

\[ \text{https://github.com/lasigeBioTM/MER/tree/dev} \]
position 14 and 65, as expected, without affecting the match of nicotinic acid
D-ribonucleotide.

3 Annotation Server

TIPS is a novel task in BioCreative aiming at the evaluating the performance of
NER web servers, based on reliability and performance metrics. The entities to
be recognized were not restricted to a particular domain.

The web servers had to respond to single document annotation requests. The
servers had to be able to retrieve the text from documents in the patent server,
the abstract server and PubMed, without using any kind of cache for the text or
for the annotations. The annotations had to be provided in, at least, one of the
following formats: BeCalm JSON, BeCalm TSV, BioC XML or BioC JSON.

3.1 Lexicons

The first step to participate in TIPS was to select the data sources from which we
could collect terms related with the following accepted categories: Cell line and
cell type: Cellosaurus\(^5\); Chemical: HMDB\(^6\), ChEBI\(^7\) and ChEMBL\(^8\); Disease:
Human Disease Ontology\(^9\); miRNA: miRBase\(^10\); Protein: Protein Ontology\(^11\);
Subcellular structure: cellular component aspect of Gene Ontology\(^12\); Tissue
and organ: tissue and organ subsets of UBERON\(^13\).

All these data files suffered a post-extraction processing which consisted in
lowercasing all terms, deleting leading and trailing white spaces and removing

---

\(^5\) http://web.expasy.org/cellosaurus/
\(^6\) http://www.hmdb.ca/
\(^7\) https://www.ebi.ac.uk/chebi/
\(^8\) https://www.ebi.ac.uk/chembl/
\(^9\) http://www.obofoundry.org/ontology/doid.html
\(^10\) http://www.mirbase.org/
\(^11\) http://www.obofoundry.org/ontology/pr.html
\(^12\) http://www.geneontology.org/
\(^13\) http://uberon.github.io/
repeated terms. Since overlapping annotations were not allowed, we created another lexicon containing terms that appeared on more than one of the other lexicons. The terms matched to this lexicon were considered to be of the category Unknown, as suggested by the organization. The software to extract the list of terms from the above data sources can be found in the GitHub repository branch dedicated to TIPS\textsuperscript{14}.

Figure 5 shows the number of terms, the number of words, and the number of characters of each lexicon created. MER was therefore recognizing more than 1M terms composed of more than 2M words and more than 25M characters. All lexicons are available for reuse as a zip file\textsuperscript{15}.

3.2 Input and Output

We adapted MER to provide the annotations in the BeCalm TSV format. Thus, besides the input text and the lexicon, MER had to receive also the document identifier and the section as input. In Figure 6, the document identifier is 1 and section is A. The score column is calculated by $1 - 1/\ln(nc)$, where $nc$ represents the number of characters of the recognized term. This is based on the assumption that longer terms are less ambiguous, and in that case the match should have a higher confidence score. Note that MER only recognizes terms with three or more characters, so the minimum score is 0.08976 and the score is always lower than 1. An instance of MER with this output format and using the lexicons described above is available through a RESTful Web service\textsuperscript{16}.

We used jq\textsuperscript{17} a command-line JSON processor to parse the requests. The download of each document was implemented using the popular cURL tool, and we developed a specific parser for each data source to extract the text to be annotated. The parsers are also available at the TIPS branch\textsuperscript{18}.

\begin{itemize}
  \item \textsuperscript{14} https://github.com/lasigeBioTM/MER/tree/biocreative2017/data_parsers
  \item \textsuperscript{15} https://github.com/lasigeBioTM/MER/raw/biocreative2017/data/TIPS_MER_lexicons_Jan2017.zip
  \item \textsuperscript{16} http://labs.fc.ul.pt/mer/
  \item \textsuperscript{17} https://stedolan.github.io/jq/
  \item \textsuperscript{18} https://github.com/lasigeBioTM/MER/tree/biocreative2017/external_services
\end{itemize}
Table 1. Annotation server performance values at April 20, 2017. MTDV (mean time in seconds per document volume) = Total processing time(s)/sum of document sizes (bytes). MPDV (total predictions per document volume) = Total predictions/sum of document sizes (bytes).

<table>
<thead>
<tr>
<th></th>
<th>Patent Server</th>
<th>Abstract Server</th>
<th>PubMed</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total requests</td>
<td>88,556</td>
<td>135,938</td>
<td>92,811</td>
<td>317,305</td>
</tr>
<tr>
<td>Total predictions</td>
<td>1,388k</td>
<td>4,035k</td>
<td>2,766k</td>
<td>8,189k</td>
</tr>
<tr>
<td>Total processing time</td>
<td>2d 01h</td>
<td>4d 18h</td>
<td>3d 19h</td>
<td>10d 14h</td>
</tr>
<tr>
<td>Mean predictions/document</td>
<td>15.6</td>
<td>29.7</td>
<td>29.8</td>
<td>22.4</td>
</tr>
<tr>
<td>Mean processing time/document (s)</td>
<td>2.02</td>
<td>3.03</td>
<td>3.57</td>
<td>2.90</td>
</tr>
<tr>
<td>MTDV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.00238</td>
</tr>
<tr>
<td>MPDV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0184</td>
</tr>
</tbody>
</table>

3.3 Infrastructure

Our annotation server was deployed in a cloud infrastructure composed of three Virtual Machines (VM). Each VM had 8GB of RAM and 4 CPUs @ 1.7 GHz, using CentOS Linux release 7.3.1611 (Core) as the operating system. We selected one VM (primary) to process the requests, distribute the jobs, and execute MER. The other two VMs (secondary) just execute MER. We installed the NGINX HTTP server running CGI scripts given its high performance when compared with other web servers [3]. We also used the Task Spooler tool to manage and distribute within the VMs the jobs to be processed.

The server is configured to receive the REST API requests defined in the BeCalm API documentation. Each request is distributed to one of the three VMs according to the least-connected method of NGINX. When a `getAnnotations` request is received, the server first downloads the documents from the respective sources, and then processes the title and abstract of each document in the same VM. Two jobs are spawned in background, corresponding to the title and abstract. Each annotation server handles all the entity types mentioned in Figure 5, spawning a separate job for each entity type. The name of the entity type is added as another column to the output of Figure 4. These jobs run in parallel since they are independent from each other and the output of each job can be easily merged into a final TSV output file. When a job finishes processing, a script checks if the other jobs associated with the same requests have also finished processing. If that is the case, then the results of every job are concatenated and sent back to BeCalm using the `saveAnnotations` method.

3.4 Results

Table 3.3 presents the performance values for our annotation server available at the BeCalm web interface on April 20, 2017. Our minimal annotation server was able to efficiently process the documents by taking less than 3 seconds on average. 

---

\[\text{http://vicerveza.homeunix.net/~viric/soft/ts/}\]
without using any type of cache. We note that all documents, irrespectively of
the source, were annotated using all the entity types presented in Section 3.1.

We compared the time necessary to process the same sentence on the same
hardware using MER and a more complex machine learning system, IBEnt [1],
using the sentence of Figure 3. While IBEnt took 8.25 seconds to process the
sentence, MER took only 0.098 seconds. Although IBEnt is optimized for batch
processing, therefore reducing the time per document as the number of docu-
ments increases, MER is still 84 times faster than IBEnt in this experiment.
Thus, besides being easy to install and configure, MER is also a highly efficient
and scalable NER tagger.

4 Conclusions

We presented MER a minimal NER tagger that was developed with the con-
cepts of flexibility, autonomy and efficiency in mind. MER is flexible since it
can be extended with any lexicon composed of a simple list of terms. MER is
autonomous since it only requires a GNU Bash shell with awk and grep tools,
which are omnipresent in almost any Unix-like operating systems. MER is effi-
cient since it takes advantage of the high-performance capacity of grep as a file
pattern matcher.

MER was integrated in an annotation server deployed in a cloud infra-
structure for participating in the TIPS task of BioCreative V.5. Our server was
fully developed in-house with minimal software dependencies and is open-source.
Without using any kind of cache, our server was able to process each document
in less than 3 seconds on average. In the future, we intend to implement the en-
tity linking functionality in MER, without undermining its flexibility, autonomy
and efficiency.

Acknowledgments. This work was produced with the support of the Por-
tuguese National Distributed Computing Infrastructure (http://www.incd.pt).
This work was supported by FCT through funding of the LaSIGE Research Unit,
ref. UID/CEC/00408/2013

References

1. Lamurias, A., Filipe, L., Couto, F.M.: IBEnt: Chemical Entity Mentions in Patents
   using CheBI. Proceedings of the BioCreative V.5 Challenge Evaluation Workshop
   (2017)
   M., Lourenço, A.: Benchmarking biomedical text mining web servers at BioCreative
   V.5: the technical Interoperability and Performance of annotation Servers - TIPS
   12–21 (2017)
   2008(173), 2 (2008)
SIA: Scalable Interoperable Annotation Server

Johannes Kirschnick and Philippe Thomas

DFKI Language Technology Lab
DFKI Intelligent Analytics for Massive Data Lab
Alt-Moabit 91c, 10559 Berlin, Germany

Abstract. Recent years showed a strong increase in biomedical sciences and an inherent increase in publication volume. Extraction of specific information from these sources requires highly sophisticated text mining- and information extraction-tools. However, the integration of freely available tools into customized workflows is often cumbersome and difficult. We describe SIA, our contribution to the BeCalm-TIPS task, a scalable, extensible, and robust annotation service. The system currently covers three named entity types (i.e., Mutations, Diseases, and miRNA) and is freely available under Apache 2.0 license at https://github.com/Erechtheus/sia.

Key words: Annotation service, Robustness, Scalability, Extensibility

1 Introduction

A vast amount of information on biomedical processes is scattered over millions of scientific publications. Manual curation of this information is expensive and cannot keep up with the ever increasing volume of biomedical literature [7]. To this end, several sophisticated natural language processing tools have been proposed to assist professionals in finding specific information from texts. Many of these highly specialized tools are provided as open source projects to the community. However, the integration of state-of-the-art open source tools into customized text-mining workflows is often difficult and cumbersome [10,12]. Standardized interchange formats, such as BioC [5], enable the exchange of text mining results but the initial set-up of these tools is still an unsolved issue. Exposing tools via public web services implementing common specifications bypasses this problem and allows a code-agnostic integration of specific tools by providing an interoperable interface to third parties. This enables simple integration, comparison, and aggregation of different state-of-the-art tools. In this publication we present SIA, our contribution to the BeCalm TIPS task [9], a robust, scalable, extensible, and generic framework to combine multiple named entity recognition tools into a single system.

The publication is organized as follows: First, we provide a general description of the system architecture, followed by details concerning the implementation and failure handling, and end with a summary and future work section.
2 General Architecture

Design Goals: SIA is designed around the following three main concepts:

1. **Scalability** Ability to handle a large amount of concurrent requests, tolerating bursts of high request rates over short periods of time.
2. **Robustness** Temporary failures (*e.g.*, networking problems or server failure) should be handled gracefully and not lead to dropped requests.
3. **Extensibility** Enable simple integration of arbitrary NLP tools to reduce initial burdens for providing an annotation service.

Figure 1 shows a high level overview of the general architecture. Overall, SIA consists of three logical parts, the front end, back end as well as a result handling component. A message based architecture and route handling based on Enterprise Integration Patterns specifies how requests are handled by different components and flow through the system. While a complete discussion of the integration patterns is out of scope of this publication, interested readers can refer to [6] for a detailed description.

![Diagram](image_url)

**Fig. 1:** General Architecture of SIA. The front end handles new requests and forwards them to the back end over a message bus. Each message is transformed through a series of components, which in turn are connected via named queues. The result handler collects the annotation responses and returns them to the calling client.

The **front end** is the user facing component, handling incoming annotation requests. Received requests are forwarded to the **back end**, which downloads individual documents and feeds them through a chain of annotators. The results are made available to the **result handler**, which sends the annotations back to the requester.
All components are connected to a message bus and exchange messages. This loosely coupled design allows to easily scale, replace and augment each participant in the message flow independently. Persistent named queues are defined as input and output for all components. These queues are stored for the entirety of the systems lifetime. This architecture provides fault tolerant and scalable processing. Fault tolerance is enabled through component wise acknowledgment of each successful message processing, which allows replaying all unacknowledged messages during system recovery.

Messages carry information through the system and consist of a Header and Payload part. The Header contains meta information, such as expiry date, global ids or requested annotation types and is used by the system to route messages to the respective consumers. The Payload contains the actual data to be processed. Each requests is translated into a new message that flows through the system, is enhanced, transformed and aggregated by parts of the message flow to derive a annotation result.

The following sections describe each individual system component in details.

2.1 Front end
The front end encapsulates the annotation processing from the clients and serves as the entry point to the system. Currently it provides a REST endpoint according to the Becalm-TIPS task specification\(^1\). Incoming requests are translated into messages and forwarded to the input queue. This way, the overall processing in the front end is very lightweight and new requests can be handled irrespectively of any ongoing annotation processing.

To handle multiple concurrent requests with varying deadlines, we make use of the fact that the input queue is a priority queue, and prioritize messages with an earlier expiry date\(^2\). The message expiry date, as provided by the calling clients, is translated into a message priority. Using the currently processed messages and their deadlines as well as past elapsed processing time statistics allows us to estimate the individual message urgency.

The front end also handles validation and authorization, which moves this logic into a central place. To monitor the system, statistics are served about average requests rate, document types as well as back end processing counters.

2.2 Back end
The back end is concerned with fetching documents from the supported resources, calling the requested annotators for each resulting text fragment, aggregating the results and feeding them to a result handler.

The back end process is modeled using a sequence of message transformations, which subsequently read from message queues and post back to new ones. The message flow starts by reading new requests from an input queue. As a

\(^1\) Other entry points can easily be added

\(^2\) Already running requests will not be canceled, the priority is just used as a fast path
A single annotation request consists of multiple document ids, incoming messages are first split. Splitting takes one message as input and generates as many individual messages as there are document ids specified. Each document id is then retrieved by passing it through a corpus adapter, which fetches the raw text from the respective endpoint. The outcome is the retrieved text separated into abstract and title.

Texts are delivered to registered annotators using a recipient list. As annotators have a dedicated input queue in the system, each message header is inspected for any requested annotation types, forwarding the message to all matching queues. This design allows to easily add new annotation components by registering a new input queue and adding it to the routing endpoint candidates. All annotators forward their results to the same result queue, where they are collected and aggregated. Aggregation is the reverse of splitting and combines all annotation results into a single message. The aggregation key is a unique request id, set by the front end and stored in the message header. Finally, the aggregated message is forwarded to an output queue.

While the processing flow is specified in a sequential manner, this does not entail single threaded execution. Each individual transformer, such as a download adapter or an annotator, works independently and can be further scaled out. For example, having more than one annotator of the same type (potentially spread across multiple machines) can be used to increase document throughput. Furthermore, multiple requests can be handled in parallel at different stages of the pipeline. Fault tolerance is achieved by transacting the message delivery to each transformer and retrying on failure. Overall, the back end specifies an ordered execution flow and provides two injection points where users can add new functionality with additional corpus adapters or new annotation handlers.

It is noteworthy to point out, that annotation handlers can be hosted inside of SIA, or externally, which enables integrating tools independent of programming language or operating system.

2.3 Result handler

Aggregated annotation results from the back end are committed to the message bus and picked up for further processing. In the current design only one result handler is specified. We implemented a REST handler according to the TIPS task definition, which posts annotation results back to the requester. Additional consumers, such as statistics gatherer or result archival processes can easily be added.

3 Implementation

SIA is implemented in Java and uses RabbitMQ\(^3\) as message bus implementation. To exemplify the extensibility of our approach, we integrated NER components for three different entity types: Mutation names are extracted using

\(^3\) https://www.rabbitmq.com/
Listing 1.1: Extension interface definition for annotators

```java
public interface Annotator {
    Set<PredictionResult> annotate (ParsedInputText payload);
}
```

Listing 1.2: Extension interface definition for corpus adapter

```java
public interface DocumentFetcher {
    List<ParsedInputText> load (IdList idList);
}
```

SETH \[11\]. For micro-RNA mentions we implement a set of regular expressions\(^4\), which follow the recommendations for micro-RNA nomenclature \[3\]. Disease names are recognized using a dictionary lookup \[1\], generated from UMLS disease terms \[4\].

The messaging abstraction provides a clean separation of message routing and the actual message processing. Listing 1.1 shows the general interface contract SIA is expecting for each annotator, which does not expose the messaging infrastructure. Thus integrating the aforementioned annotators is as simple as implementing three wrappers and registering them as routing endpoints.

To increase the throughput of the back end, multiple instances of SIA can be started on different machines, where each instance would process requests in a round robin fashion. If this scaling is coupled with an input queue monitoring, the back end can be automatically scaled up or down to respond to changes in load pattern. Instead of scaling the complete back end, individual components can also be duplicated in a similar fashion, if they present a processing bottleneck.

### 3.1 Corpus Adapters

SIA contains corpus adapters for PubMed, PMC, Patent server and BeCalm abstract server. These components are represented as transformers, which process document ids and return retrieved source texts. They are implemented following the interface definition shown in Listing 1.2, which similarly does not expose the messaging infrastructure. If an adapter supports bulk fetching of multiple documents, we feed a configurable number of ids in one invocation.

Since connecting to these endpoints is effectively calling a potentially unreliable remote service over an unreliable channel, retry on failure is used. This is backed up by the observation, that the most commonly observed error was a temporarily unavailable endpoint. To spread retries we use exponential backoff on continuous failures with an exponentially increasing time interval, capped at

\(^4\) https://github.com/Erechtheus/mirNer
a maximum (initial wait 1s, multiplier 2, max wait 60s). If an endpoint fails to respond after a configurable number of retries, we mark that document as unavailable and continue the processing. This allows a trade-off between never processing any results and giving up too early.

4 Failure Handling

In the following we describe the strategies implemented in SIA for dealing with errors.

**Invalid requests** Invalid requests represent client calls with wrong or missing information. These are handled in the front end using request validation and are communicated back to the caller with detailed error descriptions.

**Backpressure** To avoid that a large number of simultaneous requests can temporarily overload the processing system, SIA buffers all accepted requests in the input queue - using priorities to represent deadlines. Processing components can be scaled up or down by attaching more back end instances.

**Front end fails** If the front end stops, new requests are simply not accepted, irrespective of any ongoing processing in the back end.

**Back end unavailable** Messages are still accepted and buffered when there is enough storage space, otherwise the front end denies any new annotation requests.

**Back end fails** If the back end stops while there are still messages being processed, these are not lost but retried upon restart. This is enabled by acknowledging each message only upon successful processing per component.

**Corpus adapter fails** Each adapter retries, using exponential backoff, to fetch a document before it is marked as unavailable. As the BeCalm-TIPS task does not specify how to signal unavailable documents, these are just internally logged. Any subsequent processing treats a missing document as one with no content.

**Annotator fails** If an annotator fails on a particular message, this can potentially harm the entire back end when annotators are embedded in the system. As annotators are software components not under the control of the processing pipeline, we catch all recoverable errors and return zero found annotations in these cases - logging the errors for later analysis.

**Result Handling fails** The TIPS task description expects the result of an annotation request to be delivered to a known endpoint. If this fails, it is retired in a similar manner to the corpus adapter failure handling.

**Message expired** Clients can define a time until when a processing has to be finished. This is mapped to a time-to-live attribute of each message. This results in automatically dropping any expired message from the message bus.

5 Runtime

SIA is very lightweight and runs anywhere there is a Java environment and a connection to RabbitMQ available. Annotators can be directly embedded or
configured to run externally, exchanging messages through the bus. We deployed SIA into Cloud Foundry, a platform as a service provider, which enables deployments of cloud components [8]. The front end and back end are deployed as two separate application containers. To ease development and running the service, we used a continuous integration workflow. Any code changes automatically trigger a redeployment of the service upon successful test runs.

Figure 2 shows that our system is capable of sustaining a high number of daily requests. Furthermore we observed that the request handling is dominated by corpus downloading times, which make up about 50% of the overall request time. This validates our decision to support bulk downloading of requests, as this amortizes the networking overhead over a number of documents. PubMed articles tend to be longer and thus incur higher annotation times. We also estimated the message bus overhead to about 10%, stemming from individual message serialization and persistence.

6 Summary and Future Work

We described SIA our contribution to the BeCalm-TIPS task which provides scalability - through component replication, fault tolerance - through message acknowledgement, and extensibility - through well defined injection points – with a particular emphasis on failure handling. The message bus provides a good design blueprint, which can be augmented with additional components. One interesting further development path is to port SIA to a distributed streaming environment such as Flink [2] or Spark [13]. These systems reduce the overhead of the message bus at the expense of more complex stream handling and aggrega-
tion. While many of the existing components could be reused, most engineering would need to be spent on implementing a fault tolerant window aggregation.

To encourage further discussion, the source of our current solution is freely available under Apache 2.0 license at https://github.com/Erechtheus/sia along with detailed descriptions on how to deploy the system locally.

Acknowledgments. This research was partially supported by the German Federal Ministry of Economics and Energy (BMWi) through the projects MACSS (01MD16011F), SD4M (01MD15007B) and by the German Federal Ministry of Education and Research (BMBF) through the project BBDC (01IS14013E).

References

High-throughput, interoperability and benchmarking of text-mining with BeCalm biomedical metaserver

Miguel Madrid\textsuperscript{1} and Alfonso Valencia\textsuperscript{2}

\textsuperscript{1}Structural Biology and Biocomputing Programme, Spanish National Cancer Research Centre, 28029 Madrid, Spain
\textsuperscript{2}Life Sciences, Barcelona Supercomputing Centre, Barcelona, Spain.

\texttt{mmadrid@cnio.es}
\texttt{alfonso.valencia@bsc.es}

Abstract. Biomedical annotators are very specific tools applied to a highly complex field. Therefore, this kind of software suffers from an extreme complexity which impedes its usage. This complexity, which is reflected in usability problems, is the main cause of disuse, rejection and low impact. This document discusses several of these problems, as well as possible solutions. As a use case, the NLProt protein-names annotator and its benchmarking with the biomedical annotation metaserver BeCalm is analysed in detail.

Key words: text-mining, biomedicine, BioCreative, metaserver, BeCalm, high-throughput, interoperability, benchmarking.

1 Introduction

In the data-mining age, text-mining is a field of highest interest because the largest part of digital information is stored as unstructured text. The complexity is proportional to the amount information that contains: several idioms/dialect/jargon, language focused on different audiences (formal, informal, technical, simplified), expressiveness (feelings, assessments, context changes, annotations), figures, images, etc. A great effort and extensive knowledge are required to produce tools to analyse and extract relevant information and transform it into more compact, efficient and reusable formats.

In addition, the complexity of text and the importance of the topics addressed make it difficult to interpret them. For instance, in the Life Sciences (Biology, Molecular Biology and Biomedicine) the creation and usage of this kind of tools and systems increases constantly. However, there are no standards or shared evaluation criteria to establish quality measures of high-throughput, interoperability, scalability and reproducibility [1, Krallinger, M., & Valencia, A. (2005)]. This is why initiatives such as BioCreative, evaluation platforms such as BeCalm, and biomedical annotators such as NLProt are critical and of utmost importance.
1.1 BioCreative

*BioCreative* (Critical Assessment of Information Extraction systems in Biology) is an initiative in the text-mining field that was conceived to provide collaborative solutions to the problems arising in the construction and use of information extraction systems in the Life Science domain [2, Hirschman, L., Yeh, A., Blaschke, C., & Valencia, A. (2005)] [3, van der Vet, P. E., van Ommen, G. J., Nijholt, A., & Valencia, A. (2001)]. Rather than a challenge, BioCreative is an effort to improve comparison methods, which define new resources for developing solid and effective gold standards by database curators and domain experts.

1.2 BeCalm

*BeCalm* (Biomedical Annotation Metaserver) was based on the *BioCreative* metaserver, a system for the remote administration of annotation servers [4, Leitner, F., Krallinger, M., Rodriguez-Penagos, C., Hakenberg, J., Plake, C., Kuo, C. J., ... & Johnson, C. A. (2008)] [5, Krallinger, M., Erhardt, R. A. A., & Valencia, A. (2005)]. *BeCalm covers continuous assessment, interoperability between services and standard evaluation measures. Furthermore, it offers the possibility of including annotation web services as well as other resources to the final users.*

1.2.1 TIPS

*TIPS* (Technical interoperability and performance of annotation servers) is a specific task of the *BioCreative* V.5 competition, which is evaluated by the *BeCalm* metaserver [6, P´erez-P´erez, M., P´erez-Rodr´ıguez, G., Blanco-M´ıguez, A., Fern´andez-Riverola, F., Valencia, A., Krallinger, M., & Lourenco, A.]. This task is focused on the most technical part of text-mining (seconds per document, mean annotations per document, mean time in seconds to seek annotations and mean time in seconds per document volume) and it consists of providing a REST API for the annotator, which receives and answers requests from *BeCalm* in a number of given formats (*JSON*, *XML*, *TSV* or *BioC*).

1.3 NLProt

*NLProt* [7, Mika, S., & Rost, B. (2004)] is a tool for extracting and tagging protein-names in text by the combination of a dictionary added to a *Support Vector Machine* (SVM) [8, Tong, S., & Koller, D. (2001)] and linking them to their *UniProtKB*/*TrEMBL* identifications. Furthermore, *NLProt* is also able to tag tissues and species.
2 System description and methods

The requirements for the task consist of an annotation server accessed by a REST API. The aim is to measure three technical key aspects of annotation systems:

*High-throughput*: capacity to deal with large document volumes.
*Interoperability*: flexibility to handle different types of input and output.
*Benchmarking*: performance (mainly in speed) in different tasks.

2.1 BeCalm API

The BeCalm metaserver provides a very simple REST API for linking the annotator, comprising the following methods:

- `updateServerState`: The metaserver keeps continuous contact with the annotation server to analyse the behaviour and evolution of workload assigned to different subtasks.
- `getAnnotations`: The metaserver sends requests with a list of documents to annotate, including relevant meta information, such as document source and maximum execution time. Additionally, the method allows the incorporation of custom parameters.
- `saveAnnotations`: The annotation server returns annotations, complying strictly with the information provided by the `getAnnotations` request.

![Fig. 1: BeCalm API diagram](image-url)
2.2 NLProt

NLProt is an exceptional protein-name annotator. When considering partially tagged names as errors, NLProt still reached a precision of 75% at a recall of 76%. Nevertheless, despite its performance in tagging abstracts (see maximum times table [Subsection 3.1.2]), NLProt is not a command line tool designed for high-throughput batch processing and interoperability. In fact, NLProt does not process volumes of documents (all abstracts and titles have to be included in the same file) and both, the input and the output, have to be adapted to be used by other tools. Because of this, we developed the following wrapper solution:

2.2.1 Wrapper solution

Due to the above difficulties. A wrapper was developed with the programming language Crystal\(^1\) with the help of the web framework Kemal\(^2\), in order to facilitate the creation of the REST API. This wrapper solved all demands of the task [Fig. 2].

- **High-throughput:** The tool was prepared for massive batch processing. Moreover, an even-loop with fibres (as system threads but lighter and cooperatives) was implemented over the libevent library\(^3\).
- **Interoperability:** Both, input and output of titles and abstracts, were adapted to the JSON BeCalm format.
- **Benchmarking:** With the above demands covered, the benchmarking of the annotator system is easily handled by the BeCalm metaserver.

\(^1\) https://crystal-lang.org/
\(^2\) http://kemalcr.com/
\(^3\) http://libevent.org/
Fig. 2: Workflow of the web service annotator
3 Discussion

A virtual machine with 2 CPUs, 1GB of RAM and 5GB of hard drive was used for the task. Maximum times are due to the long stack of requests processed one by one considering hardware constraints.

3.1 Results

Results with all measures collected by the BeCalm metaserver.

3.1.1 Benchmarking by server

The BeCalm metaserver requires documents from three different servers preferably by POST requests (only way to download a large number of documents per volume). BeCalm monitors continuously maximum, minimum and average annotation times.

- Patent: server provided by BeCalm [Fig. 3].
- Abstract: server provided by BeCalm [Fig. 4].
- Pubmed: NCBI server [9, Canese, K., & Weis, S. (2013)] accessed through E-utilites [10, Sayers, E. (2009)] [Fig 5].
Fig. 3: *Becalm* Patent Server benchmarking

Fig. 4: *Becalm* Abstract Server benchmarking

Fig. 5: *Becalm* PubMed Server benchmarking
3.1.2 Benchmarking by number of documents

*BeCalm* is focused on bursts of one document volumes. *NLProt* annotation time follows the trend of a linear function $f(x) = ax + b$ with $R^2 = 0.997$ [11, Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009)] with the number of documents per volume.

<table>
<thead>
<tr>
<th>Documents (max time)</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>getDocuments</td>
<td>0s</td>
<td>1s</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>NLProt</td>
<td>10s</td>
<td>22s</td>
<td>4m41s</td>
</tr>
<tr>
<td></td>
<td>saveAnnotations</td>
<td>0s</td>
<td>1s</td>
<td>3s</td>
</tr>
</tbody>
</table>

Table 1: Number of document per maximum time

3.1.3 General benchmarking

*BeCalm* also monitorizes a summary with average times from all document servers [Fig 6].

3.2 BeCalm API, web and feedback

The *BeCalm* API is simple but very powerful. It can be adapted easily with the custom parameters section of the requests. The web interface was designed to be minimalistic and user friendly, providing help and custom queries to test the annotation server and provide a summary of the query. In the future, it could be considered to add daily annotation statistics from other competitors.
4 Conclusion

The *BioCreative* task proves the critical importance of the evaluation of biomedical annotators with metaservers such as *BeCalm*, and demonstrates the need to adapt text-mining annotators to this type of standards. The evaluation should provide a qualitative assessment of the annotation servers in terms of functionality, as well as quantitative rating of technical key parameters. Evaluations have to be carried out against gold standards to obtain reliable and comparable measures for parameters such as precision, recall or F-score.

To highlight, wrapper strategy over the annotator also has been tested successfully with the *Conditional Random Fields (CRF)* based [12, Okazaki, N. (2007)] *miRNA* entity tagger [13, Sammartino, J. C., Krallinger, M., & Valencia, A. (2016)] using the NERsuite toolkit [14, Cho, H. C., Okazaki, N., Miwa, M., & Tsujii, J. (2010)] for improving its behavior and to meet the demands (high-throughput, interoperability and benchmarking) of the task.
References

Performance and interoperability assessment of Disease Extract Annotation Server (DEAS)

Jitendra Jonnagaddala*1,2, Hong-Jie Dai3,4, Chen-Kai Wang5, Po-Ting Lai6

1 School of Public Health and Community Medicine, UNSW Sydney, Australia
2 Prince of Wales Clinical School, UNSW Sydney, Australia
3 Department of Computer Science and Information Engineering, National Taitung University, Taitung, Taiwan, R.O.C,
4 Interdisciplinary Program of Green and Information Technology, National Taitung University, Taitung, Taiwan, R.O.C,
5 Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan, R.O.C,
6 Department of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan, R.O.C.

*1,2z3339253@unsw.edu.au; 3,4hjdai@nttu.edu.tw; 5dennisckwang@gmail.com; 6potinglai@gamil.com

Abstract. Over the past decade, many biomedical information extraction tools have been developed. Some of these tools are provided with access via web services. However, many of these tools are not interoperable with web services performance not evaluated. In this study, we implemented and evaluated the performance of an interoperable web service that annotates disease entities called Disease Extract Annotation Server (DEAS). The DEAS evaluation was carried out over a period of two months. Interoperability, stability, speed and batch processing capabilities of DEAS were evaluated. DEAS was able to process documents from multi data sources and types with response times varying from one second to six seconds per document. The performance evaluation assisted in improving the underlying CRF-based disease entity recognition pipeline. In future, we would like to support more data format standards and also improve DEAS performance by employing distributed and parallel processing techniques.

Keywords. Named entity recognition; biomedical entity recognition; biomedical meta-services; disease annotation server

*Corresponding author
1 Introduction

The exponential increase of published biomedical literature is creating a great demand to effectively retrieve and extract relevant information. Many methods have been proposed to extract unstructured information effectively. For example, Dai. et al presented methods to normalize species and gene/protein mentions [1]. However, often these methods focus on a very specific extraction or retrieval tasks. Also, sometimes these methods are not interoperable. The data formats are different and are usually developed and supported for different platforms. It is also difficult to unify extracted information from multiple information extraction systems. To some extent this can be resolved by implementing meta-services [2]. Meta-services integrate multiple information extraction (IE) and information retrieval (IR) systems by enforcing standards using web services where suitable. Web services are software packages designed to support communications by wide range of devices and platforms using web standards. BioC, a data interchange format is a good example which can be adapted into meta-services [3].

The Biomedical Annotation meta-server2 (BeCalm) platform supports meta-services by providing access to various annotation servers. BeCalm reinforces a minimal set of standards to harmonize various annotation servers; which in turn can be subjected to comparative assessment and continuous evaluation. In this study, we implemented an interoperable REST (Representational State Transfer) based web services that annotate disease entities based on the BeCalm meta-server standards. We call our implementation as Disease Extract Annotation Server (DEAS). We also evaluated the performance of the DEAS by carrying out an evaluation which lasted for two months as part of the BioCreative V.5 Technical interoperability and performance of annotation servers (TIPS) task3.

2 Methods

The DEAS4 is an extension to our previous work [4, 5]. In specific for the purpose of this study, we extended our previous web services to support BeCalm meta-server API specifications. The DEAS mainly include

2 http://www.becalm.eu/api/
3 http://www.becalm.eu/pages/biocreative
4 https://github.com/TCRNBioinformatics/DiseaseExtract
three components: 1) data retrieval 2) web services and 3) infrastructure layer. DEAS supports automatic retrieval of biomedical articles, patents and abstracts from three different data sources (table 1) in the data retrieval layer. In this component, the DEAS retrieves the documents requested by a meta-server or another other client and passes it over to the web services component for further handling.

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Data type</th>
<th>Data source details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Server</td>
<td>Biomedical Abstracts</td>
<td><a href="http://193.147.85.10:8088/abstractserver/json/">http://193.147.85.10:8088/abstractserver/json/</a></td>
</tr>
</tbody>
</table>

Table 1: DEAS Data retrieval sources

DEAS web services were developed using the REST paradigm [6]. REST is an architectural style that allows to create communication services using The Hypertext Transfer Protocol (HTTP) standard. REST is most commonly known for its scalability and light-weight characteristics. We have used Swagger⁵ framework to develop, document, and consume the DEAS web services. The DEAS currently supports both requests and responses in JSON format. However, interoperable BioC format [3] is not supported, although it is supported in the underlying disease entity recognition system. A sample DEAS web services request and response is presented in table 2. In this sample, DEAS automatically retrieves a patent from the patent server and annotates the content of the patent for disease entities. Please refer to the DEAS documentation for complete list of web services supported.

```
BeCalm Request:
getAnnotations:
{
  "name": "BeCalm",
  "method": "getAnnotations",
  "becalm_key": "0000000000",
  "custom_parameters": {},
  "parameters": {
    "documents": [{"document_id": "CA2073855C", "source": "PATENT SERVER"}]
  }
```

⁵ http://swagger.io/
"types":["Disease"],
"communication_id":1000
}
}

DEAS Response:
{"status":200,"success":true,"key":"0000000000"}

Table 2: DEAS API Example

The data retrieval and web services components were wrapped around an infrastructure layer where we used a Linux machine with 1GB memory and 1 Intel Xeon 2.4 GHz CPU. The stability, speed and batch processing capabilities of DEAS were evaluated using the metrics, constructed mainly of the number of predictions made and response times. The response times here refer to the time taken by an annotation server such as DEAS to respond to the request made by a client, such as BeCalm. If the request includes processing multiple documents, the response time is expected to be higher. Similarly, the response time could vary depending on the infrastructure and other underlying components of an annotation server.

3 Results and Discussion

The DEAS evaluation began on 1st of February and ended on 31st of March 2017. Table 3 presents the performance of our disease annotation server developed by each data source. DEAS was able to process majority of the requests from BeCalm server where either patent server or, PMC and PubMed server is the data source. However, the majority of the requests from the abstract server were not processed which is reflected in terms of the number of exceptions and predictions. It is also important to note that the large maximum response times are mainly due to the exceptions caused.

<table>
<thead>
<tr>
<th></th>
<th>Patent server</th>
<th>Abstract server</th>
<th>PMC and PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total processing time</td>
<td>26 days 21h:10m 8: s</td>
<td>3 days 14h:0m 1: s</td>
<td>3 days 02h:5m 0: s</td>
</tr>
<tr>
<td>Total predictions</td>
<td>189 k</td>
<td>0 k</td>
<td>374 k</td>
</tr>
<tr>
<td>Mean predictions per request</td>
<td>3.3</td>
<td>0</td>
<td>4.9</td>
</tr>
<tr>
<td>Exceptions/Total requests</td>
<td>7177/82213</td>
<td>23798/135938</td>
<td>7920/92811</td>
</tr>
<tr>
<td>Minimum response time (seconds)</td>
<td>1.72</td>
<td>1.65</td>
<td>1.84</td>
</tr>
</tbody>
</table>
Maximum response time (seconds) | 30249.38 | 895.45 | 24568.9
Mean response time (seconds)  | 3116.81  | 2.78  | 967.52

Table 3: Performance by data source

Figure 2 presents the DEAS daily response time from 15th of March to 31st of March. The response times were collected at irregular intervals during the day and the figure illustrates that our response rates varied somewhere between 2 seconds and 12 seconds.

The batch processing capabilities were evaluated by posting requests with multiple documents in each request varying from 10 to 2400 documents per request. Table 4 presents the metrics for batch processing. The DEAS capability to process in bulk has been significantly improved by reducing the processing time from 5.9 seconds per document (40 documents) to 1.19 seconds per document (2400 documents).

<table>
<thead>
<tr>
<th>Total Time(mm:ss.ms)</th>
<th>Time(s)/documents</th>
<th>#Documents</th>
<th>#Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>47:38.9</td>
<td>1.19122</td>
<td>2400</td>
<td>5667</td>
</tr>
<tr>
<td>15:03.0</td>
<td>0.75254</td>
<td>1200</td>
<td>2546</td>
</tr>
<tr>
<td>06:21.0</td>
<td>0.76208</td>
<td>500</td>
<td>1031</td>
</tr>
<tr>
<td>03:30.1</td>
<td>2.2116</td>
<td>95</td>
<td>262</td>
</tr>
</tbody>
</table>

* minutes: seconds. milliseconds
Table 4: Batch processing performance metrics

<table>
<thead>
<tr>
<th></th>
<th>03:54.4</th>
<th>5.86047</th>
<th>40</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:46.9</td>
<td>4.69455</td>
<td>10</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

The initial results from this study assisted us in improving our underlying core entity recognition pipeline which was based on conditional random fields (CRFs) [7]. Our core pipeline [4] used Stanford PTBTokenizer\(^7\) and was failing to process documents containing non-UTF characters. In other words, DEAS could not find any disease annotations and was sending empty responses to the BeCalm meta-server. This assisted us in identifying and fixing the underlying issue with the tokenizer. Similarly, we were able to detect and fix memory related performance issues resulting to improved scalability and reliability of the underlying pipeline. On the other hand, we believe that the BeCalm meta-server can also be improved in the API, documentation and communication areas. For example, at the beginning of the evaluation period BeCalm did not require an annotation server to support the Abstract server. However, the addition of the Abstract server data source requirement was not communicated. Thus, our annotation server could not annotate majority of the requests with the Abstract server as the data source (Table 3). One major limitation of this study is the performance metrics used. The performance is evaluated for speed, stability and batch processing capabilities only, but not for the quality or accuracy of underlying disease entity recognition which typically is reported using precision, recall and F-measure.

4 Conclusion

In conclusion, we present the performance and interoperability assessment of DEAS, a disease extract annotation server. DEAS employs CRF-based entity recognition to extract disease entities from biomedical articles and patents. The evaluation results show that DEAS can effectively interoperate with the BeCalm meta-server and process documents from multiple sources. In our future work, we also would like to support the processing of documents with non-UTF characters. Additionally, aside from supporting requests and responses in JSON format, we would also like to enable support for BioC format. Lastly, we would also like to

\(^7\) https://nlp.stanford.edu/software/tokenizer.shtml
decrease response times by employing distributed and parallel processing techniques.

5 Acknowledgment

This study was conducted as part of the electronic Practice Based Research Network (ePBRN) and Translational Cancer research network (TCRN) research programs. We would like to thank the organizers of BioCreative V.5 TIPS Track for providing us, with the sample code to develop BeCalm compliant APIs.

REFERENCES

TextImager as an interface to BeCalm

Wahed Hemati, Tolga Uslu, and Alexander Mehler

Text Technology Lab,
Goethe Universität Frankfurt am Main, Germany
{hemati,uslu,amehler}@em.uni-frankfurt.de
http://www.hucompute.org/

Abstract. TIPS (Technical interoperability and performance of annotation servers) is a novel BioCreative task. It focuses on the technical aspects of making NER taggers available as webservies. In this paper, we present the functionality and architecture of BeCalm metaserver integrated into TextImager. We present the integration of in-house developed NER tagger and also the integration and adaption of available NER tagger.

Key words: TextImager, BeCalm, TIPS, NER

1 Introduction

The novel BioCreative task TIPS focuses on making NER taggers available as web servers. We introduced TextImager in [1] where we focused on the arihctecture and the functions of its backend. In this paper, we present TextImager as an annotation server on the BeCalm platform. TextImager is a UIMA[2]-Based framework that offers a wide range of NLP. It is modular and expendable, due to the underlying UIMA architecture. We made TextImager available online. It can be accessed by BeCalm metaserver requests and is able to answer such requests in the BeCalm TSV format. Currently we trained and implemented 6 NER systems for the BeCalm GPRO and CEMP tasks, namely StanfordNER, MarMot, CRF++, MITIE, Glample and CRFVoter. Each of these tagger can be used in a pipeline together with other NLP-tools that are already integrated in TextImager.

2 Systems description and methods

In this section we will describe the technical implementation of BeCalm metaserver integration into TextImager. We implemented a REST API that provides access and responds to the BeCalm metaserver requests described in http://www.becalm.eu/api

In order to obtain annotation documents the BeCalm metaserver sends a getAnnotations() request for a set of documents. The getAnnotations() message is in JSON format, which is exemplified in Listing 1.1. The request contains specification about the documents that need to be processed, authentication information and the custom parameter annotator.
After receiving the getAnnotations() request, our server responds to BeCalm with an acknowledge message.

After the acknowledgement, our server executes the internal workflow, which starts by processing the BeCalm JSON request. The retrieved documents are downloaded by the TextImager Rest API from their specific sources, namely Abstract Server, Patent Server and PubMed Server. The documents are then passed to the TextImager backend. TextImagers backend is a UIMA-based framework that offers a range of NLP tools. Every TextImager component is configured as a UIMA-AS service, which may run standalone or in a pipeline (see Figure 2, Stanford NER service Interface, MarMot Service Interface, etc). All service instances are located on servers. Note that these instances can be distributed among different servers (see Figure 2, Server 1, Server 2, Server X). We trained and integrated 6 NER systems for the BeCalm GPRO and CEMP tasks into TextImager, namely StanfordNER[3], MarMot[4], CRF++[5], MITIE[6], Glample[7] and CRFVoter.

The TextImager Rest API sends the request and the documents to the TextImager Orchestrator. The custom parameter: annotator (see Listing 1.1) is used by the orchestrator to determine which NER service to run. The TextImager Orchestrator selects and acquires components and their resources: it arranges components into pipelines and grants the ability to parallelize them. Each of the 6 integrated NER systems require tokenization, lemmatization and morph tagging as preprocessing steps. Finally after the orchestration step, the documents

### Listing 1.1. Example BeCalm metaserver JSON request

```json
{
  "method": "getAnnotations",
  "becalm_key": "141xxx",
  "name": "becalm",
  "custom_parameters": {
    "annotator": "BioGproS"
  },
  "parameters": {
    "expired": "2017-05-02T15:10:00+02:00",
    "documents": [
      {
        "source": "ABSTRACT SERVER",
        "document_id": "12140745"
      },
      {
        "source": "PATENT SERVER",
        "document_id": "EP0959896 B1"
      },
      ...
    ],
    "communication_id": "4672794"
  }
}
```
are processed by means of the selected service pipeline. After processing is done, the output is passed to the TextImager Rest API, which saves the output to the BeCalm server, by calling the method `saveAnnotations`.

![Architecture of BeCalm metaserver integrated into TextImager](image)

**Fig. 1.** Architecture of BeCalm metaserver integrated into TextImager

### 3 Discussion

In this work, we presented TextImager as a BeCalm annotation server. Our annotation server is a distributed and modular framework. We currently integrated 6 NER systems for CEMP and GPRO. In future work, we will focus on integrating more NER systems into TextImager and make it available as annotation server. We will also provide more output formats.

---

1. [http://www.becalm.eu/api](http://www.becalm.eu/api)
Acknowledgments. We gratefully acknowledge support by the Deutsche Forschungsgemeinschaft via the Specialised Information Services Biodiversity Research.

References

Olelo’s named-entity recognition web service in the BeCalm TIPS task

Hendrik Folkerts and Mariana Neves

Hasso Plattner Institute, August-Bebel-Str. 88 Potsdam, 14482 Germany
{Hendrik.Folkerts@student.hpi.uni-potsdam.de
Mariana.Neves@hpi.de}
https://hpi.de/plattner

Abstract. Named-entity recognition (NER) is an important preliminary tasks in many text mining systems. However, few web services are currently freely available for use. The BeCalm TIPS challenge aims to evaluate web services for biomedical NER in terms of reliability and performance. We participated with our dictionary-based NER which is part of our Olelo question answering system. Since the start of the evaluation on March 2017, our web services has received more than 290,000 requests and returned more than 30 millions annotations for ten annotation types and for three types of documents (patents, abstracts and PubMed).

Key words: Named-entity recognition, dictionary-based matching, MeSH, UMLS, in-memory database, web services.

1 Introduction

Named-entity recognition (NER) is an important preliminary task in many text mining workflows [1]. Although there are plenty of tools available for extracting a variety of entity types, such as for species [4] or disease names [3], text miners still need to install and integrate various tools for the various entity types. Web services offers the ability to obtain annotations for documents without the need to deal with installation and functionality of individual tools.

The BeCalm TIPS (technical interoperability and performance of annotation servers) challenge \(^1\) [5] aims to evaluate the performance of web services for biomedical NER. Previously, many challenges have evaluated the performance of NER taggers for various entity types in terms of F-measure, precision and recall, such as [7]. These metrics focus mainly on the quality of the annotations returned by the systems, i.e., the rate of the correct annotations returned by the tools as well as whether the tools missed correct annotations.

As opposed to these previous challenges, BeCalm focus on the technical aspects of NER taggers and uses different evaluation metrics, such as reliability and performance indicators. Web services need to comply with some requirements, such as to provide a REST interface and output in standard format (e.g., BioC

\(^1\) http://www.becalm.eu/pages/biocreative
Finally, services are also required not to cache neither the documents nor the previously predicted annotations. The BeCalm TIPS challenge took place from beginning of February 2017 and ran until April 2017. The challenge is composed of four phases, as described below:

- Phase 1 - one document per request: five requests of single documents (patents or abstracts). The documents need to be retrieved from the corresponding server and annotations should be returned.
- Phase 2 - stress test: simultaneous requests to check the whether systems are able to manage these without collapsing.
- Phase 3 - bulk processing: request of multiple and large documents to check robustness and scalability of the systems.
- Phase 4 - PubMed/PMC: requests for documents from patent servers, PubMed and PubMed Central (PMC) to check the flexibility of the system to retrieve and process documents from various sources.

In this work, we present our Olelo web service for biomedical NER. Olelo\(^2\) is a question answering (QA) system for biomedicine \([2]\) which includes many natural language processing (NLP) components, such as question understanding, document retrieval, NER, answer extraction and summarization. The system is built on top of an in-memory database (IMDB) and uses a dictionary-based NER approach, as previously described in our last participation in the BioASQ challenge \([6]\).

This paper is structured as follow: next section describes details of our NER system, followed by the results that we obtained in the BeCalm TIPS challenge.

## 2 System description

In this section we describe the details of our web service that participated in the BeCalm TIPS challenge. The system relies on the NER components behind our Olelo QA system \([6]\). Figure 1 illustrates the many components in the service. Details for each of them is provided below.

The components of our workflow which are not included in the IMDB, i.e, JSON reader, document retrieval and BioC writer, are implemented in the Java programming language. We use the Spring framework to combine the different components of our application. The core of our system is a IMDB (SAP HANA) that runs in a machine with 120 cores and 2067Gb of main memory.

*JSON reader*. Requests from BeCalm TIPS are in JSON format and includes the document identifiers and databases.

\(^2\)http://hpi.de/plattner/olelo
Document retrieval. Given the document identifiers and documents databases extracted in the previous step, we proceed to retrieve the documents from the respective document servers. Participating systems need to provide support for four document servers: a patent server, an abstract server (both maintained by BeCalm), as well as PubMed and PubMed Central e-utils web servers. For each of these documents, we extract the title and the text of the abstract.

XS API call. In this step we send the document content to the XS application of our IMDB. The XS application uses an built-in procedure from the IMDB which includes some preliminary NLP procedures such as language detection, sentence splitting, tokenization, stemming and part-of-speech tagging. The response of this API call includes all recognized named-entities. All documents sent to this
API are later deleted to comply with the requirement of the BeCalm challenge of not to keeping local copies of the documents. A screen-shot of some recognized named-entities is shown in Figure 2.

<table>
<thead>
<tr>
<th>UMLS types</th>
<th>BeCalm types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anatomic components</td>
<td>T017 (Anatomical Structure), T029 (Body Location or Region), T022 (Body System), T021 (Fully Formed Anatomical Structure), T018 (Embryonic Structure)</td>
</tr>
<tr>
<td>Cell line and cell type</td>
<td>T025 (Cell)</td>
</tr>
<tr>
<td>Chemical</td>
<td>T103 (Chemical), T120 (Chemical Viewed Functionally), T104 (Chemical Viewed Structurally)</td>
</tr>
<tr>
<td>Disease</td>
<td>T020 (Acquired Abnormality), T190 (Anatomical Abnormality), T049 (Cell or Molecular Dysfunction), T019 (Congenital Abnormality), T047 (Disease or Syndrome), T050 (Experimental Model of Disease), T033 (Finding), T037 (Injury or Poisoning), T048 (Mental or Behavioral Dysfunction), T191 (Neoplastic Process), T046 (Pathologic Function), T184 (Sign or Symptom)</td>
</tr>
<tr>
<td>Gene</td>
<td>T028 (Gene or Genome)</td>
</tr>
<tr>
<td>miRNA</td>
<td>T086 (Nucleotide Sequence)</td>
</tr>
<tr>
<td>Organism</td>
<td>T100 (Age Group), T011 (Amphibian), T008 (Animal), T194 (Archaeon), T007 (Bacterium), T012 (Bird), T204 (Eukaryote), T099 (Family Group), T013 (Fish), T004 (Fungus), T096 (Group), T106 (Human), T015 (Mammal), T001 (Organism), T011 (Patient or Disabled Group), T002 (Plant), T098 (Population Group), T097 (Professional or Occupational Group), T014 (Reptile), T010 (Vertebrate), T005 (Virus)</td>
</tr>
<tr>
<td>Protein</td>
<td>T116 (Amino Acid, Peptide, or Protein)</td>
</tr>
<tr>
<td>Subcellular structure</td>
<td>T026 (Cell Component)</td>
</tr>
<tr>
<td>Tissue and organ</td>
<td>T023 (Body Part, Organ, or Organ Component)</td>
</tr>
</tbody>
</table>

Table 1. Mapping of UMLS semantic types to BeCalm annotation types.

**Custom dictionaries.** We compiled a custom dictionary based on MeSH terms and concepts from the UMLS database. Previously, we identified ten high-level nodes of the MeSH tree and mapped these to UMLS semantic groups or types: A (Anatomy), B (Diseases), C (Species), D (Drugs), E01 (Clinical Diagnostics), G05.360.340.024.340 (Genes), D12.776 (Proteins), Z (Geographicals), E02 (Treatments), C23 (Symptoms). In order to account for the annotation types supported by BeCalm, we manually mapped these to the UMLS semantic types, as shown in Table 1. Currently, we are providing annotations for ten of the twelve annotations types supported by BeCalm TIPS. We do not support entity name normalization.

**BioC writer.** In this step we generate the BioC file with the corresponding annotations to the requested documents. We keep no copy of the extracted an-
notation from previous request, in order to comply with this requirement from the BeCalm TIPS challenge.

3 Evaluation

In this section we describe the metrics utilized by BeCalm TIPS for evaluation of the systems and we present the results that we obtained so far with our web service. These are the official results that are available to each participants in the BeCalm TIPS participant’s area.

3.1 Evaluation metrics

As discussed earlier, BeCalm TIPS utilizes evaluation metrics which focus on the reliability and time performance of the systems instead of the quality of their annotations. More details are available in the BeCalm documentation.

Regarding reliability, BeCalm proposes two metrics:

- Mean time between failures (MTBF): average of the time elapsed between failures of an annotation server, i.e., the time elapsed between two consecutive failures.
- Mean time to repair (MTTR): time required to repair a failure, i.e., the time elapsed between a failure and the running of the system again.

Regarding performance metrics, the TIPS challenge proposes four metrics:

- Mean annotations per document (MAD): total number of annotations divided by the total number of documents.
- Mean time per document volume (MTDV): average time taken to annotate a document based on the sum of document sizes in bytes.
- Mean time seek annotations (MTSA): sum of response time divided by the total number of annotations returned.
- Average response time (ART): sum of response time divided by the total number of responses, i.e., average time per request/response.

3.2 Results

The values for some of the above metrics are available for participants in the participant’s area, as well as additional statistics about the performance of the system. In this section we present the results we have obtained so far. Unfortunately, we only have access to our own results and cannot yet compare these with results from other participating systems.

Since the start of the evaluation period, Olelo has received a total of 291,452 requests (as of April 3rd, 2017). Table 2 summarizes the statistics of our results for the different document types.

\[^3\text{http://www.becalm.eu/files/material/BeCalm_TIPS.pdf}\]
Table 2. For each document server, results are shown for the total number of predictions, the total number of requests, the mean number of predictions per request, the total number of exceptions and the processing time (mean, minimum and maximum). Values are as of April 3rd, 2017.

<table>
<thead>
<tr>
<th></th>
<th>Docs.</th>
<th>Pred.</th>
<th>Req.</th>
<th>Mean pred/req</th>
<th>Excep.</th>
<th>Time (mean/min/max/)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patent</td>
<td>6822k</td>
<td>90,213</td>
<td>77.9</td>
<td>2,339</td>
<td></td>
<td>1.23s/0.35s/102.84s</td>
</tr>
<tr>
<td>Abstract</td>
<td>14,833k</td>
<td>115,938</td>
<td>128.7</td>
<td>394</td>
<td></td>
<td>1.17s/0.36s/202.88s</td>
</tr>
<tr>
<td>PubMed</td>
<td>8,421k</td>
<td>85,301</td>
<td>121.5</td>
<td>15,049</td>
<td></td>
<td>1.99s/0.9s/37.08s</td>
</tr>
</tbody>
</table>

**Fig. 3.** Screen-shot of the historical time response of Olelo as shown in the BeCalm participant’s area.

Regarding the metrics described above (cf. section 3.1), we obtained a MAD (mean annotations per document) of 100.8349, a MTSA (mean time seek annotations) of 0.01959 and a MTDV (mean time per document volume) of 0.0343. Additionally, the mean processing time per document was 1.39571 seconds.

Our historical time response is shown in Figure 3. So far, we obtained a total of 30,615 exceptions (as of April 3rd, 2017), i.e., cases in which our system did not successfully respond to the requests. We manually checked the last 50 exceptions returned by Olelo and the results are the following errors: 25 of type UNREACHEABLE_SERVER_Proxy, 23 of REQUEST_TRIES_OVERLIMIT, one NOT_VALID_BY_SCHEMA error and one REQUEST_CLOSED error. The most common error relates to DNS lookup failure for our server, an issue that we are currently looking into. The second most frequent exception indicates that our
server reached the maximum number of attempts for accepting a request, which is probably related to the previous exception. Finally, the NOT_VALID_BY_SCHEMA error was due to errors in the XML DTD schema and the REQUEST_CLOSED exception seems to be due our system submitting more than one response for the same request.

4 Conclusions

We presented our participation on the BeCalm TIPS challenge for the evaluation of reliability and performance of web services for biomedical NER. We participated with the dictionary-based NER procedure behind our Olelo QA system, which relies on custom dictionaries from UMLS and MeSH. Our web service received so far more than 290,000 requests and returned annotations for almost 90% of these, totaling more than 30 millions annotations.

Acknowledgments. We would like to thank Matthias Herzog for technical support and Milena Kraus for her support of mapping the semantic types.

References

OGER: OntoGene’s Entity Recogniser in the BeCalm TIPS Task

Lenz Furrer and Fabio Rinaldi

University of Zurich, Institute of Computational Linguistics
Andreasstr. 15, CH-8050 Zürich, Switzerland
{furrer,rinaldi}@cl.uzh.ch

Abstract. We present OGER, an annotation service built on top of OntoGene’s biomedical entity recognition system, which participates in the TIPS task (technical interoperability and performance of annotation servers) of the BeCalm (biomedical annotation metaserver) challenge. The annotation server is a web application tailored to the needs of the task, using an existing biomedical entity recognition suite. The core annotation module uses a knowledge-based strategy for term matching and entity linking. The server’s architecture allows parallel processing of annotation requests for an arbitrary number of documents from mixed sources. In the discussion, we show that network latency is responsible for significant overhead in the measurement of processing time. We compare the preliminary key performance indicators with an analysis drawn from the server’s log messages. We conclude that our annotation server is ready for the upcoming phases of the TIPS task.

Key words: knowledge-based named entity recognition, biomedical entity linking, parallel processing

1 Introduction

The technical interoperability and performance of annotation servers (TIPS) task [4] is part of the biomedical annotation metaserver (BeCalm) challenge [3]. Participants are asked to build a web service for annotating biomedical entities in given documents on the fly. The goal of the task is to build a fast and reliable annotation server, which can act as a contractor in an inter-universitary biomedical annotation cloud.

2 System description and methods

In the following, we describe OGER, an annotation server built on top of the OntoGene entity recognition suite.
Fig. 1. System architecture of the annotation server.

2.1 Environment

The OGER annotation server is hosted on our institute’s server infrastructure. It is run on a virtual machine (VM) dedicated to services. This VM has a high priority regarding the distribution of processing time among all VMs running on the same machine. It features 128 G of RAM (allocated exclusively to the service VM) and 16 CPUs (shared with other VMs). The operating system is the Debian-based Proxmox Virtual Environment. Incoming HTTPS requests are redirected to the annotation server by an Nginx reverse proxy server.

2.2 Annotation server architecture

The OGER annotation server is written in Python. Besides the standard library, we used two third-party libraries. The micro web-framework Bottle\(^1\) is used to implement the REST API. XML parsing and serialising is performed through lxml\(^2\).

An overview of the system architecture is given in Figure 1. In order to allow a high processing capacity without blocking the API, the annotation server supports parallel processing. The main process runs the REST API, listening to incoming requests. Quick operations are handled directly, such as responding to the \texttt{getServerState} request or sending an acknowledgment response. A number\(^3\) of annotation subprocesses (workers) are responsible for handling

\(^1\) \url{http://bottlepy.org/}
\(^2\) \url{http://lxml.de/}
\(^3\) The number of annotation workers is fixed when the server is (re-)started. Throughout phase 1 of TIPS, we used 5 workers.
the `getAnnotations` requests. Each worker runs a separate instance of the term annotation pipeline (see Section 2.3). A separate subprocess (the sending worker) receives the extracted annotations from all annotation workers. It concatenates them (if necessary) and issues a `saveAnnotations` request to BeCalm’s Metaserver. Finally, another subprocess (not shown in Fig. 1) collects processing errors and other incidents. These problem reports are sent by email to the person in charge of maintaining the server.

The annotation workers operate on a batch basis. Every batch consists of a list of document IDs and a source specifier, referring to one of PubMed, PubMed Central, BeCalm’s abstract server, or BeCalm’s patent server. The requested documents are obtained from their respective remote source, using either NCBI’s `efetch` or BeCalm’s REST API. Both interfaces allow an unlimited number of documents to be requested at once, which means that the entire batch can be retrieved through a single request. Upon retrieval, the documents are converted to a unified internal representation and passed to the entity recogniser. The extracted annotations of the whole batch are accumulated and serialised into the required output format (BeCalm JSON, BeCalm TSV, or BioC XML).\(^4\) The formatted annotations are then pushed to the sending worker.

The annotation workers can only handle documents from a single source per batch. Therefore, if an annotation request asks for documents from different sources,\(^5\) the main process groups the documents by source and creates multiple batches. For example, for a 10-document request, it would initiate a batch of 5 documents to be fetched from PubMed, and another batch of 5 abstracts to be requested from BeCalm’s patent server. The batches are then processed in parallel by separate workers. This parallelisation results in a speed benefit even for small batches, since the network-related waiting times do not add up (as they would in a fully sequential approach).

The sending worker, finally, ensures that there is exactly one `saveAnnotations` request for each `getAnnotations` request. If an incoming request was split up into multiple batches, it waits for the formatted annotations from each batch to be completed and merges them into a single structure before sending them to the Metaserver. The concatenation of multiple batches needs some care, in that certain structural elements must not be repeated: In TSV format, the headers may only occur once. In JSON, the top-level array must span the entire collection using a single pair of brackets. Similarly, in XML, there can only be one root node, and BioC’s collection-level metadata may not be repeated.

\(^4\) Currently, the output format is specified through the server configuration and cannot be changed without restarting the server. However, the API could easily be changed to accept the output format as a parameter.

\(^5\) This was never the case in phase 1 of TIPS, as all requests asked for a single document only. However, we successfully tested this functionality with private requests through BeCalm’s web interface.
2.3 OntoGene term annotation pipeline

The OntoGene term annotation pipeline is a knowledge-based concept recognition system for biomedical entities. Designed for information extraction systems targeting scientific literature, it has been successfully applied to a range of entity types (genes/proteins, chemicals, diseases, among others [7, 8, 5, 6]). It has been recently reimplemented in Python as an integral processing suite [2], replacing the former amalgamation of modules written in various programming languages, communicating through a multitude of intermediate files written on disk. While the new pipeline provides a command-line interface with a lot of flexibility, we used it as a Python library for the annotation server.

As a knowledge-based system, the core recognition procedure relies on a list of target terms, which are connected to entity identifiers. The coverage of matching term variants is raised through a series of preprocessing steps with a normalising effect, such as an aggressive, lossy tokenisation strategy which collapses spelling alternations like e.g. “SRC1”/“SRC 1”/“SRC-1” into the same representation. A more detailed description of the annotation process can be found in [1].

For the present work, we used the following terminology resources (with their respective entity types):

- Cellosaurus (cell lines)\(^6\)
- Comparative Toxicogenomics Database (CTD) (chemicals, diseases)\(^7\)
- Gene Ontology (cellular components only, labelled “subcellular structure” in TIPS)\(^8\)
- NCBI Taxonomy (organisms)\(^9\)

These resources were aggregated and converted to a unified format using the Bio Term Hub.\(^{10}\) Due to the highly flexible design of the system, the range of supported entity types can be extended very easily. By simply including additional terminology resources, more target entities can be covered.

3 Discussion

At the time of writing, only the key performance indicators are available. Also, global results have not been released, meaning that each participating group only sees results for their own system(s).

3.1 Performance

According to the statistics in the participant area of BeCalm’s web interface, the key performance indicators for our system are as follows:

\(^6\) http://web.expasy.org/cellosaurus/
\(^7\) http://ctdbase.org/
\(^8\) http://geneontology.org/
\(^{10}\) http://pub.cl.uzh.ch/purl/biodb/
The first two indicators (MAD and MPDV) hint at the sensitivity of the annotation server, as they represent the number of annotations per document and per Byte of a document, respectively. Without evaluating the correctness of the annotations, however, it is not clear what conclusions can be drawn from these figures. MAD might help put the other performance indicators in context (MTSA, in particular).

The other three indicators (MTDV, MTSA, ART) are very similar, in that they represent the average time needed to process one Byte of a document, one annotation, and one document, respectively. In our analysis, we will focus on ART, the time needed to process a document.

Based on previous experience with our annotation pipeline, an average processing time of more than a second per document seems exceptionally high. Especially when processing abstracts, we expect the annotation process to be faster by two or three orders of magnitude. It is thus important to understand how the processing time is measured.

In phase 1 of TIPS, all requests were concerned with a single document only. This means that the processing overhead (handling network connections, communication between subprocesses) per document is maximal. Indeed, the proportion of overhead in the total processing time is substantial: Using the BeCalm web interface, we triggered private requests with either one or ten documents to be annotated. In this (non-representative) test, the observed time difference between the two request sizes was negligible; there were even counter-intuitive examples, where a 10-document request was processed in less time than a single-document request.

Figure 2 shows all major processing steps that contribute to the total time of completing one request (which equals the response time for one document in phase 1 of TIPS). While we do not know the exact details of how the task organisers define the processing time, the depicted interpretation of start and end point is our best guess based on the documentation and on email conversations with the BeCalm team. The figure shows clearly that each request involves three separate network connections (hatched boxes). Compared to annotating an abstract as short as several hundred characters, we estimate the time contribution of the network latency to be inordinate. It is therefore difficult to draw meaningful conclusions from ART.

As another concern, the choice of measuring mean time has a strong bias for statistical outliers on the positive side. For example, if a single request (for whatever reason) takes 100 times longer to complete than the typical case, this has a much larger impact on the mean time than the inverse would have – an unbelievably fast request that is 100 times faster than the typical case. Measuring
median time, however, is much more robust in this scenario, as it gives more of an impression of the typical processing time.

Moreover, statistical outliers can have a negative effect on the expressiveness of the ART measure. If a few requests take many times longer than usual to complete, this is most likely due to a technical incident, such as a network problem or a server component being temporarily unavailable, causing a delay before the request can be processed or responded. Thus, we argue that outliers reflect aspects of the server’s reliability, rather than its performance under normal conditions. Therefore we think that they should be captured by the reliability metrics rather than ART/MTSA/MTDV.

For these reasons, we carried out an alternative analysis of processing time based on the logs of our annotation server. The analysis covers the period starting on February 20, 2017, 11:51 CET until the end of phase 1 on March 31. We excluded all private requests from the analysis as well as all requests received on March 6, since on that day electrical power cuts caused BeCalm’s infrastructure to malfunction. To our understanding, these data are also excluded from the official evaluation. Based on the total number of requests given in BeCalm’s web interface (314,539 requests), there is a substantial overlap between the data sets.

Table 1 shows the results of our log-based analysis. Instead of measuring all steps involved in an annotation request, we restricted the analysis to the steps at the core of the annotation process. The measured period starts with the first log entry for a request, which is written immediately after the web-framework has preprocessed an incoming request, and ends with the last log message which is written just before sending back the formatted annotations (see the span labelled “log-based analysis” in Figure 2). Thus, the analysed periods span only...
Table 1. Processing time analysis based on the server logs. All time measurements are in seconds.

<table>
<thead>
<tr>
<th></th>
<th>requests</th>
<th>min.</th>
<th>max.</th>
<th>average</th>
<th>median</th>
<th>std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed</td>
<td>100,332</td>
<td>0.424</td>
<td>78.154</td>
<td>0.540</td>
<td>0.526</td>
<td>0.440</td>
</tr>
<tr>
<td>Abstract server</td>
<td>130,206</td>
<td>0.079</td>
<td>167.037</td>
<td>0.170</td>
<td>0.099</td>
<td>1.854</td>
</tr>
<tr>
<td>Patent server</td>
<td>36,945</td>
<td>0.079</td>
<td>13.671</td>
<td>0.103</td>
<td>0.096</td>
<td>0.144</td>
</tr>
<tr>
<td>all</td>
<td>267,483</td>
<td>0.079</td>
<td>167.037</td>
<td>0.300</td>
<td>0.111</td>
<td>1.336</td>
</tr>
</tbody>
</table>

one of the three network connections involved in a complete cycle. The start and end point of each request were obtained by parsing the log messages, which are printed with millisecond precision.

The most evident finding is that disregarding the initial and final network connections substantially reduces the measured time span: The average processing time of all requests (last row) is 300 ms, rather than 1069 ms (ART). The median processing time is again much lower (111 ms). It is also interesting to see the differences by origin: Fetching and processing a PubMed abstract takes considerably longer (540 ms on average) than to BeCalm’s abstract (170) and patent server (103). A possible explanation for this discrepancy is the fact that we obtain PubMed abstracts in XML format, while BeCalm provides the abstracts in a flat JSON structure, which is much more lightweight. Another observation is that PubMed abstracts tend to be larger in terms of file size (if only for the additional markup), which might have an impact on transmission time. It might also be that the NCBI servers are simply busier than BeCalm’s in terms of network traffic.

Another conclusion that can be drawn from these statistics is that parallel processing did not pay off in phase 1 of TIPS, which is no surprise. In busy times, the Metaserver issued a request every 2 to 10 seconds. Thus, most of the time, when a new request arrived, the previous one had long been completed, meaning that the “parallel” annotation workers almost never worked in parallel effectively.

3.2 Conclusion

In the current evaluation, where each request asked for a single document only, it is hard to measure the speed of the annotation process. The task’s protocol with three separate network connections for each annotation request entails significant overhead. In future phases of TIPS, where multiple documents per request, larger documents (full-text) and simultaneous requests will be required, our annotation server will be able to better show its strengths. The OGER annotation server is ready now for phases 2 through 4!

References


READ-Biomed-Server: A Scalable Annotation Server Using the UIMA Concept Mapper

Ruichen Teng and Karin Verspoor

School of Computing and Information Systems
The University of Melbourne
Melbourne VIC 3010, Australia
ruichen.teng@gmail.com
karin.verspoor@unimelb.edu.au

Abstract. For the BeCalm Technical Interoperability and Performance of annotation Servers (TIPS) task, we produced a fast dictionary lookup tool implemented as a standalone web service. It is based on a web service wrapper of the UIMA ConceptMapper, originally developed for the BioCreative-IV Comparative Toxicogenomics Database web service annotation task. We focused on annotation of Gene Ontology terms as the target annotation type for TIPS. We integrated this annotator into a scalable, micro-service architecture to build our annotation server. Message queues and thread pools were used to handle high concurrency and heavy computation. We also addressed handling errors and exceptions.

Key words: gene ontology, concept annotation, dictionary lookup, micro-service

1 Introduction

The BeCalm Technical Interoperability and Performance of annotation Servers (TIPS) task required the development of a web service for document annotation, and specifically annotation of named entities. The Reading, Extraction, and Annotation of Documents Biomedical Server (READ-Biomed-Server) is a web service set up to participate in this task, providing detection and annotation of terms in text from a given vocabulary using flexible dictionary-based matching. The server accepts annotation requests consisting of references to documents, and the types of entities that should be identified and annotated in the response. For the TIPS task, READ-Biomed-Server was set up to return annotations of terms in the Gene Ontology [2, 7], using the UIMA ConceptMapper dictionary-based concept recognition tool [10]. This tool has been shown to be effective for concept recognition of terms from large vocabularies [6], and for the Gene Ontology specifically works well when coupled with synonym generation strategies [5]. For TIPS, we emphasise its deployment in a robust and responsive architecture.
2 Annotator Description

2.1 Annotator methods

For the annotation web service, the implementation of the core annotator was largely based on the code developed for Comparative Toxicogenomics Database (CTD) Annotator [9], submitted for the BioCreative-IV CTD Challenge 2013 [1,11]. It is a dictionary-lookup system using the UIMA [3,4] ConceptMapper\(^1\) [10], in which the task of annotating a document is treated as (possibly fuzzy) matching its tokens with terms in the dictionary. For this task we kept the default, strict matching strategy used in the original CTD Annotator [9], namely:

- \texttt{OrderIndependentLookup = false} only match if the order of the tokens is the same as in the dictionary.
- \texttt{FindAllMatches = false} only find the longest match, ignoring any shorter spans within.
- \texttt{SearchStrategy = ContiguousMatch} matched tokens should be adjacent to each other.

2.2 Dictionary

We used the vocabulary of the Gene Ontology\(^2\) from the Gene Ontology project [2,7]. This project aims at building structured representations of human knowledge related to gene function. We extracted all terms and synonyms, resulting in a dictionary of over 46,000 canonical terms and nearly 170,000 synonyms in ConceptMapper dictionary format. Due to memory limitations on the server used for the TIPS task, we did not load the extended GO synonym set [5].

Below is an entry in the dictionary:

```xml
<token canonical="ribosomal subunit export from nucleus">
  <variant base="ribosomal subunit export from nucleus"/>
  <variant base="ribosomal subunit export from cell nucleus"/>
  <variant base="ribosomal subunit export out of nucleus"/>
  <variant base="ribosomal subunit transport from nucleus to cytoplasm"/>
  <variant base="ribosomal subunit-nucleus export"/>
  <variant base="ribosome export from nucleus"/>
</token>
```

The annotator will match any synonym variant, and return an annotation to the canonical term.


\(^2\) [http://geneontology.org/page/download-ontology](http://geneontology.org/page/download-ontology)
2.3 Implementation details

As explained in [9], the CTD annotator is implemented as a standalone web service. It has a client-server architecture in which a client sends a message with the text to be annotated, and the server responds with the annotation result. In addition, the dictionary is transformed into the data structures required by the ConceptMapper tool and loaded into memory [10]. The CTD annotator was adapted for use in TIPS, substituting the dictionaries used for the CTD annotation task with the Gene Ontology dictionary, but otherwise using the same REST-based framework for processing annotation requests and translating UIMA annotation data structures into a correctly formatted response.

3 Web Server Description

3.1 System Architecture

The architecture of our approach is illustrated in Figure 1. Our annotation server has two major components. In the forefront, we implemented a “query dispatcher” type of Http server using Node.js to listen to requests from BeCalm metaserver, following the API defined at http://www.becalm.eu/api. This server does three things:

1. responds to “getState” requests with server state
2. responds to “getAnnotations” requests with ack 200
3. parses “getAnnotations” requests and sends a “query” to the Java Http server

Here, a query consists of the communication identifier (ID) and a series of document identifiers and document sources. The query is then passed on to the Java/Scala module to respond to this query, including requesting documents from different servers and annotating each of the documents. Some requests used in the TIPS evaluation included thousands of documents.

Within the Java/Scala module, in order to handle a large number of concurrent requests given that a request can take a long time to process if the number of documents requested is high, we implemented two message queues: one for storing queries, another one for storing annotation results. When the Java Http server receives a query, it adds the query to Query Queue and spawns some threads in the thread pool to process the queries. The processing includes:

- retrieving document texts from the source servers (Patent, Abstract, PubMed)
- calling the CTD Annotator to obtain annotation results
- putting results in the Result Queue
- sending results to BeCalm metaserver
- handling errors and exceptions

### 3.2 Document representation

Fig. 2: PubMed document Representation

Listing 1 shows the document representation from abstract and patent servers where title and text can be extracted in json values. Figure 2 shows the document representation from PubMed server where it takes more sophisticated parsing to extract title and text.
3.3 Error and exception handling

An important subtask in this competition is error and exception handling. Below are a few examples of errors and exceptions we have handled in our system:

- Failure to send message between components
- Failure to send annotation results (network error, response code not 200)
- Thread-related errors
- Java ApacheHttpClient related errors (connection pool management)

Particularly, if a result fails to be sent, it will be added back to Result Queue, waiting to be resent.

3.4 Multiple annotations over the same span

As shown in Listing 2, it is possible for the CTD annotator to identify multiple different annotations over the same span of text, for instance when the text contains an ambiguous term – a single string that maps to multiple concepts. In the standard ConceptMapper implementation in UIMA, each concept that the string matches is represented as a separate annotation. However, we found that we were unable to maintain this representation in the TIPS framework. Instead, we needed to return a single annotation, with each of the relevant matched canonical terms included in a comma-separated list in the database id field of the annotation response, as shown in Listings 3. To produce this alternative representation, we had to post-process the annotations to identify and combine those that overlapped the same span of text.
3.5 Performance

Table 1 summarises the key performance metrics of the READ-Biomed-Server in the TIPS evaluation, returning Gene Ontology term annotations for requested
<table>
<thead>
<tr>
<th>Category</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>ART (average response time)</td>
<td>3.74128s</td>
</tr>
<tr>
<td>MAD (mean annotations per document)</td>
<td>8.896</td>
</tr>
<tr>
<td>MTS (Mean time in seconds seek annotations)</td>
<td>0.42042</td>
</tr>
<tr>
<td>MTDV (Mean time in seconds per document volume)</td>
<td>0.00307</td>
</tr>
<tr>
<td>Mean processing time per document</td>
<td>3.74125</td>
</tr>
<tr>
<td>MPDV (total predictions per document volume)</td>
<td>0.0073</td>
</tr>
</tbody>
</table>

Table 1: READ-Biomed-Server System Performance

<table>
<thead>
<tr>
<th>Time(s)/documents</th>
<th>Time(s)/predictions</th>
<th>#Documents</th>
<th>#Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00962</td>
<td>0.00098</td>
<td>4807</td>
<td>47100</td>
</tr>
<tr>
<td>0.00986</td>
<td>0.00103</td>
<td>4837</td>
<td>46266</td>
</tr>
<tr>
<td>0.00932</td>
<td>0.00101</td>
<td>2900</td>
<td>26757</td>
</tr>
<tr>
<td>0.00873</td>
<td>0.00091</td>
<td>2813</td>
<td>26915</td>
</tr>
<tr>
<td>0.01222</td>
<td>0.00124</td>
<td>1169</td>
<td>11527</td>
</tr>
</tbody>
</table>

Table 2: Performance for requests with over 1000 documents

documents. As shown in Table 2, our system has very good performance for requests that contain a large number of documents (in thousands). This type of request did not occur during the TIPS evaluation although it was initially proposed as a relevant performance requirement. This was therefore something we had in mind when designing our system.

3.6 Discussion

The micro-service oriented architecture we have employed, illustrated in Figure 1, is highly scalable. Each component (CTD Annotator, Message Queues, Thread Pool) can be easily detached from the module and “scaled out”. In production environments, this architecture is beneficial as it has good separation of concerns [8], due to modularity and encapsulation, and no single point of failure.

During the evaluation period, we modified the server several times in order to improve the system performance. The changes we implemented were focused on optimising thread architecture for our CPU and reducing polling message queues to save system resources, but the annotator module remained unchanged.

4 Conclusions

For the TIPS evaluation, we adapted a UIMA ConceptMapper-based annotation server previously developed for BioCreative IV. We primarily focused on implementing our annotation server within a scalable architecture, as well as emphasising robustness and error handling. Since our annotator had already been implemented as a web service, we decided to build loosely coupled components
around it. The end result is an annotation server with a micro-service oriented architecture where different components are implemented as web services.

References

Neji: DIY web services
for biomedical concept recognition

André Santos and Sérgio Matos

DETI/IEETA, University of Aveiro,
Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{andre.jeronimo, aleixomatos}@ua.pt
http://bioinformatics.ua.pt/

Abstract. The BioCreative V.5 task on technical interoperability and performance of annotation servers evaluated the provision of named entity annotations through web services.
This paper describes Neji, a web-services ready text processing and annotation framework, and shows how its modular and flexible architecture allowed simple adaptation to the requirements of the task. The configured service offers the annotation of eight concept types through five dictionaries and three machine-learning models, and has support for a variety of input and output formats.

Key words: Named entity recognition, biomedical text mining, web-services

1 Introduction

The BioCreative\(^1\) community has promoted the development and evaluation of biomedical information retrieval and extraction tools, through the organization of various shared tasks focused on document triage, entity recognition (e.g. genes, chemicals) and relation extraction (e.g. protein-protein interactions, chemical-disease associations).

The technical interoperability and performance of annotation servers (TIPS) task, part of BioCreative V.5, focused on evaluating the technical aspects of providing inter-operable web services for named entity recognition [1]. We describe the latest developments of Neji, a modular framework for biomedical text processing and concept recognition, including the in-built support for REST web-services. Neji web server was used for participation in the TIPS task with a concept recognition service configured for annotating eight concept types through five dictionaries and three machine-learning models.

\(^{\ast}\) Corresponding author
\(^{1}\) http://www.biocreative.org/
2 System description

Neji\textsuperscript{2} is a flexible and extensible concept recognition framework specially optimized for biomedical text \cite{2}. As illustrated in Figure 1, the main component in Neji’s architecture is the processing pipeline, which manages and sequentially executes a series of independent modules, each responsible for a specific processing task. Each module is implemented as a custom deterministic finite automaton (DFA) using Monq\textsuperscript{3}jfa, a library for fast and flexible text filtering with regular expressions.

Neji includes natural language processing modules, based on GDep \cite{3} and Apache OpenNLP \cite{4}, for machine learning (ML) and dictionary-based concept recognition, and for post-processing, including parentheses correction, abbreviation resolution, and false positives filtering. The machine learning component is based on Gimli \cite{4}, and makes use of MALLET \cite{5} for providing simple training and application of Conditional Random Fields (CRFs) models \cite{6}.

Neji’s modular architecture allows end users to configure the processing of documents according to their specific requirements, by simply combining existing modules for reading from and writing results to a variety of supported formats, and by using the appropriate dictionaries and machine learning models according to the concept types of interest, all of which can be achieved through the simple command line interface or through the provided API. Additionally, Neji can be extended by creating custom modules for reading from or writing to specific formats, or by adding new processing modules.

\textsuperscript{2} Available from https://github.com/EMDSoftware/neji
\textsuperscript{3} https://github.com/HaraldKi/monqjfa
\textsuperscript{4} https://opennlp.apache.org/
Fig. 2. Simple definition of an annotation service using dictionaries and ML models.

Neji web server is built on top of the Neji framework, proving a straightforward way of defining and managing annotation services, each accessible through a REST API end-point. The server also provides simple interfaces for managing annotation resources (dictionaries, or ML models previously trained with Neji) and for creating annotation services based on those resources, as shown in Figure 2. A web page with interactive annotation is also created for each service, allowing inspection of the annotation results and offering several exporting options to different formats (Figure 3). Neji server was developed in Java and uses a Jetty server and a SQLite database for storing the service configurations. The client side interfaces are based on HTML5, CSS3, JavaScript and Bootstrap, offering support on all modern browsers and platforms.

For the TIPS task, we developed four new writer modules to support all the output formats proposed in the task, namely TSV, JSON, BioC and BioC JSON. Additionally, the REST API was extended and adapted according to the task requirements. An annotation service was configured that allows annotating the following concept types: Anatomic Component, Diseases, Subcellular structure, Tissue and Organ, and Organism, through dictionaries compiled from the UMLS Metathesaurus, and Chemicals, Genes and Proteins, through machine learning models trained on the BioCreative V CHEMDNER corpus [7], and Mutations, using an ML model trained on the tmVar corpus [8]. The server accepts raw text as input, as well as PubMed and PubMedCentral identifiers, which are used for obtaining the documents to be processed. The output format and annotated
Fig. 3. A user interface is provided for each annotation service.

concept types can be configured by using the custom API parameters, as shown below. By default, all concept types are returned.

```
{
  "format": "BECALM_JSON",
  "groups": {ANAT, DISO}
}
```

3 Results

The annotation service for participating in the TIPS task was configured to run with 23 threads and was deployed on a Docker container with 32GB of memory over a server with 24 processing cores.

We performed a simple evaluation in terms of processing times by submitting several requests to the server, with different number of documents. We followed the procedure defined for the TIPS task, in which the document text is obtained from the BeCalm abstract and patent servers, and measured the time since the request was submitted to the Neji annotation service until the annotation results were returned. We observed average processing times ranging from 11.5 seconds for abstracts and 9.35 seconds for patents when annotating a single document, to 0.347 seconds per abstract and 0.173 seconds per patent when annotating sets of 1000 documents (Table 1).

We also measured the processing time for documents sent directly to the annotation server, that is, without request to the BeCalm document servers. In these tests, the full Craft corpus [9], composed of 67 full text documents containing more than 560000 tokens in total, was annotated in 15 minutes, which corresponds to an average processing time of 13.55 seconds per document and a processing speed over 600 tokens per second. Documents were sent to the annotation service one at a time and as raw text.
Table 1. Average processing times, in seconds, for documents obtained from the Be-Calm document servers.

<table>
<thead>
<tr>
<th>No. documents</th>
<th>abstracts</th>
<th>patents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.5</td>
<td>9.35</td>
</tr>
<tr>
<td>100</td>
<td>0.421</td>
<td>0.236</td>
</tr>
<tr>
<td>1000</td>
<td>0.347</td>
<td>0.173</td>
</tr>
</tbody>
</table>

Acknowledgments This work was supported by Portuguese National Funds through FCT - Foundation for Science and Technology, in the context of the project IF/01694/2013. Sérgio Matos is funded under the FCT Investigator programme.

References

NTTMU-SCHEMA BeCalm API in BioCreative V.5

Hong-Jie Dai*1,2, Mira Anne C. dela Rosa3, Ding-You Zhang1, Chung-Lin Chen1, Chen-Kai Wang3

1Department of Computer Science and Information Engineering, National Taitung University, Taitung, Taiwan, R.O.C, 2Interdisciplinary Program of Green and Information Technology, National Taitung University, Taitung, Taiwan, R.O.C, 3Institute of Chemistry, Academia Sinica, Taipei, Taiwan, 4Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan, R.O.C.

*hjdai@nttu.edu.tw; thegreatseer@gmail.com;
foodingding@gmail.com;
u10411237@ms104.nttu.edu.tw;
dennisckwang@gmail.com

Abstract. With the emerging of new experimental techniques, there has been a remarkable increase in the amount of available biomedical data. Processing and mining large volumes of data in chemistry has now presented a challenging issue. In order to deal with the challenge, we developed SCHEMA (Spark-based CHEMical entity recognizer), a robust and efficient chemical entity recognition system on top of Apache Spark. SCHEMA is developed by following the asynchronous queue design pattern, which has been employed in service-oriented architecture for providing scalable and resilient services. SCHEMA can retrieve patents in a form of unstructured free text from different websites and recognize chemical named entities described in them. To programmatically interact with SCHEMA, a restful Web application programming interface is provided. By using the custom request tests of the BeCalm (Biomedical annotation meta-server) platform, the test results illustrated that SCHEMA can process 5,000 patients within 5 minutes, indicating an average of only 0.06 second for processing one patent including the data fetch and analysis time.

Keywords. Chemical named entity recognition; Spark; parallel processing

1 Introduction

The emergence of new experimental techniques such as high throughput screening along with the development of automatic data mining al-
gorithms bring forth large quantities of chemistry data. In addition to experimental data in publicly available databases such as PubChem [1], chemical patents represent one of the rich resources for chemical information. There is an increasing demand to efficiently mine the large scale of data in chemistry for the future development of chemical, pharmaceutical, agrochemical, biotechnological and fragrance industries [2].

In order to promote the effective access and integration of multiple text mining systems for processing unstructured document collections, BioCreative has had introduced the idea of the meta-services for biomedical information extraction since 2008 [3]. Despite the relevance of these previous efforts, some crucial aspects have been insufficiently or only partially addressed including continuous evaluation, extraction of textual content from heterogeneous sources, harmonization of multiple biomedical text annotations and visualization and comparative assessment of automatic and manual annotations. The BeCalm (Biomedical annotation meta-server) platform [4] in BioCreative V.5 provides the first solution to address the above mentioned issues.

As one of the participants in the BioCreative V.5 TIPS (Technical interoperability and performance of annotation servers) task, the NTTMU team developed SCHEMA\(^1\), a Spark-based CHEMicAl entity recognition system, and implemented a REST (Representational State Transfer) application programming interface (API) to continuously listen and respond to the requests from BeCalm and other end users.

2 Method

We followed the messaging pattern in service-oriented architecture to develop SCHEMA, which enables the core of SCHEMA can be interacted with other services or applications through a loosely coupled and asynchronous message-based communication model. Figure 1 shows the detail workflow. The core of SCHEMA is a program runs on Apache Spark\(^2\). The SCHEMA core itself runs as a background process in an operation system and monitors messages sent to the request queue. When a processing request is posted to the request queue, the SCHEMA core retrieves that message and removes it from the queue. The request is then processed by several text mining workers implemented on Apache Spark.

---

\(^1\) SCHEMA is available at http://210.240.162.49/SCHEMA/
\(^2\) Apache Spark (http://spark.apache.org) is a general engine for large-scale data processing.
engine. Through the above asynchronous queuing design pattern, SCHEMA can be deployed in many distributed environments and provide asynchronous, scalable and resilient services.

For interacting with the SCHEMA core through, we developed a REST web service, the SCHEMA Web Server in Figure 1, which continuously listens requests from BeCalm or authorized end users. The authorized request is then placed in a message queue waiting for processing. When the SCHEMA core completes the processing request, the results are placed in the response queue which will be later consumed by the thread of the SCHEMA web server and delivered to the requester in the format of BeCalm JSON.

![SCHEMA workflow diagram]

Figure 1. SCHEMA workflow.

The SCHEMA core is a Spark application consists of

1. SCHEMA Driver: A Spark driver program that monitors the message queues and launches parallel text mining operations on the SCHEMA cluster. In the current implementation, the driver defines the distributed datasets based on the processing request and then asks the SCHEMA works to execute various operations to the datasets.

2. SCHEMA workers: The distributed agents that execute the text mining tasks. The tasks in current implementation includes: 1) download patterns from remote servers including PMC, PubMed and the patent server provided by the TIPS organizers, 2) sentence splitting, 3) tokenization, 4) part-of-speech tagging, 5)
chemical named entity recognition based on our previous work [5], and 6) recognition refinement.

In the current implement, the SCHEMA core was configured with 24 cores run on three virtual machines.

3 Results and Discussion

Figure 2 shows the web console of SCHEMA core. The results illustrated the processing time for each BeCalm request in the TIPS evaluation phase 1. In the phase, BeCalm requests for processing one document per request to validate the implementation of our annotation server within variable time intervals. As one can see that the SCHEMA core spends around 40 seconds to process one patent including the data fetch time from remote databases and the text mining processing time.

![Figure 2. SCHEMA core web console for processed jobs.](image)

Figure 3 shows the evaluation interface provided by BeCalm platform. Here we used the custom request function to adjust the number of documents in a request to examine the processing time of SCHEMA. As one can see that SCHEMA spends around 50 seconds to process 10 patents. With the numbers of patents per request increasing from 10 to 300, we did see significant increase of processing time of SCHEMA. We can also observe that the time for processing one patent was decreased from ~5 seconds to 0.192 seconds. The time for generating one predication was also reduced from 1.512 seconds to 0.050 seconds.

In addition, Figure 4 shows the processing time of requests containing more than 1000 patents. SCHEMA can process 2000 documents per request within two minutes. We can also observe that the average pro-
cessing time per document did not increase when the number of documents per request increase. All of the above results illustrate the reliability and the power of parallel processing of SCHEMA core.

Figure 3. BeCalm evaluation interface for processing 10 to 400 patents.
Figure 4. BeCalm evaluation interface for processing requests with more than 1000 patents.

Figure 5. The official historical server response timeout record.

Figure 5 and 6 show the official evaluation results on the BeCalm platform. Started from 2017/1/23, SCHEMA processed 292,540 requests and generated around 493k, 648k, and 482k predictions for the patent server, abstract server and PubMed, respectively. At the end of March, we shutdown SCHEMA for one week because we had other computing tasks required for running on the VM server. However, we forgot to stop the SCHEMA web server, therefore it continues to accept request from BeCalm. After we restarted the SCHEMA driver, it started to process all
received requests in the request queue and replied to BeCalm, which may one of the reason lead to the large max processing time.

Among all requests, SCHEMA generated 5,853 exceptions. Most of the exceptions were occurred with an error message of “Request getState not retrieve data” which occurred constantly since the beginning of April after we restart our SCHEMA server. Since then BeCalm cannot receive the correct running state of SCHEMA and throws aforementioned exceptions. We don’t know the reason cased the errors.

Figure 6. Statistics by document provider.
4 Conclusion

In the paper, we give a briefly introduction of the development of NTTMU team’s SCHEMA. The architecture of SCHEMA is flexibly and its computing power can be easily scale up by adding more cores within the Apache Spark platform. In the future, we will review the performance report evaluated by the TIPS task and study the effect of different configurations of our SCHEMA cores.

5 Acknowledgment

The study was supported by Ministry of Science and Technology (MOST) 105-2221-E-143-003 and Ministry of Education-Talent Cultivation Program for Advanced Mobile Broadband Technology-Big Data module.

REFERENCES

5. Dai HJ, Lai PT, Chang YC, Tsai RT: Enhancing of chemical compound and drug name recognition using representative
Micro-RNA Recognition in Patents in BioCreative V.5

Chen-Kai Wang¹, Hong-Jie Dai²,³*, Nai-Wun Chang⁴

¹Graduate Institute of Biomedical Informatics, Taipei Medical University, Taipei, Taiwan, R.O.C., ²Department of Computer Science and Information Engineering, National Taitung University, Taitung, Taiwan, R.O.C., ³Interdisciplinary Program of Green and Information Technology, National Taitung University, Taitung, Taiwan, R.O.C., ⁴Institute of Information Science, Academia Sinica, Taipei, Taiwan

dennisckwang@gmail.com; hjdai@nttu.edu.tw; naiwun@gmail.com;

Abstract. MicroRNAs (miRNAs) have been considered as good candidates for early detection or prognosis biomarkers for various diseases. Patents related to methods of identifying, isolating and amplifying miRNAs and potential use of miRNAs as biomarkers for cancers are increasing rapidly. In this work, we extend our miRNA recognition method based on the statistical principle-based approach and develop a web service followed the communication protocol defined by the biomedical annotation meta-server (BeCalm) platform to provide a service of miRNA recognition. The method can achieve an F-score of 0.988 for miRNA recognition on a manually annotated test dataset. During the participation of the BioCreative V.5 Technical Interoperability and Performance of Annotation Servers (TIPS) task, we set up our web service successfully and it can exchange the status message with the BeCalm platform and process the requests from BeCalm. Unfortunately, we met technical problems to send back the annotation results to the BeCalm platform.

1 Introduction

MicroRNAs (miRNAs) are small non-coding RNAs of approximately 23 nucleotides, which negatively regulate the gene expression at the post-transcriptional level. Recently miRNAs have been considered as good candidates for early detection or prognosis biomarkers for various diseases. Patents related to methods of identifying, isolating and amplifying miRNAs and potential use of miRNAs as biomarkers for cancers

* Corresponding author
are therefore increasing rapidly. To facilitate the understanding of the state-of-the-art researches and applications of miRNAs, we present a REST (Representational State Transfer) web service for miRNA recognition.

2 Method

Our RESTful service contains three main components. The first is the data retrieval component which retrieves patents from remote data sources. In our current implementation, four sources are supported. The first and two data sources are PubMed Central (PMC) and PubMed. We used NCBI E-utils to fetch requested data from the two data sources. The third and fourth sources are the pattern server and the abstract server released by the TIPS task.

The core of the web service is a miRNA recognition component based on our statistical principle-based approach (SPBA) [1]. The component integrates several natural language process and information extraction modules to process downloaded patents. Given a patent, MedPost [2] is used to split text into sentences and generate tokens for each sentence. We then employed our SPBA-based miRNA recognition method to recognize miRNAs in the preprocessed sentences. Our miRNA recognizer is developed based on the corpus released by S Bagewadi, T Bobic, M Hofmann-Apitius, J Fluck and R Klinger [3].

The training phase of SPBA consists of three main steps. The first is knowledge construction. In this step, we represent the knowledge related to miRNA terms through semantic slots and principles manually or semi-automatically with Information Map [4]. Figure 1 illustrates the hierarchical knowledge structure constructed for representing a miRNA in our approach. The root node is “miRNA” indicating that the structure represents the knowledge for miRNA names. The first child node of the root node is the “SLOT” node, under which we define the fundamental semantic unit (i.e. slot) for the root node. Consider the miRNA name as an example. Similar contents can be found among descriptions about miRNAs, which form the backbone of miRNA’s slots. For instance, both the miRNA “cel-miR-123-5p” and “hsa-microRNA-24-3P” consists of a species (cel and hsa), the indicating word “miRNA” and a hairpin that possess unique feature in representing a miRNA.
After constructing the knowledge, the principle generation step was applied. In this step, slots are assembled and summarized by observing the arrangement of principle slots which can accomplish the miRNA recognition task. For example, both the miRNA “cel-miR-123-5p” and “hsa-microRNA-24-3P” can be designated using the following combination of slots “[Species][miRNA][order][Hair-pin]”. Here we use brackets to enclose a slot name for representing a slot. For example, “[Species]” is a slot that encodes the species in which the miRNA appears. “[miRNA]” is the slot representing the word indicating an occurrence of a miRNA name.

![Diagram of semantic slots and principles for miRNA]

Figure 1. Semantic slots and principles defined for miRNA.

Lastly, a flexible principle matching algorithm allowing insertion, deletion, and substitution is applied to extract miRNAs represented by the compiled principles in the given text. Unlike traditional template matching that involves rigid left-right relation of slots in a sentence, a scoring
criteria during principle alignment was used in SPBA, in which the collocation and bigram statistics is incorporated to estimate matching scores. During the principle matching procedure, we score those possible candidate principles based on matched slots, slot relations and insertions. Each exactly matched slot gets a score of 4. If there are insertion/deletion/substitution in the string, the scoring mechanism will assign scores accordingly. We calculate the score of an insertion by gathering its left (resp. right) bigram statistics with its neighboring left (resp. right) slots in the training set. A substitution is either a partial match or a category match of the slot, which is assigned a score of 1. The final score of a principle is the sum of all the scores of this principle. The length of a principle is used as the threshold to determine whether this principle is matched or not. Finally, the longest principle or a principle which contains the most slots will be considered as matched.

The last component is the BeCalm communication module. The module listens requests from BeCalm platform [5], check the correctness of the authentication key provided in each request, authorized the requests and then respond to BeCalm with an acknowledge message. All approved requests are sent to the first component for downloading patents from remote data sources. The download patents are then processed by the core of our service for miRNA recognition. Finally, the recognized miRNAs are encoded in the JSON format defined by the TIPS task and send to BeCalm through the saveAnnotations method provided by the BeCalm platform.

3 Results and Discussion

![Image of results and discussion]

Proceedings of the BioCreative V.5 Challenge Evaluation Workshop
Figure 2. Statistical results of our web service.

Figure 2 shows the official statistical results of our web services. As one can see that our server can successfully receive the request from BeCalm. Most requests were requested for data from the patent server. The results illustrate that our server did not generate any predictions for all of the three data sources. We tried to debug our server by store the downloaded data from the remote sources. In total of 818 patents were saved for our analysis. We then applied an offline processing for all the downloaded data. Table 1 shows the statistical results on the dataset. We can observed that our miRNA recognizer can recognize miRNAs from the patent documents. The zero predictions shown in Figure 2 seems owing to the failure of our communication module in replying the annotation results to the BeCalm platform.

<table>
<thead>
<tr>
<th># of Processed Patents</th>
<th># of Predictions</th>
<th># of Patents Containing Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>818</td>
<td>34</td>
<td>15</td>
</tr>
</tbody>
</table>

4 Conclusion

In the paper, we introduce our miRNA recognition web service developed for the BioCreative V.5 TIPS task. Although our miRNA recognizer can recognize miRNAs mentioned in patents, our service fails to response processing results to BeCalm. In the future, we will fix the defeats and manually annotate miRNAs observed in collected patents to evaluate the performance of our miRNA recognition in patent documents. We will also study the distributions of recognized miRNA mentions among PubMed/PMC articles and chemical patents.

5 Acknowledgment

The study was supported by Ministry of Science and Technology (MOST) 105-2221-E-143-003 and MOST-106-2922-I-038-014.
REFERENCES


