Benjamin's Named Entity Recognition Exercise

**( B - NER – E )**

Semi-supervised learning Approach

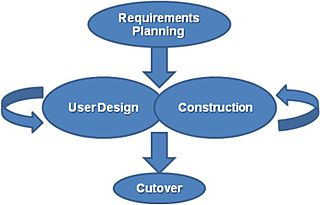
Contents

1. Software Development Life Cycle – SDLC 2
2. Remote Programming Exercise – Instructions 3
3. Problem Domain 3
4. Model the Solution 4
5. Proof of Concept 5
6. Use-Case 6
7. Acceptance Test 6
8. Software Development Life Cycle –

The Agile Unified Process (Agile UP) is a streamlined approach to software development based on IBM's Rational Unified Process (RUP). The Agile UP lifecycle is serial in the large, iterative in the small, delivering incremental releases over time.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Goals** | **Milestone** |
| 1. Inception | To identify the initial scope of the project, a potential architecture for your system, and to obtain initial project funding and stakeholder acceptance. | Lifecycle Objectives (LCO) |
| 2. Elaboration | To prove the architecture of the system. | Lifecycle Architecture (LCA) |
| 3. Construction | To build working software on a regular, incremental basis which meets the highest-priority needs of your project stakeholders. | Initial Operational Capability (IOC) |
| 4. Transition | To validate and deploy your system into your production environment. | Product Release (PR) |

As this exercise is a Software component (program) and not an entire Software system, the process can be modified to deliver a quality program without losing a through and comprehensive approach to development. **R.A.D. modeling** has been adopted as illustrated below:



Within Each phase of Agile UP, 7 Disciplines are performed in an iterative manner. Throughout this Exercise I will ***only*** focus on the first 4.

|  |  |
| --- | --- |
| **Discipline** | **Overview** |
| Model | The goal of this discipline is to understand the business of the organization, the problem domain being addressed by the project, and to identify a viable solution to address the problem domain. |
| Implementation | The goal of this discipline is to transform your model(s) into executable code and to perform a basic level of testing, in particular unit testing. |
| Test | The goal of this discipline is to perform an objective evaluation to ensure quality. This includes finding defects, validating that the system works as designed, and verifying that the requirements are met. |
| Deployment | The goal of this discipline is to plan for the delivery of the system and to execute the plan to make the system available to end users. |
| Configuration Management | The goal of this discipline is to manage access to your project work products. This includes not only tracking work product versions over time but also controlling and managing changes to them. |
| Project Management | The goal of this discipline is to direct the activities that takes place on the project. This includes managing risks, directing people (assigning tasks, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget. |
| Environment | The goal of this discipline is to support the rest of the effort by ensuring that the proper process, guidance (standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed. |

2. Remote Programming Exercise - Instructions

This project should be well tested and code should be checked into GitHub along with any output produced from running your program. Write your solution in the Java programming language, using only the standard libraries. Do not use any third-party libraries for natural language processing. Describe any assumptions you make in your implementation. What are the limitations of your approach? What other approaches might be possible? At the end of the assignment there should be a commit/push for each of the following features.

**1. Write a program that identifies sentence boundaries and tokenizes the text in the file “ *nlp\_data.txt* ” into words. It should correctly process all symbols, including punctuation and whitespace. Every word must fall into a sentence. Create data structures that efficiently express the data you have processed. When your program runs it should output an XML representation of your Java object model.**

**2. Modify your program from #1 to add rudimentary recognition of proper nouns (“named entities”) in the input, and print a list of recognized named entities when it runs. The list of named entities is in the file “ *NER.txt* ”. Enhance your data structures and output schema to store information about which portions of the text represent named entities.**

**3. Modify your program from #2 to use “ *nlp\_data.zip* ” as its input. Use a thread pool to parallelize the processing of the text files contained in the zip. Aggregate the results and modify the output schema accordingly.**

1. Problem Domain

Given an input of text derived from a txt file, zip file or Ad-Hoc text entry, use appropriate recognition techniques to determine which items in the text map to sentences, recognize proper nouns (as a class of named entities), and entity location / position / Aggregation within the data structure representing portions of text as compared to a given list of named entities within another text source.

Perform parallel processing of a ZIP file as an input source with multiple computational threads from a thread pool. Update the output schema to accommodate the aggregated results.

**@ Inception Phase - achieve stakeholder consensus regarding the objectives for the project.**

***Requirements Model*** –

* Use-Case Model: Researcher’s coverage List
* User Interface model
  + Input delivery via text file or Zip File.
  + Output an XML representation
* Acceptance tests
  + Attached Email

Business Rule Specification

* Do not use any third-party libraries for natural language processing

Technical requirements

* Java Programming Language with Standard Libraries – JDK 1.7.0\_79
* GitHub - Online project hosting using Git or GitBash
* Eclipse – IDE (Development Environment, Mars 4.5.0)

Assumptions: The data set is a Unilingual analysis based in English.

**Project Feasibility** – In other words, you should be able to build it, once it's deployed you should be able to run it, and it should make economic sense to do these things. My research of various modeling techniques used to perform the required tasks, resulted in 4 main Approaches. A Model fitness test is recommended as a component of Accuracy as indicated from the f-measure defined by the Conference on Natural Language Learning (CoNLL). Essentially the f score is a harmonic mean of Precision and Recall (Correct / Select).

**Main Approach –** Semi-supervised Learning - as a class of supervised learning tasks and techniques that also makes use of unlabeled data for training.

**Alternative Approaches** –

* Linguistic grammar-based techniques – regular expressions
* Statistical Modeling - require a large amount of manually annotated training data
* Machine learning - conditional random fields being a typical choice.

**Assumptions** used in semi-supervised learning –

* Smoothness assumption – yields a preference for geometrically simple decision boundaries.
* Cluster assumption – data forms discrete cluster, sharing label within cluster.
* Manifold assumption – data lie approximately on a manifold of lower dimension that the input.

**@ Elaboration Phase - prove the architecture for the system**

1. Model Solution

*Design Model –*

* Object Model
  + Java Object – entity( Entity\_id , Entity\_word, Sentence\_id, Sentence\_position, Class\_id)
  + XML Schema
* Data Model
  + Efficiently express sentence boundaries, capture tokenized words , recognition class, relative location of entity, Class Aggregation
* Test Model –
  + Regression Test Suite - A collection of test cases, the code to run them in the appropriate order
    - [acceptance tests](http://localhost:8091/tech.php?page=Deliverables#AcceptanceTests),
    - unit tests
    - system tests
  + Automated tests to increase frequency of use.
* User Interface Prototyping
  + Ad-Hoc – Copy Paste content to be analyzed
  + Single File upload
  + ZIP file upload
  + Submit query method
* System Overview Document

*Model Methods –*

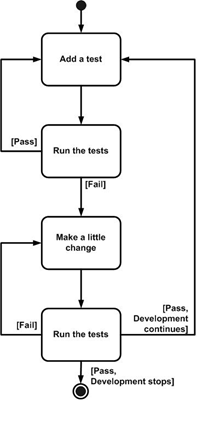
* Rudimentary Recognition – For purposes of this exercise,
* Generative models - Bayesian inference - the process of analyzing statistical models with the incorporation of prior knowledge about the model or model parameters. 
  + The idea is that, even if you cannot compute the posterior distribution analytically, you can generate a random sample from the distribution and use these random values to estimate the posterior distribution or derived statistics such as the posterior mean, median, standard deviation, etc.
* Low-density separation - place boundaries in regions where there are few data points (labeled or unlabeled).
* Graph-based methods - graph representation of the data, with a node for each labeled and unlabeled example.
* Heuristic approaches

**@ Construction Phase - develop the system to the point where it is ready for pre-production testing**

1. Proof of Concept –

Transform the models into executable code and to perform a basic level of testing. Coding and testing the software will now depend heavily on the IDE and proven methods of using the automated testing regression suite.

Test First Development – illustration



Test-driven development (TDD) = Refactoring + TFD

The next phase of SDLC would be to focus on Deployment, Transition and Product Release. These tasks encapsulate the disciplines of Configuration management, Project Management and Environment. Training would be conducted and at the same time feedback would be gathered. This Document will serve as the Project Documentation.

Researcher’s coverage List Use-Case

* **Actor** – Research Analysts
* **Preconditions** – No prior system state required before beginning the initial use case, otherwise output data should be achieved accordingly.
* **Normal course** – narrative form :
  + Research analyst maintains topic list of relevant People, Places and Events
* **Alternate courses** –
  + Ad-Hoc items are added to the Researcher Analyst’s list on demand resulting from ongoing Business Activity.
* **Exception courses** – Delivery of specific system error in area of processing such as input data format, logic fault tolerances within parsing models or unhandled exceptions.
* **Post-conditions** – XML output will contain the results of the program output.
* **Frequency of use** – A component of variable nature of the data sources.
* **Assumptions** – Any assumptions that are implicit in the definition of the use case.

Acceptance tests via Email

As a form of an acceptance test, Please allow me to ask a few questions about this exercise in order to correctly identify the requirements.

Is there a preferred Java Language edition?  I currently use Java SE 8. **Java 7 please**

Is there a Java bit size requirement, such as 32 or 64? ( i.e. compatible integration of Java Native Interface). **64 bit is fine**

Are the unit test considered to be part of the Solution? **Yes**

Can JUnit be used as the unit testing framework despite being a 3rd party framework? **jUnit is acceptable**

Are there any required Element declarations or Attributes in the output XML Schema? **No**

Is there a specification as to standard of accuracy within any modelling that may be used?  **No**

In accordance with the instruction to use a "Thread Pool", Is this a design consideration or a particular specification?

**The intent is to display the use of Multi-threading to increase performance. Use your judgement as to how to implement.**

Is there a Thread Pool Size specification? **No**

Can the Thread pool be Fixed or Dynamic?  **Either**