Recovering Traceability Links in Requirements Documents

Zeheng Li[1], Mingrui Chen[1], LiGuo Huang[1], and Vincent Ng[2]

[1]Southern Methodist University [2]The University of Texas at Dallas

Task

- Given a set of high-level requirements and a set of low-level requirements, recover the **traceability links** between them
- Two requirements should be linked if one is a refinement of the other

Example

High-Level Requirements

HR01
The underlined character in each menu selection shall be a shortcut key. When control and the shortcut key are pressed, the menu selection should be loaded.

HR02
The system shall have an address book available to store contacts.

HR03
The system shall have a help system that offers tips and explanation for each screen and each item on the screens upon demand.

……

Low-Level Requirements

UC01

Use case name:	store a contact's information
Summary:	the address book should store a contact's name, email, address and phone number
Description:	1. enter "pine" command in terminal 2. either enter "a" or use arrows to make "address book" line highlighted and enter "enter" 3. enter "@" 4. enter nickname, fullname, fcc, comment and addresses. may leave some fields blank 5. press ctrl+x to save the entry

UC02

Use case name:	access help system
Summary:	user accesses help system
Description:	user presses help key

- Requirements traceability is many-to-many mapping:
  - A high-level requirement can be refined by multiple low-level requirements
  - A low-level requirement can refine multiple high-level requirements
- HR01 is refined by UC01
  - UC01 specifies the shortcut key for saving an entry in the address book
- HR02 is refined by UC01
  - UC01 specifies how to store contacts in the address book
- HR03 is refined by UC02
  - Both of them are concerned with the help system

Why is it important for Software Engineering?

- Software system development is guided by the evolution and refinement of requirements
- Requirements specifications are refined with additional design details and implementation information as the development life cycle progresses

Why is it challenging for NLP?

- Abundant information irrelevant to the link establishment
- Information irrelevant to the establishment of one link could be relevant to the establishment of another link involving the same requirement

Datasets

- Pine - Email system of University of Washington
- WorldVistA – Health information system

Datasets	Pine	WorldVistA
▪ # of high-level requirements	49	29
▪ # of low-level requirements	51	317
▪ Avg. # of words per high-level requirement	17	18
▪ Avg. # of words per low-level requirement	148	26
▪ Avg. # of links per high-level requirement	5.1	13.6
▪ Avg. # of links per low-level requirement	4.9	1.2
▪ # of pairs that have links	250	394
▪ # of pairs that do not have links	2249	8799

Baseline Systems

I. **Unsupervised baselines**

- Link two documents if their *Cosine similarity* exceeds a certain threshold
- Employ two ways to represent a document
  - as a vector of unigrams
    - Feature values are the *tf-idf* values
  - as a vector of *n* topics induced by an LDA model
    - Feature values are the probabilities the document belongs to the topics
    - n = 10, 20, … , 50 (Pine) and 50, 60,… , 100 (WorldVistA)
    - n is tuned on test data (thus giving an unfair advantage to these baselines)

II. **Supervised baseline**

- Linking decisions made by a binary classifier trained using LibSVM
  - Create instances by pairing each high-level requirement with each low-level requirement
  - Positive if the two requirements should be linked, and negative otherwise
  - Two types of binary-valued features:
    - Word pairs: a pair of words (w_i, w_j) from the high- and low-level documents respectively, indicating their presence in these documents
    - LDA-induced topic pairs: topic pair (t_i, t_j) whose value is 1 if t_i and t_j are the most probable topics for the high- and low-level requirements
  - C (the regularization parameter) is tuned on development data

Knowledge-rich Approach

Goal: Improve supervised baseline using two types of **human-supplied** knowledge

I. **Noun and verb clusters**

- Two ways to create noun and verb clusters
  - **Manually**
    - First define domain-relevant noun and verb categories, then populate them
    - Pine: 8 noun clusters and 10 verb clusters
    - WordVistA: 31 noun clusters and 14 verb clusters
    - A time-consuming process
  - **Automatically** (using single-link agglomerative clustering)
    - Each noun (verb) is represented using the verbs (nouns) it co-occurs with
      - We only cluster nouns/verbs in the training data that (1) have at least three characters, and (2) appear in only high-level or only low-level documents
    - Each noun/verb is initially in its own cluster
    - In each iteration, it merges the two most similar clusters and stops when the desired number of clusters is reached
    - Number of clusters: 10, 15, 20 (Pine) and 10, 20, 30, 40, 50 (WorldVistA)
- Use the manual/induced clusters to Create five additional types of features to be tuned jointly with C value on development data
  - **Verb pairs: pairs** of verbs collected from high- and low-level requirements
  - **Verb group pairs:** replace verbs in the verb pairs with their cluster ids
  - **Noun pairs:** pairs of nouns collected from high- and low-level requirements
  - **Noun group pairs:** replace nouns in the noun pair with their cluster ids
  - **Dependency pairs:** created by pairing each noun-verb pair found in high-level requirement with each noun-verb pair found in low-level requirement
- Cluster-based features can provide better generalization than word-based features

II. **Annotator rationales**

- Rationales are words/phrases in a training document that are considered relevant to the classification task at hand by the human annotator
- For each link in the training set, we asked the annotator to identify words/phrases from the associated requirements that are relevant to establishing the link
- Rationales are used to create two types of additional training instances
  - One pseudo positive instance is created from each positive training instance
    - Created by first removing the rationales from the two requirements
  - Three pseudo negative instances are created from each negative training instance
    - The first is created by removing only rationales from high-level document
    - The second is created by removing only rationales from low-level document
    - The third is created by removing rationales from both requirements
- Potentially allow the learner to focus on learning from the relevant phrases
- Example

  A1: The system shall have an address book available to store contacts.

  - Terms in red are rationales that are helpful for recovering the link
  - Retain terms in red results in pseudo positive instance
  - Retain terms in blue results in pseudo negative instance

Evaluation

- **Evaluation metrics**
  - Recall: percentage of recovered links in the gold standard
  - Precision: percentage of correctly recovered links
  - F-score: unweighted harmonic mean of recall and precision
- **Results**

System	Pine						WorldVistA					
	No Pseudo			Pseudo pos+neg			No Pseudo			Pseudo pos+neg		
	R	P	F	R	P	F	R	P	F	R	P	F
Tf-idf Baseline	73.6	43.3	54.5	--	--	--	60.4	37.8	46.5	--	--	--
LDA Baseline	30.4	39.2	34.2	--	--	--	25.9	10.6	15.1	--	--	--
Supervised baseline	50.4	67.0	57.5	53.9	73.8	62.3	52.5	79.9	63.3	55.9	80.6	66.0
+ manual clusters	54.4	73.9	62.6	57.6	77.0	65.9	52.5	82.8	64.2	57.1	83.0	67.6
+ induced clusters	53.6	72.8	61.7	55.2	75.0	63.6	52.8	83.2	64.6	57.1	82.1	67.4

- **Discussion**
  - When pseudo-instances are not used,
    - the Tf-idf baseline significantly outperforms the LDA baseline
    - the supervised baseline significantly outperforms the two unsupervised baselines
    - adding cluster-based features significantly improve the results of the supervised baseline
  - When pseudo-instances are used,
    - adding cluster-based features significantly improve the results of the supervised baseline
    - results are significantly better than when no pseudo-instances are used
  - Relative error reductions of 11.1-19.7% compared to the tf-idf baseline