KDD- Service based Numerical Entity Searcher (KSNES)

Presentation 1 on Feb 24th, 2009

Naga Sowjanya Karumuri
sowji@ksu.edu

CIS 895 – MSE PROJECT
OUTLINE

- Terms
- Project Overview
  - Motivation
  - Goal
  - Purpose
  - Block Diagrams and Data Flow Diagrams
- Project Requirements
- Project Schedule
- Cost Estimation
- Software Quality Assurance Plan
- Prototype Demonstration
- Questions / Comments
TERMS

- **Knowledge Discovery in Databases (KDD)**
  - a group headed by Dr. Hsu
  - primary focus is machine learning, data mining, human-computer intelligent interaction

- **Natural Language Processing (NLP)**
  - To allow computers to process and understand human languages
  - Some areas like
    - Text Segmentation (identify word boundaries)
    - Part-of-speech tagging
    - Word sense disambiguation (words with more than one meaning)
Named Entity Recognition (NER)

- Locating and classifying atomic elements (single part of speech) in text into predefined categories such as
  - Names of Persons
  - Names of Locations
  - Names of Organizations
  - Names of Miscellaneous Entities

Example

- Dr. William H. Hsu is a Professor in Kansas State University located in Manhattan, Kansas.
- Dr. [PER William H. Hsu ] is a Professor in [ORG Kansas State University ] located in [LOC Manhattan ] , [LOC Kansas ].
TERMS (CONT.)

- **Shallow Parsing/Chunking**
  - NLP technique that attempts to look for key phrases but not to fully parse into a parse tree.
  - Output - series of words mostly nouns, verbs, preposition phrases etc.,

- **Example**
  - **Chunker:** [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only L1.8 billion]
  - **Full Parser:** (PRP)He (VBZ)reckons (DT)the (JJ)current (NN)account (NN)deficit (MD)will (VB)narrow (TO)to (RB)only (L)L (CD)1.8 (CD)billion

Parse Tree: represents syntactic structure of a string according to some formal grammar.
APPLICATION AREAS

• Textual Entailment (TE) Recognition
  • Given two fragments, whether the meaning of one text can be inferred from another text.

• Question Answering (QA) System
  • Identifies text that entails the expected answer.

Ex: During 1997, 10,000 cattle were killed because of the RVF.

• Possible inferences (TE)
  • 10,000 cattle were killed because of RVF.
  • RVF occurred during 1997.

• Possible Questions (QA)
  • How many cattle were killed during 1997 RVF outbreak?
  • When did RVF occur?
PROJECT OVERVIEW

- Motivation

  Identify Number-Unit /Date
  - Occurrence of events is naturally anchored in time within the narrative text
    - India was attacked by Pakistan in 1997.
  
  - To know the quantities of entities
    - The highest temperature recorded in the 2008 is 100 F.
PROJECT OVERVIEW (CONT.)

- **Goal**
  - To develop a system that
    - extracts *Numerical Phrases* from raw text
    - displays value – unit – unit-type
  - System is set as a service on the web server
  - User interacts through a webpage

- **Numerical Phrase: Types**
  - Number Phrase
    - 33 dollars, 100 Watts, 13 years, two miles
  - Date Phrase
PROJECT OVERVIEW (CONT.)

Purpose
- To understand the Timestamp of an event
- To understand the order of occurrence of events
- To understand the persistence of an event i.e., the time period over which the event occurred and continued
- Quantities of entities

For KDD Group
- Sample NABC (National Agricultural Bio-Security Centre) data is given to the system for testing
  - How many cattle have been effected by the virus?
  - When did the disease break out?
SYSTEM OVERVIEW

User Input

Webpage

Runs Chunker

Service based Numerical Entity Searcher

Sends Chunked Data

Output
SERVICE ORIENTED ARCHITECTURE (SOA)

- **Service**: A function that is well defined and self-contained

- **Basic SOA**
  - separates functions into distinct units or services
  - developers make them accessible over a network
  - users can combine and reuse them in the production of business applications.

- **Current System - KSNES**
  - Acts as service and should be able to be set on the server
SERVICE ORIENTED ARCHITECTURE (SOA) (CONT.)

- Service Consumer: A webpage
- Service Provider: KSNES
- Implementation:
  - User sends raw text from the webpage
  - Current system acts as a server and sends the requested information to the requesting webpage
PROJECT DATA FLOW DIAGRAM:

NUMERICAL ENTITY SEARCHER
MODULES IN THE PROJECT

- **Webpage (JSP):** For requesting and receiving information from the service.

- **POS Tagger (Java):** Stanford POS Tagger
  - Raw Text as Input
    - Joe lost 33 USD on gambling in 2000
  - Tagged Text as Output
    - Joe/NNP lost/VBD 33/CD USD/NNP on/IN gambling/NN in/IN 2000/CD

- **Numerical Phrase Extractor (Java):** To be implemented using Shallow Parsing Technique

- **Number-Unit/Date Pattern Recognizer (C++):** To be implemented based on the Numerical Quantifier developed by Benjamin Sapp, UIUC.
SOME SAMPLE UNITS AND TYPES

Example

- Dollars, INR, dinars, penny, cents  : Currency
- Foot, inches, steps, metres, miles : Length
- Kgs, lbs, kulas, karat, ton       : Weight
- Week, century, decades           : Time
- Degree, Kelvin, plank            : Temperature
- Acre, barn, hectare, square-metre : Area
- Litres, cubic-centimetres        : Volume
- Faraday, abcoulomb               : Charge
- Watt, horsepower                 : Power
- Metre-per-sq.second             : Acceleration
- Dyne, poundal, newton           : Force

And so on…

Units for work, sound, pressure, velocity, acid concentration, plane/solid angle are added in the list.
USE CASE DIAGRAM

Actor

Open the Webpage

Enter Text

View Output on the Webpage
PROJECT REQUIREMENTS

- Requirements broken into 4 sections
  - Application Requirements
  - POS Tagger Requirements
  - Numerical Phrase Extractor Requirements
  - Number-Unit/ Date Pattern Recognizer Requirements
- Each section has it’s own identifier and numbering system
- Each requirement has an associated build release noted in the Vision Plan
Demos

Number and Date Quantization Demo

Please enter your sentence(s) here:

I lost thirty-three thousand dollars on gambling in 1998.

http://l2r.cis.uiuc.edu/~cogcomp/number_demo.php
Numerical analyzer output:

I lost thirty-three thousand dollars on gambling in 1998.

chunks found: 1 number-type, 1 date-type

<table>
<thead>
<tr>
<th>chunk</th>
<th>thirty-three thousand dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>number</td>
</tr>
<tr>
<td>unit</td>
<td>dollars</td>
</tr>
<tr>
<td>value</td>
<td>33000</td>
</tr>
<tr>
<td>bound</td>
<td>=</td>
</tr>
</tbody>
</table>

| chunk          | 1998.                         |
| type           | date                          |
| format         | standard                      |
| value          | 0/0/1998                      |
| modifiers      | none                          |
| bound          | =                             |
APPLICATION REQUIREMENTS

- **ARI 1 [CR]** – The application shall provide a GUI for the user interaction
- **ARI 2** – The application shall be executable in single step
- **ARI 3 [CR]** – The application shall be started when the user enters the text into the text box in the webpage
- **ARI 4** – The application shall invoke the other modules of the project when the user clicks on the send button
- **ARI 5** – The application shall clear the text in the text box when the clear button is clicked by the user
- **ARI 6 [CR]** – The POS Tagger and the Numerical Phrase Extractor shall be platform independent while Number-Unit/Date Pattern Recognizer shall work on Linux machine
- **ARI 7 [CR]** – The user shall be able to view the chunked output on the webpage
- **ARI 8** – The user shall be able to stop the running of the application
- **ARI 9** – The user shall be able to run the application again with a new input
- **ARI 10 [CR]** – The user shall be able to enter the input of any size

CR stands for Critical Requirement
**POS TAGGER REQUIREMENTS**

This module of the project does not interact with the user but since it’s an important part of the project, it is mentioned to understand its underlying working.

- **PTRI 1 [CR]** – The POS tagger shall be given the raw input text that user enters on the webpage
- **PTRI 2 [CR]** – The POS Tagger shall be a service request from the main webpage
- **PTRI 3** – The POS Tagger shall be able to tokenize the given text
- **PTRI 4** – The POS Tagger shall be able to tag the tokens
- **PTRI 5 [CR]** – The POS Tagger shall be able to produce the tagged sentence
- **PTRI 6 [CR]** – The POS tagger shall be able to sent the tagged sentence to the next module which is the Numerical Phrase Extractor.

CR stands for Critical Requirement
NUMERICAL PHRASE EXTRACTOR REQUIREMENTS

This module of the project does not interact with the user but since it’s an important part of the project, it is mentioned to understand its underlying working.

- **NPERI 1 [CR]** – The Numerical Phrase Extractor shall be able to take the tagged sentence from the POS Tagger
- **NPERI 2** – The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the numbers and the units
- **NPERI 3** – The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the dates
- **NPERI 4 [CR]** – The Numerical Phrase Extractor shall be able to produce the filtered number-unit or the date phrase
- **NPERI 5 [CR]** – The Numerical Phrase Extractor shall be able to send the filtered phrase to the next module which is the Number-Unit/Date Pattern Recognizer

CR stands for Critical Requirement
NUMBER-UNIT/DATE PATTERN RECOGNIZER REQUIREMENTS

This module of the project does not interact with the user but since it’s an important part of the project, it is mentioned to understand its underlying working.

- **NDPRRI 1 [CR]** – The Number-Unit/Date Pattern Recognizer shall be able to take the extracted phrase from the Numerical Phrase Extractor
- **NDPRRI 2** – The Number-Unit/Date Pattern Recognizer shall be able to identify the numbers, units and unit-type if present in the phrase
- **NDPRRI 3** – The Number-Unit/Date Pattern Recognizer shall be able to identify the date, month and the year if present in the given phrase
- **NDPRRI 4 [CR]** – The Number-Unit/Date Pattern Recognizer shall be able to produce the phrase type, value, unit and unit-type
- **NDPRRI 5 [CR]** – The Number-Unit/Date Pattern Recognizer shall be able to display the phrase type, value, unit and unit-type to the user on the webpage

CR stands for Critical Requirement
PROJECT SCHEDULE

Key Dates

- Presentation 1: February 24th, 2009
  - Complete Numerical Sub-Chunker

- Presentation 2: March 24rd, 2009
  - Numerical Phrase Extractor Milestone

- Presentation 3: April 24th, 2009
  - Patch up the modules
  - Develop a GUI
  - Set them up on the server
## PROJECT SCHEDULE

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Phase One - Inception Phase</td>
<td>26 days</td>
<td>Wed 1/26/09</td>
<td>Fri 2/26/09</td>
</tr>
<tr>
<td>2</td>
<td>Documentation</td>
<td>24 days</td>
<td>Thu 1/29/09</td>
<td>Fri 2/23/09</td>
</tr>
<tr>
<td>3</td>
<td>Visio Document 1.0</td>
<td>19 days</td>
<td>Thu 1/25/09</td>
<td>Fri 2/20/09</td>
</tr>
<tr>
<td>4</td>
<td>Software Quality Assurance Plan 1.0</td>
<td>3 days</td>
<td>Thu 2/1/09</td>
<td>Mon 2/6/09</td>
</tr>
<tr>
<td>5</td>
<td>Project Plan 1.0</td>
<td>4 days</td>
<td>Sat 2/7/09</td>
<td>Wed 2/11/09</td>
</tr>
<tr>
<td>6</td>
<td>Presentation 1 Slides</td>
<td>9 days</td>
<td>Mon 2/20/09</td>
<td>Wed 2/25/09</td>
</tr>
<tr>
<td>7</td>
<td>Coding</td>
<td>17 days</td>
<td>Tue 2/2/09</td>
<td>Mon 2/23/09</td>
</tr>
<tr>
<td>8</td>
<td>Implementing Numerical Sub-Chapter</td>
<td>10 days</td>
<td>Fri 2/5/09</td>
<td>Wed 2/20/09</td>
</tr>
<tr>
<td>9</td>
<td>POS Tagger initiation</td>
<td>2 days</td>
<td>Sat 2/7/09</td>
<td>Mon 2/9/09</td>
</tr>
<tr>
<td>10</td>
<td>Preparing logic for Phrase Extractor</td>
<td>14 days</td>
<td>Mon 2/20/09</td>
<td>Wed 2/25/09</td>
</tr>
<tr>
<td>11</td>
<td>Project Prototype 1.0</td>
<td>1 day</td>
<td>Mon 2/25/09</td>
<td>Mon 2/25/09</td>
</tr>
<tr>
<td>12</td>
<td>Presentation 1</td>
<td>1 day</td>
<td>Thu 2/26/09</td>
<td>Thu 2/26/09</td>
</tr>
<tr>
<td>13</td>
<td>Phase Two - Elaboration Phase</td>
<td>25 days</td>
<td>Fri 2/27/09</td>
<td>Thu 3/26/09</td>
</tr>
<tr>
<td>14</td>
<td>Action Items</td>
<td>1 day</td>
<td>Thu 3/2/09</td>
<td>Thu 3/2/09</td>
</tr>
<tr>
<td>15</td>
<td>Documentation</td>
<td>21 days</td>
<td>Fri 3/3/09</td>
<td>Sun 3/23/09</td>
</tr>
<tr>
<td>16</td>
<td>Software Quality Assurance Plan 2.0</td>
<td>3 days</td>
<td>Fri 2/27/09</td>
<td>Sun 3/1/09</td>
</tr>
<tr>
<td>17</td>
<td>Visio Document 2.0</td>
<td>3 days</td>
<td>Mon 3/2/09</td>
<td>Wed 3/4/09</td>
</tr>
<tr>
<td>18</td>
<td>Project Plan 2.0</td>
<td>1 day</td>
<td>Mon 3/3/09</td>
<td>Mon 3/3/09</td>
</tr>
<tr>
<td>19</td>
<td>Formal Requirements Specification</td>
<td>3 days</td>
<td>Tue 3/6/09</td>
<td>Thu 3/9/09</td>
</tr>
<tr>
<td>20</td>
<td>Architecture Design 1.0</td>
<td>4 days</td>
<td>Tue 3/6/09</td>
<td>Fri 3/9/09</td>
</tr>
<tr>
<td>21</td>
<td>Test Plan 1.0</td>
<td>3 days</td>
<td>Mon 3/9/09</td>
<td>Wed 3/11/09</td>
</tr>
<tr>
<td>22</td>
<td>Inspection Checklist 1.0</td>
<td>3 days</td>
<td>Thu 3/14/09</td>
<td>Sat 3/16/09</td>
</tr>
<tr>
<td>23</td>
<td>Formal Technical Evaluation</td>
<td>5 days</td>
<td>Wed 3/22/09</td>
<td>Sun 3/29/09</td>
</tr>
<tr>
<td>24</td>
<td>Presentation 2 Slides</td>
<td>1 day</td>
<td>Mon 3/26/09</td>
<td>Mon 3/26/09</td>
</tr>
<tr>
<td>25</td>
<td>Coding</td>
<td>21 days</td>
<td>Fri 3/27/09</td>
<td>Sun 4/2/09</td>
</tr>
<tr>
<td>26</td>
<td>Connecting GUI to tagger</td>
<td>5 days</td>
<td>Fri 3/27/09</td>
<td>Fri 4/3/09</td>
</tr>
<tr>
<td>27</td>
<td>POS Tagger service culp</td>
<td>4 days</td>
<td>Sun 3/30/09</td>
<td>Wed 4/3/09</td>
</tr>
<tr>
<td>28</td>
<td>Numerical Phrase Extractor Implementation</td>
<td>14 days</td>
<td>Thu 3/31/09</td>
<td>Sat 4/2/09</td>
</tr>
<tr>
<td>29</td>
<td>Project Prototype 2.0</td>
<td>2 days</td>
<td>Sat 4/2/09</td>
<td>Sun 4/2/09</td>
</tr>
<tr>
<td>30</td>
<td>Presentation 2</td>
<td>1 day</td>
<td>Mon 4/3/09</td>
<td>Mon 4/3/09</td>
</tr>
<tr>
<td>31</td>
<td>Phase Three - Production Phase</td>
<td>26 days</td>
<td>Tue 3/31/09</td>
<td>Fri 4/24/09</td>
</tr>
<tr>
<td>32</td>
<td>Action Items</td>
<td>1 day</td>
<td>Tue 4/3/09</td>
<td>Tue 4/3/09</td>
</tr>
<tr>
<td>33</td>
<td>Documentation</td>
<td>24 days</td>
<td>Tue 3/31/09</td>
<td>Wed 4/22/09</td>
</tr>
<tr>
<td>34</td>
<td>Software Quality Assurance Plan 3.0</td>
<td>3 days</td>
<td>Wed 3/31/09</td>
<td>Fri 4/6/09</td>
</tr>
<tr>
<td>35</td>
<td>Visio Document 3.0</td>
<td>2 days</td>
<td>Fri 4/6/09</td>
<td>Mon 4/9/09</td>
</tr>
<tr>
<td>36</td>
<td>Project Plan 3.0</td>
<td>3 days</td>
<td>Tue 4/9/09</td>
<td>Thu 4/12/09</td>
</tr>
<tr>
<td>37</td>
<td>Component Design</td>
<td>2 days</td>
<td>Fri 4/9/09</td>
<td>Mon 4/12/09</td>
</tr>
<tr>
<td>38</td>
<td>Assessment Evaluation</td>
<td>4 days</td>
<td>Thu 4/11/09</td>
<td>Fri 4/13/09</td>
</tr>
<tr>
<td>39</td>
<td>User’s Manual</td>
<td>1 day</td>
<td>Sun 4/11/09</td>
<td>Sun 4/12/09</td>
</tr>
<tr>
<td>40</td>
<td>Formal Technical Inspection Letters</td>
<td>3 days</td>
<td>Mon 4/13/09</td>
<td>Wed 4/15/09</td>
</tr>
<tr>
<td>41</td>
<td>References</td>
<td>1 day</td>
<td>Thu 4/16/09</td>
<td>Thu 4/16/09</td>
</tr>
<tr>
<td>42</td>
<td>Presentation 3 Slides</td>
<td>4 days</td>
<td>Mon 4/20/09</td>
<td>Thu 4/23/09</td>
</tr>
<tr>
<td>43</td>
<td>Coding</td>
<td>20 days</td>
<td>Wed 4/25/09</td>
<td>Sat 5/1/09</td>
</tr>
<tr>
<td>44</td>
<td>Publishing all the module</td>
<td>17 days</td>
<td>Wed 4/25/09</td>
<td>Wed 5/12/09</td>
</tr>
<tr>
<td>45</td>
<td>Completing GUI</td>
<td>6 days</td>
<td>Mon 5/5/09</td>
<td>Mon 5/11/09</td>
</tr>
<tr>
<td>46</td>
<td>Setting up the service</td>
<td>4 days</td>
<td>Tue 4/29/09</td>
<td>Sat 5/1/09</td>
</tr>
<tr>
<td>47</td>
<td>Sub-Channels Output</td>
<td>4 days</td>
<td>Sun 4/29/09</td>
<td>Wed 5/2/09</td>
</tr>
<tr>
<td>48</td>
<td>Deliverables Executable</td>
<td>4 days</td>
<td>Sat 4/29/09</td>
<td>Wed 5/2/09</td>
</tr>
<tr>
<td>49</td>
<td>Testing</td>
<td>45 days</td>
<td>Fri 2/27/09</td>
<td>Tue 4/21/09</td>
</tr>
<tr>
<td>50</td>
<td>Functionality Testing</td>
<td>20 days</td>
<td>Wed 5/19/09</td>
<td>Tue 5/19/09</td>
</tr>
<tr>
<td>51</td>
<td>Testing Evaluation</td>
<td>3 days</td>
<td>Fri 5/14/09</td>
<td>Fri 5/14/09</td>
</tr>
<tr>
<td>52</td>
<td>Testing Evaluation</td>
<td>1 day</td>
<td>Fri 5/14/09</td>
<td>Fri 5/14/09</td>
</tr>
</tbody>
</table>
COST ESTIMATION MODEL

- COCOMO II: The Post-Architecture Model
  - Is used before we start developing the architecture

- COCOMO II is chosen as it considers important factors like
  - Reliability
  - Software Development Complexity
  - Database and Memory Usage

  and many other…
COST ESTIMATION FORMULAE

\[
\text{Effort} = 2.45 \times \text{EAF} \times (\text{KSLOC})^{1.09}
\]

\[
\text{Time} = 2.5 \times (\text{Effort})^{0.38}
\]

- Effort = Number of person months (PM)
- Time = Duration time in months for project
- KSLOC = Estimated number of source lines of code for the project (expressed in thousands)
- EAF = Effort Adjustment Factor
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Classification</th>
<th>Value</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Low</td>
<td>0.88</td>
<td>Project is not safety critical, and does not have to be completely reliable</td>
</tr>
<tr>
<td>DATA</td>
<td>Low</td>
<td>0.94</td>
<td>Data used by the project is only the user’s input which is relatively small in size</td>
</tr>
<tr>
<td>CPLX</td>
<td>High</td>
<td>1.02</td>
<td>Processing natural text and extracting the required features from them are complicated</td>
</tr>
<tr>
<td>TIME</td>
<td>Nominal</td>
<td>1.00</td>
<td>Response time is important but not a overly critical issue</td>
</tr>
<tr>
<td>STOR</td>
<td>Nominal</td>
<td>1.00</td>
<td>Tagging and Numerical Phrase Extraction will not require high memory usage</td>
</tr>
<tr>
<td>VIRT</td>
<td>Low</td>
<td>0.87</td>
<td>Low complexity of the hardware and software</td>
</tr>
<tr>
<td>TURN</td>
<td>Low</td>
<td>0.87</td>
<td>The turnaround time on results is low as the project is developed by a single developer</td>
</tr>
<tr>
<td>ACAP</td>
<td>High</td>
<td>0.86</td>
<td>Developer has 3+ years experience in Software Engineering</td>
</tr>
<tr>
<td>AEXP</td>
<td>Nominal</td>
<td>1.00</td>
<td>Developer has 2+ years experience in applications development</td>
</tr>
<tr>
<td>PCAP</td>
<td>Nominal</td>
<td>1.00</td>
<td>Developer has 6 month of experience in the application area</td>
</tr>
<tr>
<td>VEXP</td>
<td>Low</td>
<td>1.10</td>
<td>Developer has 6 months of experience and is learning Java Virtual Machine</td>
</tr>
<tr>
<td>LEXP</td>
<td>Low</td>
<td>1.15</td>
<td>Developer has 6 months of experience and is learning Java</td>
</tr>
<tr>
<td>TOOL</td>
<td>Nominal</td>
<td>1.00</td>
<td>Moderate experience with the tools being used</td>
</tr>
<tr>
<td>MODP</td>
<td>Very High</td>
<td>0.83</td>
<td>Developer has 4+ years experience in employing modern Software Engineering Practices</td>
</tr>
<tr>
<td>SCED</td>
<td>Nominal</td>
<td>1.00</td>
<td>Project is tightly scheduled but is a bit flexible</td>
</tr>
</tbody>
</table>
COST ESTIMATION CALCULATIONS

- KSLOC estimated at: 1.5
  - Based on other Numerical Phrase Extractor + server programming
- EAF = 0.58

- Effort = 2.21 person months
- Time = 3.38 chronological months

  - Result: Project should be able to be accomplished in 1 semester.
SOFTWARE QUALITY ASSURANCE PLAN

- References
  - Vision Plan
  - Project Plan

- Supervisory Committee
  - Dr. William H. Hsu
  - Dr. Torben Amtoft
  - Dr. Mitchell L. Neilsen

- Major Professor
  - Dr. William H. Hsu

- Developer
  - Naga Sowjanya Karumuri

- Formal Technical Inspectors
  - Snehal Monteiro
  - Svitlana Volkova
SOFTWARE QUALITY ASSURANCE PLAN
(CONT.)

- Documentation
  - A listing of the required documentation is available at:
  - Project Documentation will be available at:
    - http://people.cis.ksu.edu/~sowji/100jiMSE/
SOFTWARE QUALITY ASSURANCE PLAN (CONT.)

- Standards, Practices, Conventions & Metrics
  - Documentation – IEEE standards will be followed for all applicable documentation
  - Coding – Java and C++ coding standards
  - Metrics – COCOMO II will be used to measure project effort

- Reviews & Audits
  - Supervisory committee will review all documentation at each milestone
  - Formal Technical Inspectors will review the architecture before the second presentation
SOFTWARE QUALITY ASSURANCE PLAN (CONT.)

- **Testing**
  - Defined in Software Test Plan
    - Will be available by Presentation 2
    - Outputs are checked at each module
      - Check if the text is tagged
      - Check if the correct phrase is extracted
      - Check if the numeric phrase is divided as Value, Unit and Unit-Type
    - Outputs at the end of each module are compared manually

- **Problem Reporting**
  - Issues will be logged in a spreadsheet
  - All issues will be reported to Major Professor
SOFTWARE QUALITY ASSURANCE PLAN (CONT.)

- Tools, Technologies and Methodologies
  - Eclipse IDE – for software development
  - Java – for software development
  - GNU C++ – for software development
  - JSP – for server side coding
  - Apache Tomcat – for web server
  - Microsoft Word – for documentation development
  - Microsoft Excel – for risk and problem report tracking and time logs
  - Microsoft PowerPoint – for project presentation creation
  - Microsoft Project – for drawing the Gantt chart (project planning)
  - Microsoft Visio – for software design development
  - Microsoft Front Page – for developing the project webpage
  - Open Office – for document conversation to PDF
  - JUnit – for testing the java code
SOFTWARE QUALITY ASSURANCE PLAN (CONT.)

- Code and Media Control
  - Tortoisesvc – Subversion Control System will be used for source code control
  - Change logs will be maintained for all documents
  - Versions of documentation will be maintained on the developer’s computer

- Key Requirements
  - Accuracy of extracting the Numerical Phrase
  - Accuracy of the Number-Unit/Date Pattern Recognizer
  - GUI Quality – User friendly (send and clear buttons)
PROTOTYPE DEMONSTRATION

- Cognitive Computational Group @ UIUC
  - Shows the sample of the final output of this project
    - http://l2r.cs.uiuc.edu/~cogcomp/number_demo.php

- POS Tagger working
  - For now it works on the local machine

- Number-Unit/Date Pattern Recognizer
  - Shown its working from the console
PHASE 2 DELIVERABLES

- Vision Plan 2.0
- Project Plan 2.0
- Architectural Design Document
- Software Test Plan 1.0
- Technical Inspection List
- Presentation 2
- Prototype 2.0 Source Code
  - Partial Implementation of Numerical Phrase Extractor
TO-DO LIST

- Revise the Documents
- Revise Project Schedule
- Design the Numerical Phrase Extractor and implement partially
  - Will be implemented in Java
  - Using Pattern Recognition (regexp)
- Design the GUI