Homework 1 Report (afandria)

Andrew Fandrianto

My gene mention tagger is a combination of HMM chunking, standard named entity recognition, inverse set membership. The UML diagram is on the last page.

The type system included only one type, Gene_Type, which included fields:

- identifier: sentence identifier, e.g. "POOO12039782"
- start: index where the gene mention started in the sentence without spaces
- end: index where the gene mention ended in the sentence without spaces
- content: what the mention was, e.g. "cancerous hepatic triacylglycerol lipase prion ester"

I felt no need to subdivide the types into any others, because:

1. the input format was already segmented into sentences with unique identifiers
2. output only considers individual gene mentions

I did not use any patterns aside from the built-in Factory pattern for the Gene_Type.

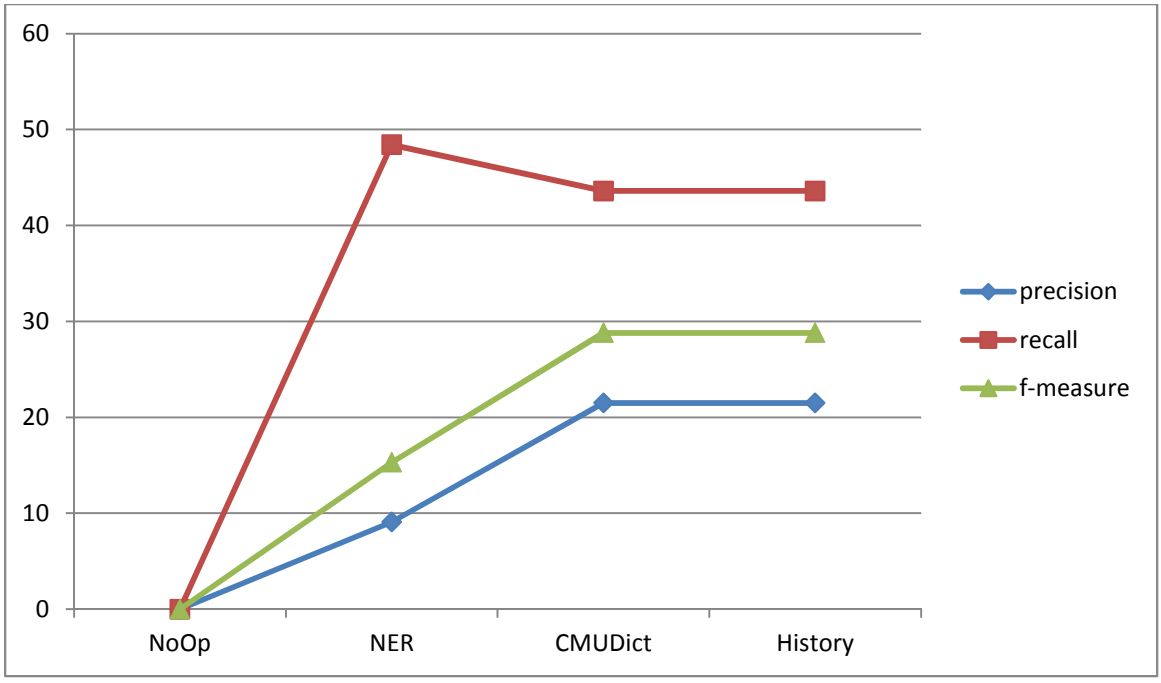
As for algorithms used, I included the standard Stanford NER and LingPipe's HMM chunker using their trained bio-gene model. I used CMUDict to filter for common English words. An enumeration:

1. LingPipe HMM chunker (LingPipe bio-genetag model)
2. Stanford NER (Stanford model)
3. CMUDict set exclusion
4. History set membership

The data flow is as follows. The document gets imported first into both the NER and HMM chunker. Both extract a base set of named entities. The NER is run after, so it only adds entities that are not already present. That base set is then filtered by excluding words in CMUDict + some heuristics. Our filter mechanism here is increasing or decreasing confidence, using a threshold to determine inclusion in our gene mention tag set. Finally, the sample set history boosts scores that were found. Finally, it will output the final set. In between each Annotator, we use an Evaluator to output precision, recall, and F-measure.

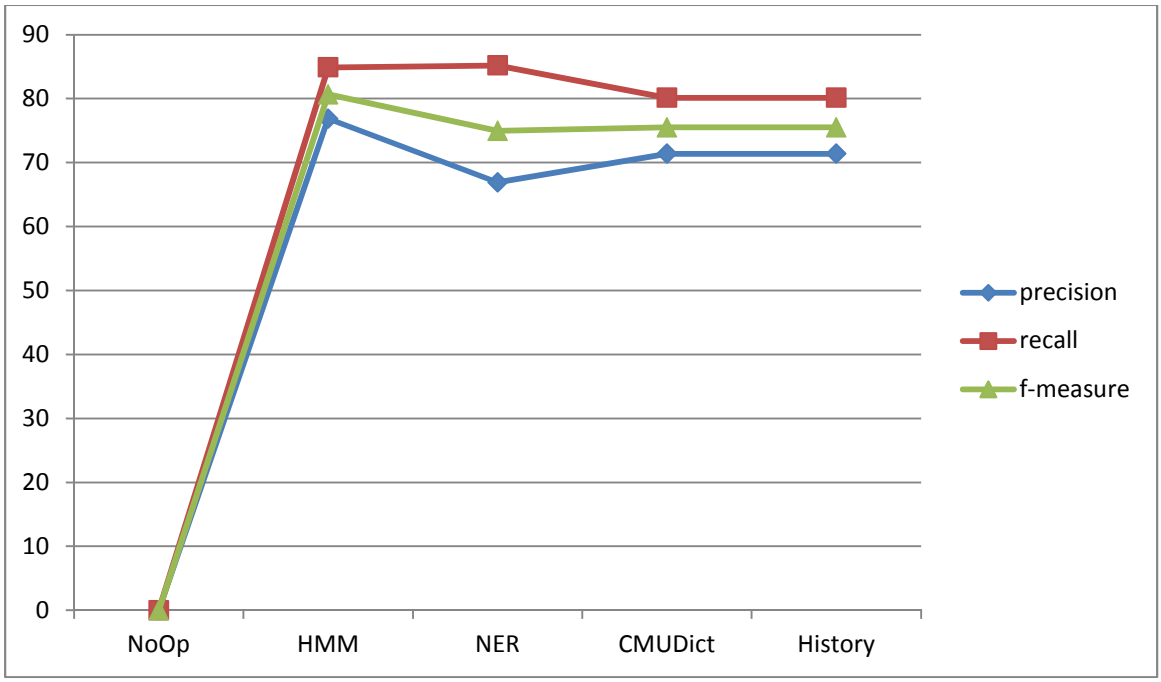
An optimization here would be to have HMM and NER run in parallel, using an adaptor to combine the results of the two. CMUDict and History need to be sequential though.

We can first look at the pipeline without LingPipe (since it's too good), so we can look at exaggerated effects of NER, CMUDict, and History.



NER alone has pretty bad results, with precision down in the single digits and recall at 50%. As expected, CMUDict filtering increases precision while recall dips slightly. However, this tradeoff is worth it according to the F-measure. The history annotation does nothing here (since we use the test set for the train set and therefore will just boost things it has already seen), but I expect it will contribute slightly in an unseen dataset. Note that the history does not extract features from the input document but boosts those already included by the base extraction.

Now, to include HMM chunking operation to the pipeline…



We jump to the performance of LingPipe's HMM annotator with 85% recall, 77% precision. This is also unsurprising because it's a specific model (vs Stanford NER) with the capability to capture word contexts. Adding the NER results to the mix bump up recall almost unnoticeably and drag down precision. Similar to the previous graph, CMUDict filters some unwanted items, and history annotation does nothing. Obviously we'd be better off just using LingPipe, but showing the results of the filtering are fun and intuitive to see.

As advertised, it's very easy to include new Annotators, Consumers, etc. UIMA is the best.

Performance wise, NER is has a very low time/reward. Other things were pretty quick.

```
Component Name:
GeneAnnotationCollectionReaderDescriptor
Event Type: Process
Duration: 97ms (0.29%)
Result: success
Component Name: NoOpAnnotator
Event Type: Analysis
Duration: 1ms (0%)
Component Name: NoOpAnnotator
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: PostNoOpAnnotationEvaluation
Event Type: Analysis
Duration: 42ms (0.12%)
Component Name: PostNoOpAnnotationEvaluation
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: HMMAnnotator
Event Type: Analysis
Duration: 4317ms (12.7%)
Component Name: HMMAnnotator
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: PostHMMAnnotationEvaluation
Event Type: Analysis
Duration: 191ms (0.56%)
Component Name: PostHMMAnnotationEvaluation
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: NNPAnnotator
Event Type: Analysis
Duration: 28636ms (84.21%)
Component Name: NNPAnnotator
Event Type: End of Batch
Duration: 0ms (0%)
```

```
Component Name: PostNNPAnnotationEvaluation
Event Type: Analysis
Duration: 56ms (0.16%)
Component Name: PostNNPAnnotationEvaluation
Event Type: End of Batch
Duration: 0ms (0%)Component Name: CmuDictAnnotator
Event Type: Analysis
Duration: 276ms (0.81%)
Component Name: CmuDictAnnotator
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: PostCmuDictAnnotationEvaluation
Event Type: Analysis
Duration: 53ms (0.16%)
Component Name: PostCmuDictAnnotationEvaluation
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: HistoryAnnotator
Event Type: Analysis
Duration: 88ms (0.26%)
Component Name: HistoryAnnotator
Event Type: End of Batch
Duration: 0ms (0%)
Component Name: PostHistoryAnnotationEvaluation
Event Type: Analysis
Duration: 75ms (0.22%)
Component Name: PostHistoryAnnotationEvaluation
Event Type: End of Batch
Duration: 1ms (0%)
Component Name: AnnotationPrinter
Event Type: Analysis
Duration: 171ms (0.5%)
Component Name: AnnotationPrinter
Event Type: End of Batch
Duration: 0ms (0%)
```

UML Diagram of HW1 system