Before this course I don't have background in language processing at all. Thanks to the this project I have got base knowledge about engineering and developing of intelligent information systems that process natural human languages and text corpora from different fields.

Project based on the Apache UIMA Tool where UIMA stands for Unstructured Information Management Applications. This is the tool that enables other applications to process large amounts of unstructured information in order to discover hidden knowledge that is user-relevant. With UIMA we can decompose our application to more simple components that interact together and could be easy described with XML-metadata descriptor files. The language that was used to develop application is Java but written XML-descriptors are platform-independent and could be used with C++ based language processing tool.

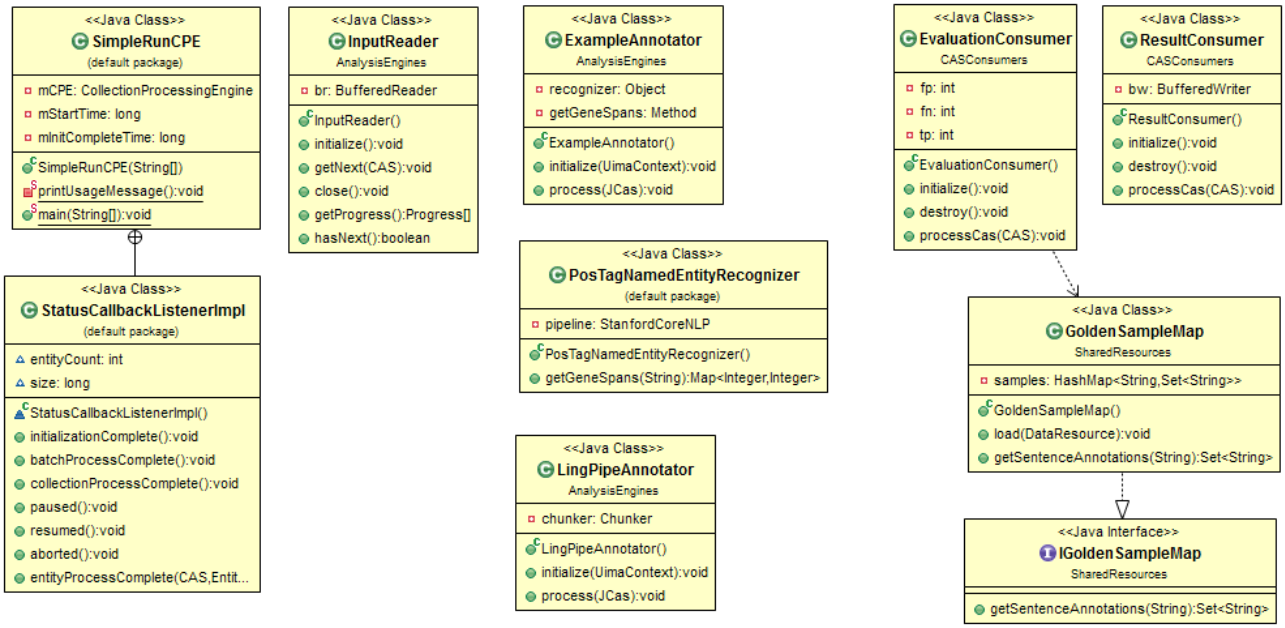

Figure 1: UML-diagram of language processing engine.

The architecture of application is pretty simple. There is the *class SimpleRunCPE* that has runnable *main()* method. It parses *cpeDescriptor.xml* file that stores base information about the structure of built language processing engine. Engine consists of three main logical parts: *CollectionReader*, *AnalysisEngine*, and *CASConsumer*. *CollectionReader* is represented by InputReader that reads information from the source file line by line and mark very first an rough information like sentence id and sentence text. Finally processed lines stored as the CASes inside UIMA.

```
Parsing CPE Descriptor
Instantiating CPE
Adding annotator tokenize
Adding annotator ssplit
Adding annotator pos
Loading default properties from tagger edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/english-left3words/english-left3words-distsim.tagger ... done [2,8 sec].
Running CPE
To abort processing, type "abort" and press enter.
CPM Initialization Complete
=====================================
Precision: 10,25
Recall: 54,67
F1: 17,27
=====================================
Completed 15000 documents; 2275538 characters
Total Time Elapsed: 45463 ms
Initialization Time: 5665 ms
Processing Time: 39798 ms
```

Figure 2: Poor quality metrics for ExampleAnnotator based on POS-tagging

An ***Analysis Engine*** (AE) is a component that analyzes artifacts (e.g. documents) and infers information about them [1]. For this project two analysis engines was built. *ExampleAnnotator* – very simple annotator based on the Stanford CoreNLP tool [2] that uses part of speech annotation to find nouns some of which is probably gene entities. Nevertheless that part of speech annotation with Stanford CoreNLP tool works well and with high quality the base idea is insufficient for proper gene items extraction. As was shown later with the help of EvaluationConsumer the quality of such analysis is very low (Figure 1).

```
Parsing CPE Descriptor
Instantiating CPE
Running CPE
To abort processing, type "abort" and press enter.
CPM Initialization Complete
=======================================
Precision: 76,85
Recall: 84,88
F1: 80,67
=======================================
Completed 15000 documents; 2275538 characters
Total Time Elapsed: 13975 ms
Initialization Time: 1814 ms
Processing Time: 12161 ms
```

Figure 3: Higher precision, recall and F1 metrics of LingPipeAnnotator based on HMM

The second more complicated and specially sharpened for gene entities extraction tool kit is LingPipe Core [3] that uses Hidden Markov Model to extract gene chunks. A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. The mathematics behind the HMM was developed by L. E. Baum and coworkers. It is closely related to an earlier work on optimal nonlinear filtering problem by Ruslan L. Stratonovich, who was the first to describe the forward-backward procedure. [4]

The other method that was used to train engine based on input data – Conditional Random Fields (CRFs) that is implemented in Stanford CoreNLP. CRFs are a type of discriminative undirected probabilistic graphical model. It is used to encode known relationships between observations and construct consistent interpretations. It is often used for labeling or parsing of sequential data, such as natural language text or biological sequences and in computer vision. Specifically, CRFs find applications in shallow parsing, named entity recognition and gene finding, among other tasks, being an alternative to the related hidden Markov models. In computer vision, CRFs are often used for object recognition and image segmentation. [5] For training model gene corpora from this source http://www.ncbi.nlm.nih.gov/pubmed/15960837 and http://www.ncbi.nlm.nih.gov/ was used. It was not included in final deliverable because shown lower then LingPipe results.

After gene sequences extracted from the input data we can evaluate efficiency of used annotators with EvaluationConsumer using precision, recall and F1 metrics and save data to output file with ResultConsumer. If you want to check the correctness of EvaluationConsumer you have to uncomment its description in *cpeDescriptor.xml*

---

[1] UIMA Tutorial at http://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html
[2] The Stanford Natural Language Processing Group download page at http://nlp.stanford.edu/software/index.shtml
[3] LingPipe Core tool kit download page at http://alias-i.com/lingpipe/web/download.html
[4] Hidden Markov Model on Wikipedia at https://en.wikipedia.org/wiki/Hidden_Markov_model
[5] Conditional Random Fields on Wikipedia https://en.wikipedia.org/wiki/Conditional_random_field

During work on this assignment large variety of information sources was used:

1. GENETAG: a tagged corpus for gene/protein named entity recognition http://www.biomedcentral.com/1471-2105/6/S1/S3
2. Evaluating gold standard corpora against gene/protein tagging solutions and lexical resources http://www.jbiomedsem.com/content/4/1/28
3. MeInfoText 2.0: gene methylation and cancer relation extraction from biomedical literature http://www.biomedcentral.com/1471-2105/12/471/
4. UIMA Tutorial at http://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html
5. The Stanford Natural Language Processing Group download page at http://nlp.stanford.edu/software/index.shtml
6. LingPipe Core tool kit download page at http://alias-i.com/lingpipe/web/download.html
7. Hidden Markov Model on Wikipedia at https://en.wikipedia.org/wiki/Hidden_Markov_model
8. Conditional Random Fields on Wikipedia https://en.wikipedia.org/wiki/Conditional_random_field