WordNet
10/25/2010

1. Module name: WordNet

2. Scope:
a. This module covers the use of a thesaurus in several information retrieval (IR) techniques: index construction (e.g., tokenization, stemming, and lemmatization), robustness to query typographical errors (e.g., the use of wildcard queries) and query refinement and expansion.

3. Learning objectives:
The user should:
a. Know the many lexical relationships that can be investigated through WordNet (i.e., all definitions in the glossary in Section 13).

b. Be capable of using the command line interface to WordNet to query for a term coupled with any of the lexical relationships from Section 3-a.

c. Be able to perform a wildcard query in WordNet.

d. Be able to provide examples of how a thesaurus can be used to perform automated query expansion as well as query refinements.

4. 5S characteristics of the module:
a. **Streams**: WordNet provides the IR system with words. The words are stored in plain text files.

b. **Structures**: WordNet is a lexical database consisting of two files for each syntactic category (noun, verb, adjective and adverb): an index file containing an alphabetized list of all the words in WordNet and a data file containing information (pointers) about synonyms and the other linguistic relationships (e.g., antonyms and homonyms).

c. **Spaces**: N/A

d. **Scenarios**: The IR system queries WordNet to check the existence of a word or to find words with certain lexical relationships.

e. **Society**: WordNet is language dependent and so reflects Society use of language. The language of the IR system requires careful consideration when utilizing a thesaurus such as WordNet.

5. Level of effort required (in-class and out-of-class time required for students):
a. The module should require 4-6 hours to complete.

i. **Out-of-Class**: Each user should expect to spend around 3-4 hours studying the module individually, including the completion of each exercise. The lexical terminology may be the biggest hurdle for most users. Careful investigation of the material in this module should be completed prior to attending class. It is imperative that users complete the exercises individually and before a group meeting.

ii. **In-Class**: Each user should spend 1-2 hours discussing the module with their teammates. Any terminology or concepts that are unclear can be clarified during this discussion. Additionally, users should take this opportunity to discuss their individual results to the exercises in Section 12.

6. Relationships with other modules (flow between modules):
a. NLTK: The WordNet module may be taught just before or after the module for the Natural Language Toolkit (NLTK). The NLTK module introduces tools in the Python programming language that can be used for collecting, tokenizing, stemming, and analyzing text. Many of these tools may be helpful during the exercises in Section 11, and the NLTK module will introduce the user to several of the basic concepts of IR that are used throughout this module.

7. Prerequisite knowledge/skills required (what the students need to know prior to beginning the module; completion optional; complete only if prerequisite knowledge/skills are *not* included in other modules):
a. The user should have already been introduced to basic IR concepts such as:

i. Tokenization

ii. Stemming

iii. Wildcard Queries

b. The user should be familiar with using a command line interface in a terminal window.

8. Introductory remedial instruction (the body of knowledge to be taught for the prerequisite knowledge/skills required; completion optional):
a. Tokenization is used to break up each unit of text into a collection of strings or tokens. Decisions on the *best* way to tokenize a document must be considered in any decent IR system, and the use of a thesaurus such as WordNet may be helpful in determining which multi-word phrases to retain as a single term.

i. **Example**: Consider the text *"The long jumper from South Carolina jumped three yards."* Whitespace may be used as the delimiter for tokenization, leading to the following terms: *the*, *long, jumper, from, south*, *carolina, jumped, three, yards.* However, since *South Carolina* may appear frequently in a document, we may wish to keep *south carolina* as a single token despite the white space.

b. Stemming algorithms may be applied after tokenization to reduce the number of tokens for a given document and to group similar terms to the same token. Multiple stemming algorithms have been developed; two well-known algorithms include the Porter stemmer and Lovins stemmer.

i. **Example**: Consider the tokens: *the*, *long, jumper, from, south carolina, jumped, three, yards*. We may choose to use a stop list to remove uninformative, common words from this list, such as "*the*" and "*from*". We may also choose to use a stemmer on each term, resulting in the following final collection of tokens: *long, jump, south carolina, three, yard*. Notice that two similar terms "*jumper"* and "*jumped"* both map to the same term "*jump*". Similarly, the *s* was dropped from the end of "*yards*".

c. Wildcard queries are often necessary to search for all terms in a document that contain a particular substring. Traditionally an asterisk (*) is used to denote the insertion of any string in a wildcard query. Thus, issuing the query "*mon*ay"* will request all terms that begin with "*mon"* and end with "*ay*".

i. **Example**: Suppose the user of an IR system wants to search a document for several terms related to the word "*legal"*, such as *legalization, legally, legality,* and *legalese*. Then the wildcard query "*legal*"* can be used to return all terms that begin with "*legal*".

d. Understanding how to use a command line interface is essential to working through this module and the exercises within. Throughout the module, all text that should appear in a terminal window is written in typewriter font as follows: `this is the terminal window`. Additionally, commands that the user can enter on the terminal window's command line will be bold **`like this`** to differentiate a command from its output in the terminal.

i. **Example**: After installing WordNet, you can verify that wordnet is installed by issuing the following command:

```
wn
usage: wn word [-hgla] [-n#] -searchtype [-searchtype...]
       wn [-l]
... (omitted the remaining output)
```

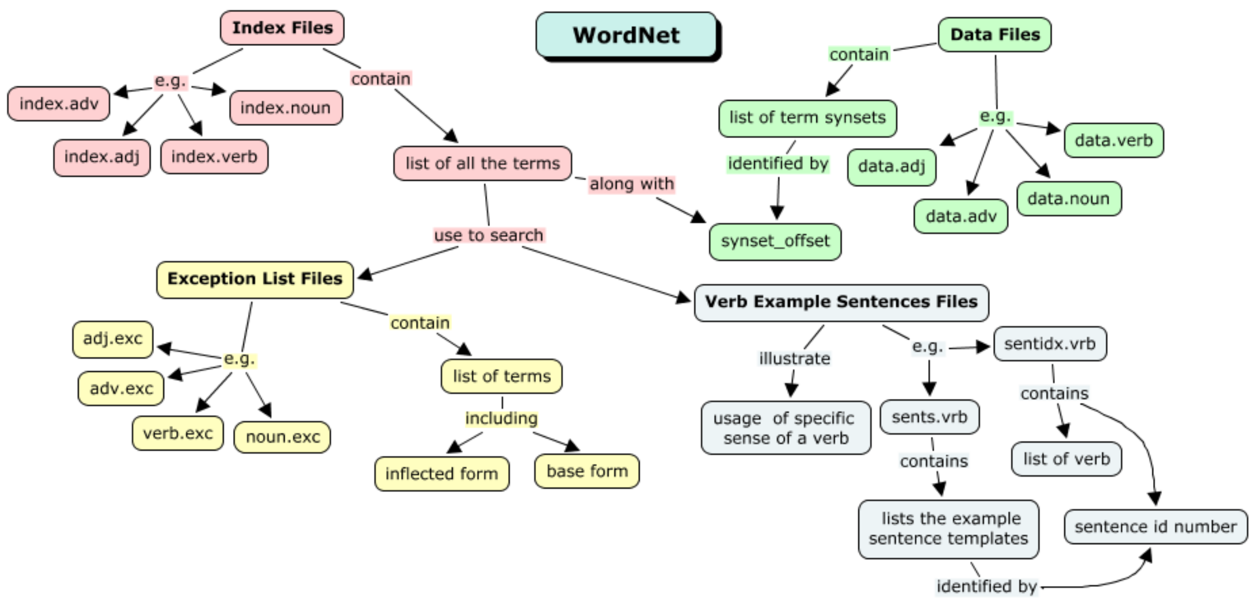
Notice that **wn** is the command that the user should issue, while the rest is simply the expected output to the terminal.

9. Body of knowledge (theory + practice; an outline that could be used as the basis for class lectures):

a. Background of WordNet

i. The effort to create WordNet was spearheaded in 1985 by George A. Miller, a professor in the Department of Psychology at Princeton University. WordNet acts as both a dictionary/thesaurus and contains sets of English synonyms, called *synsets*. Each synset is composed of a collection of words that can be interchanged with one another within text while maintaining the meaning of that text. Notice that a single term in WordNet may be associated with multiple synsets. Furthermore, WordNet provides several lexical relationships that will be discussed later in the module. The synsets and relationships among them make WordNet a powerful tool for IR concepts, such as computational linguistics, automatic text analysis, and natural language processing, among others.

The following CMap illustrates the internal structure and the data flow of WordNet. Index files contain all the terms present in WordNet and the Data files contain a list of term synsets. When the user queries a particular word, WordNet looks up the term in the appropriate index file (based on his lexical category) and follows the pointer (synset_offset) to the data file to find the term synsets. Depending on the user's search options, the term is looked up in the exception list files (to display an inflected or base form) or in the verb example sentences files (to display sentences using the verb).



b. Accessing WordNet

i. WordNet is available as a web service at the following website: http://wordnetweb.princeton.edu/perl/webwn
However, functionality is limited through the web service, and the command line interface is preferred for this module.

ii. WordNet is freely available for personal use on machines running Windows, Linux, and Mac OS at the following website: http://wordnet.princeton.edu/wordnet/download/
The user may find it helpful to download a personal copy and work through the examples while digesting the module. Note that all demonstrations of WordNet in this module were performed on a machine running Linux. As of October 15, 2010, the version for Windows is WordNet 2.1, while the version for Linux/Mac OS is WordNet 3.0. Therefore, users running the Windows version of WordNet may experience slightly different results than what is presented in this module.

iii. WordNet has been installed on an IBM cloud instance running SUSE Linux. Three pieces of information are needed to access the IBM cloud via SSH: hostname, IP address, and RSA key for the instance. Your instructor or the module author can provide you with the information needed to log into the cloud instance. If you are uncertain about how to SSH into the cloud instance using this information, please contact the instructor or authors for further instructions.

c. Using WordNet: The user is assumed to have access to a working copy of WordNet at this point. Issuing **wn** with no options will display the following help text to the user:

```
wn
usage: wn word [-hgla] [-n#] -searchtype [-searchtype...]
       wn [-l]

       -h           Display help text before search output
       -g           Display gloss
       -l           Display license and copyright notice
       -a           Display lexicographer file information
       -o           Display synset offset
       -s           Display sense numbers in synsets
       -n#          Search only sense number #

searchtype is at least one of the following:
       -ants{n|v|a|r}           Antonyms
       -hype{n|v}               Hypernyms
       -hypo{n|v}, -tree{n|v}   Hyponyms & Hyponym Tree
       -entav                   Verb Entailment
       -syns{n|v|a|r}           Synonyms (ordered by estimated frequency)
       -smemn                   Member of Holonyms
       -ssubn                   Substance of Holonyms
       -sprtn                   Part of Holonyms
       -membn                   Has Member Meronyms
       -subsn                   Has Substance Meronyms
       -partn                   Has Part Meronyms
       -meron                   All Meronyms
       -holon                   All Holonyms
       -causv                   Cause to
       -pert{a|r}               Pertainyms
       -attr{n|a}               Attributes
       -deri{n|v}               Derived Forms
       -domn{n|v|a|r}           Domain
       -domt{n|v|a|r}           Domain Terms
       -faml{n|v|a|r}           Familiarity & Polysemy Count
       -framv                   Verb Frames
       -coor{n|v}               Coordinate Terms (sisters)
       -simsv                   Synonyms (grouped by similarity of meaning)
       -hmern                   Hierarchical Meronyms
       -hholn                   Hierarchical Holonyms
       -grep{n|v|a|r}           List of Compound Words
       -over                    Overview of Senses
```

By working through an increasingly more complex collection of WordNet commands, we will cover each of the following components in the basic usage:
`wn word [-hgla] [-n#] -searchtype [-searchtype...]`

i. **`wn`** This simply calls WordNet (output above).

ii. **`word`** This represents the word for which the user wants to query in WordNet. Consider the following example for querying *football* in WordNet:

```
wn football
Information available for noun football
       -hypen             Hypernyms
       -hypon, -treen     Hyponyms & Hyponym Tree
```

```
	-synsn		Synonyms (ordered by estimated frequency)
	-partn		Has Part Meronyms
	-meron		All Meronyms
	-derin		Derived Forms
	-domtn		Domain Terms
	-famln		Familiarity & Polysemy Count
	-coorn		Coordinate Terms (sisters)
	-grepn		List of Compound Words
	-over		Overview of Senses
No information available for verb football
No information available for adj football
No information available for adv football
```
We can see that information is available for noun synsets of football, but WordNet has no verb, adjective, or adverb synsets for football.

iii. **`[-hgla]`** Any number of these four single-character options may be provided to WordNet. Consult the output shown in Section 9-c for description of each option. Options **`-h`** and **`-g`** will be the most useful for this module. **`-h`** displays help information about each searchtype given. **`-g`** displays gloss (definition and example of use, when available) for each searchtype given. Searchtypes are discussed in Section 9-c-v.

iv. **`[-n#]`** The **-n** option may be used to restrict the output from WordNet to a particular sense (meaning) of the given word.

v. **`-searchtype`** The searchtype can be any of the 28 options listed in the output in Section 9-c. Each of these options indicates a certain lexical relationship. Any WordNet terms that have the given lexical relationship are displayed to the terminal. Please consult the glossary in Section 13 for definitions of each relationship (e.g., antonym, hypernym, holonym, and meronym). Some lexical relationships will be more useful than others for the exercises in this module, but it is worth reading through and understanding each. Notice that some of the searchtype options end in **`{n|v|a|r}`**. This denotes four different options, where appending exactly one of these characters to the four-letter command indicates whether the relationship should be performed for nouns (**`n`**), verbs (**`v`**), adjectives (**`a`**), or adverbs (**`r`**). Some searchtypes do not apply to all four parts of speech, thus in some cases only a subset of these four characters may be available as valid options.

d. WordNet command examples: Next we provide several WordNet commands and the output to the terminal. A description of the command and output is given for each command.

i. The **`-over`** option provides an overview of the two synsets for the query term *football*.
```
wn football -over
Overview of noun football
The noun football has 2 senses (first 2 from tagged texts)
```

```
1. (9) football, football game -- (any of various games played with a ball (round or
oval) in which two teams try to kick or carry or propel the ball into each other's
goal)
2. (2) football -- (the inflated oblong ball used in playing American football)
```

ii. The **`-synsn`** option displays all terms in the two noun synsets of the query term *football*.

```
wn football -synsn
Synonyms/Hypernyms (Ordered by Estimated Frequency) of noun football
2 senses of football
Sense 1
football, football game
       => field game
       => contact sport
Sense 2
football
       => ball
```

iii. We can couple **`-synsn`** with the options **`-g`** and -**`n1`** to additionally display the gloss (**`-g`**) for only the first synset (**`-n1`**).

```
wn football -g -n1 -synsn
Synonyms/Hypernyms (Ordered by Estimated Frequency) of noun football
Sense 1
football, football game -- (any of various games played with a ball (round or oval) in
which two teams try to kick or carry or propel the ball into each other's goal)
       => field game -- (an outdoor game played on a field of specified dimensions)
       => contact sport -- (a sport that necessarily involves body contact between
opposing players)
```

iv. The **`-hypev`** option displays hypernyms for each of the two verb synsets of the term *cannon*.

```
wn cannon -hypev
Synonyms/Hypernyms (Ordered by Estimated Frequency) of verb cannon
2 senses of cannon
Sense 1
cannon
       => hit
           => propel, impel
               => move, displace
Sense 2
cannon
       => discharge, muster out
           => let go of, let go, release, relinquish
```

v. The **`-grep{n|v|a|r}`** option can be used to search for all WordNet terms that begin or end with the query string. Suppose we want to find all noun and verb WordNet terms that begin or end with "*heis*". This is equivalent to the wildcard query *heis* OR *heis*. Notice that this will search WordNet for all terms that begin with “*heis”* (*heis**) or end with “*heis”* (**heis*). The asterisk (*) is used to denote that any string of characters can appear at this location in the term.

```
wn heis -grepn -grepv
Grep of noun heis
heisenberg
heist

Grep of verb heis
heist
```

10. Resources (required readings for students; additional suggested readings for instructor and students):

a. Required Reading

i. George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

ii. Command Line Options: http://wordnet.princeton.edu/wordnet/man/wn.1WN.html

b. Useful Reading

i. Online Documentation: http://wordnet.princeton.edu/wordnet/documentation/

c. Additional Reference Material

i. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

11. Exercises / Learning activities:

a. (20 minutes) Get familiar with the command line interface: WordNet focuses on four major parts of speech: noun, verb, adjective, adverb. WordNet may contain synsets for all or none of these parts of speech for a particular word. Using the command line interface to WordNet, discover terms with the following properties:

i. No noun, verb, adjective or adverb synsets (i.e., the word has no synsets in WordNet).

ii. One or more verb synsets, but no noun, adjective or adverb synsets.

iii. One or more adverb synsets, but no noun, verb or adverb synsets.

iv. At least one of noun, verb, adjective and adverb synset.

b. (45 minutes) Issuing wildcard queries: You fell asleep in history class while the professor lectured on Charlemagne. You have a quiz on Charlemagne the next day but cannot recall the exact spelling of his name. Searching WordNet for *Charlemane*, *Charlemayne* or *Sharlemagne* will likely be unhelpful.

i. Suppose you know the Roman emperor's name begins with *charlem*. What command could you use in WordNet to find the proper spelling of the emperor's name using a wildcard query? What list of words are returned by your chosen command?

ii. Similarly, suppose you knew his name ended with "*magne*". What command could you use in WordNet to find the proper spelling of the emperor's name using a wildcard query? What list of words are returned by your chosen command?

iii. Is one of these wildcard queries more useful than the other? Why or why not?

iv. How could you utilize the tools used in parts i and ii to perform automated query expansion? When should such potential query expansions be suggested to the user?
(Hint: count the number of words returned from multiple wildcard queries.)

c. (45 minutes) Query expansion/refinement: Consider the following two queries:
"volcanic mountains in indonesia"
"sheep farming in new zealand"
i. Manually tokenize each query. How did you tokenize each query and are there alternative tokenizations?

ii. Stem the tokens in each query. You can do this manually (e.g., "*mountains"* can be stemmed to "*mountain"*), or you can use a stemming algorithm from NLTK, if that module has already been studied.

iii. Use the **`-grep{n|v|a|r}`** option in WordNet to find additional words containing each token as a substring.

iv. Use the **`-hype{n|v}`** option in WordNet to find several hypernyms of each token.

v. Use the **`-syns{n|v|a|r}`** option in WordNet to find additional terms that are synonymous to each token.

vi. Use the additional terms from iii-v to form 3 expanded versions of each query. What order of the use of WordNet commands (substring, hypernyms, and synsets) gives the *best* results and why?

12. Evaluation of learning objective achievement (graded exercises or assignments):
a. Exercise 12-a is designed to familiarize the user with the command line interface to WordNet.

i. **Members**: This exercise is designed as an individual exercise. As with many database tools, the best way to get familiar with the data is through "hands on" experience. If an individual is unfamiliar with using a command line, pairs may be appropriate, but each individual should execute the series of commands necessary to complete the exercise.

ii. **Goals**: After completing the exercise, each user should be comfortable with using the command line to query WordNet for a specific term. Additionally, they should be comfortable analyzing the number and type of synsets associated with that term for the available parts of speech.

b. Exercise 12-b is designed to teach the user about wildcard queries and how certain tools in WordNet can be used to execute particular types of wildcard queries.

i. **Members**: This exercise can be completed in a group 2 or 3 members. A small group is ideal, since they can discuss the available search options in WordNet and reach a consensus on the best for this task.

ii. **Goals**: The group should be able to identify **`-grep{n|v|a|r}`** as the best option to use in WordNet to perform a wildcard query. The user should recognize that the first wildcard query for "*charlem"* will return only a single result, while the second wildcard query for "*magne"* will return ~70 results. In part iii, the user should be able to explain why (or why not) one wildcard query is more useful than the other when trying to search for the term Charlemagne. Part iv requires the most creative thinking. There may be many solutions to this question, but a sample solution is the following:

"As a user types a query word, we can perform a wildcard query based on the substring that has been entered thus far. For example, while searching for "*Charlemagne"*, we issue wildcard queries for *c*, *ch*, *cha*, *char*, and so on. Count the number of terms for each of these queries. When the number of terms is large (e.g., for the wildcard query *ch*) we might decide not to suggest a potential query expansion. However, when the number of terms for a particular wildcard query is reasonably low we may suggest query expansions. For example, **`wn charle -grepn`** returns ~90 results while **`wn charlem -grepn`** returns only a single result. Thus, after the user has typed "*charlem"*, we can suggest "*charlemagne"* as a query expansion."

A more in depth answer will recognize that many of the results returned by the wildcard query for *charle* are dictionary terms that begin with Charles. Restricting the output from this query to just the first word in each dictionary item will return ~15 terms (instead of ~90 that we saw before). Thus, we may be able to perform query expansion at this stage, since 15 terms seems manageable for query suggestion.

c. Exercise 12-c encourages the user to try a wide variety of WordNet tools for performing query refinement.

i. **Members**: Parts i-vi of this exercise should first be completed individually. Users should then form groups of 2 or 3 and confer on the best collective solution.

ii. **Goals**: Individual completion of each part before meeting is essential, because the solutions will vary wildly for each member. Strictly speaking, there is no right or wrong answer for this exercise, as many valid avenues for query refinement may lead each individual to different (and sometimes conflicting) results. This provides a scaffold for the group to subsequently discuss the merits and pitfalls of each member's solution.

d. The exercises are designed as a combination of individual hands-on activities that can be completed outside of class and group activities, where group members confer to produce a team solution. Each exercise becomes increasingly more open-ended, and the later exercises will require the most collaboration among teammates.

13. Glossary:
a. Definitions
base form - the form of a word after **stemming**; inflectional forms are created from the base form

collocation - in WordNet, a term that contains two or more words (separated by spaces or hyphens)

coordinate terms - noun or verb WordNet terms that share a **hypernym**

direct antonyms - loosely, a pair of antonyms is two terms that are opposite; WordNet defines direct antonyms as "a pair of words between which there is an associative bond resulting from their frequent co-occurrence"

domain - a synset classification; can be one of CATEGORY, REGION, or USAGE

entailment - a verb X entails Y if X cannot be done unless Y is, or has been, done

group - verb **synsets** that are manually combined based on similar meaning

gloss - a **synset** property that includes a definition and example sentences, when available

holonym - Y is a holonym of X if X is a part of Y; compare to **meronym**

hypernym - Y is a hypernym of X if X is a (kind of) Y; compare to **hyponym**

hyponym - X is a hyponym of Y if X is a (kind of) Y; compare to **hypernym**

lemma - usually the **base form** of a WordNet term; the form of a term used in the WordNet index

meronym - X is a meronym of Y if X is a part of Y; compare to **holonym**

monosemous - having only one **sense** for a WordNet term; compare to **polysemous**

part of speech - WordNet recognizes four parts of speech: noun, verb, adjective, adverb

polysemous - having more than one **sense** for a WordNet term; compare to **monosemous**

polysemy count - the number of **senses** for a particular WordNet term

sense - a specific meaning of a WordNet term; each sense of a term defines a different **synset**

stemming - removing the end of a word, reducing its inflectional form to some **base form**

subordinate - see **hyponym**

superordinate - see **hypernym**

synset - a collection of WordNet terms that can be interchanged without changing the meaning of the text in which they appear; a set of **synonyms**

token - the basic unit of a dictionary and the result of **tokenizing** a document

tokenization - breaking a single string into smaller substrings, called **tokens**

troponym - a verb X is a troponym of Y if “to X” is “to Y” in some manner

b. The following table summarizes some of the lexical relationships used throughout this module. Since these relationships occur between pairs of words, the relationship is described generically, using “X” and “Y”. The vocabulary term that describes each piece of the relationship is provided under the appropriate column (“X” or “Y”). For example, consider the terms “*bumper*” and “*car*”. Notice that the first relationship (for meronym and holonym) holds if we let X=“*bumper*” and Y=“*car*”. Thus “*bumper*” is a meronym of “*car*” and “*car*” is a holonym of “*bumper*”. You can verify this by issuing the following command in a terminal window: “`wn bumper -holon`”. This command will return all noun holonyms of “*bumper*”, including the term “*car*”.

X	**Y**	**Relationship**
meronym	holonym	X is a part of Y.
hyponym	hypernym	X is a (kind of) Y. (Same as **subordinate** and **superordinate**.)
subordinate	superordinate	X is a (kind of) Y. (Same as **hyponym** and **hypernym**.)
entailment		A verb X entails Y if X cannot be done unless Y is, or has been done.
troponym		A verb X is a troponym of Y if “to X” is “to Y” in some manner.

14. Additional useful links:

15. Contributors:
a. Authors: Eric Fouh, Christopher Poirel
b. Evaluators: Edward A. Fox