On the Use of Word Embeddings for Identifying Domain Specific Ambiguities in Requirements

Siba Mishra and Arpit Sharma

Department of Electrical Engineering and Computer Science
Indian Institute of Science Education and Research, Bhopal, India

September 24, 2019
Jeju Island, South Korea
1 Motivation

2 Preliminaries

3 Our Approach

4 Results & Findings

5 Related Work

6 Conclusions & Future Work
Software Requirements

Requirements

- specify what a software is supposed to do
- serve as a legal agreement between the client and software development organization
- influence subsequent steps in software development
- provide a basis for testing
- are usually written in common natural language (NL)
Motivation

Ambiguous Software Requirements

Ambiguity

- means that a single reader can interpret the requirement in more than one way
- means multiple readers come to different interpretations
- is one of the major cause of poor quality requirements
- may lead to time and cost overrun (worst case - project failure)
Motivation

Domain Specific Ambiguity

- stakeholders with different technical backgrounds and domain expertise
- typical computer science (CS) terms may be interpreted differently by stakeholders (with no CS background)

Examples

- Platform (CS $\neq$ Petroleum)
- Tree (CS $\neq$ Environment)
- Cell (CS $\neq$ Biomedical)
- Operation (CS $\neq$ Military)
- State (CS $\neq$ Civil)
Motivation

Domain Specific Ambiguity

- stakeholders with different technical backgrounds and domain expertise
- typical computer science (CS) terms may be interpreted differently by stakeholders (with no CS background)

Examples

- Platform (CS ≠ Petroleum)
- Tree (CS ≠ Environment)
- Cell (CS ≠ Biomedical)
- Operation (CS ≠ Military)
- State (CS ≠ Civil)

Goal: Detect domain specific ambiguous CS words
Table of Contents

1 Motivation
2 Preliminaries
3 Our Approach
4 Results & Findings
5 Related Work
6 Conclusions & Future Work
Word Embeddings

- a powerful approach for analyzing language
- widely used in information retrieval and text mining
- dense representation of words (numeric vectors)
- capable of capturing the context of a word
- identifying semantically similar words, i.e., cosine similarity
- examples - GloVe (Stanford), Word2vec (Google), fastText (Facebook)
Word Embeddings

- a powerful approach for analyzing language
- widely used in information retrieval and text mining
- dense representation of words (numeric vectors)
- capable of capturing the context of a word
- identifying semantically similar words, i.e., cosine similarity
- examples - GloVe (Stanford), Word2vec (Google), fastText (Facebook)

Word2Vec
Our Approach

Table of Contents

1. Motivation
2. Preliminaries
3. Our Approach
4. Results & Findings
5. Related Work
6. Conclusions & Future Work
Our Approach

NLP Pipeline

- CS Corpora ($C_{cs}$)
- Subdomain Corpora ($C_{sd}$)

1. Tokenization
2. Punctuation Removal
3. Lowering of words
4. Stop word removal
5. PoS Tagging (Nouns)
6. Lemmatization
Our Approach

Descriptive Statistics

<table>
<thead>
<tr>
<th>Category Name</th>
<th>Pages</th>
<th>Total Sentences</th>
<th>Total Words</th>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science (CS)</td>
<td>9021</td>
<td>2,46,359</td>
<td>18,37,492</td>
<td>18,192</td>
</tr>
<tr>
<td>Building Engineering (BUE)</td>
<td>9002</td>
<td>3,52,005</td>
<td>25,77,515</td>
<td>23,538</td>
</tr>
<tr>
<td>Mechanical Engineering (MCEE)</td>
<td>7587</td>
<td>3,31,746</td>
<td>24,78,977</td>
<td>20,463</td>
</tr>
<tr>
<td>Electronic Engineering (ELCE)</td>
<td>7147</td>
<td>2,47,649</td>
<td>18,78,728</td>
<td>18,451</td>
</tr>
<tr>
<td>Civil Engineering (CIVE)</td>
<td>7071</td>
<td>2,83,337</td>
<td>21,42,500</td>
<td>20,427</td>
</tr>
<tr>
<td>Aerospace Engineering (AE)</td>
<td>4661</td>
<td>1,61,867</td>
<td>13,13,054</td>
<td>14,524</td>
</tr>
<tr>
<td>Chemical Engineering (CHEE)</td>
<td>4442</td>
<td>2,03,637</td>
<td>15,37,857</td>
<td>15,339</td>
</tr>
<tr>
<td>Environmental Engineering (ENVE)</td>
<td>2626</td>
<td>1,16,685</td>
<td>8,72,305</td>
<td>10,924</td>
</tr>
<tr>
<td>Marine Engineering (MAEE)</td>
<td>1369</td>
<td>31,712</td>
<td>2,23,956</td>
<td>4,880</td>
</tr>
<tr>
<td>Industrial Engineering (INEE)</td>
<td>1060</td>
<td>42,751</td>
<td>3,41,308</td>
<td>5,845</td>
</tr>
<tr>
<td>Military Engineering (MLEE)</td>
<td>932</td>
<td>32,068</td>
<td>2,42,944</td>
<td>5,027</td>
</tr>
<tr>
<td>Biomedical Engineering (BIEE)</td>
<td>924</td>
<td>52,599</td>
<td>3,87,492</td>
<td>8,214</td>
</tr>
<tr>
<td>Petroleum Engineering (PTEE)</td>
<td>419</td>
<td>15,148</td>
<td>1,21,614</td>
<td>2,965</td>
</tr>
<tr>
<td>Ceramic Engineering (CERE)</td>
<td>318</td>
<td>12,465</td>
<td>83,705</td>
<td>2,581</td>
</tr>
</tbody>
</table>
Table of Contents

1 Motivation
2 Preliminaries
3 Our Approach
4 Results & Findings
5 Related Work
6 Conclusions & Future Work
## Results & Findings

### Large-sized Subdomains

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (CIVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>0.49546</td>
<td>class, algorithm, model, automaton, domain</td>
<td>land, link, highway, survey, government</td>
</tr>
<tr>
<td>source</td>
<td>0.47845</td>
<td>library, tool, application, specification</td>
<td>groundwater, recovery, growth, cycle, consumption, storage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>0.59611</td>
<td>domain, set, regression, solution, element, number, property</td>
<td>mission, launch, spacecraft, shuttle, traffic, safety, satellite</td>
</tr>
<tr>
<td>system</td>
<td>0.16622</td>
<td>software, data, process, application, program</td>
<td>rocket, radio, vehicle, navigation, radar, power</td>
</tr>
</tbody>
</table>
## Results & Findings

### Large-sized Subdomains

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (CHEE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>product</td>
<td>0.57344</td>
<td>source, code, cache, mode, requirement</td>
<td>steam, soil, ammonia, combustion, methane, compound</td>
</tr>
<tr>
<td>process</td>
<td>0.52690</td>
<td>command, code, layer, requirement, specification, memory, storage</td>
<td>hydrogen, carbon, combustion, water, emission, oxygen</td>
</tr>
<tr>
<td>environment</td>
<td>0.50088</td>
<td>driver, share, encryption, resource, database</td>
<td>biodiesel, coal, pollution, impact, waste, treatment</td>
</tr>
</tbody>
</table>

### Example Sentences

- **state (CS)**: The state at which the automaton stops is called the final state.

- **state (CIVE)**: In 1872, Alexey Von Schmidt undertook the survey of the state line.
### Results & Findings

#### Medium-sized Subdomains

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (MLEE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine</td>
<td>0.65724</td>
<td>process, analysis, code, computation, data</td>
<td>defense, casualty, explosive, explosion, ammunition</td>
</tr>
<tr>
<td>operation</td>
<td>0.37621</td>
<td>object, block, integration, procedure, query</td>
<td>combat, hill, infantry, battle, attack</td>
</tr>
<tr>
<td>structure</td>
<td>0.56361</td>
<td>class, object, method, recursion, regression, procedure</td>
<td>tank, port, dock, yacht, plant, coast</td>
</tr>
</tbody>
</table>
Results & Findings

Medium-sized Subdomains

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (ENVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td>0.21638</td>
<td>heap, queue, insertion, sort, hash, merge, algorithm</td>
<td>hydrogen, reaction, biogas, reserve</td>
</tr>
</tbody>
</table>

Example Sentences

- tree (CS) : A left-leaning red–black (LLRB) tree is a type of self-balancing binary search tree
- tree (ENVE) : Van Mahotsav is an annual pan-Indian tree planting festival
## Small-sized Subdomains

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (PTEE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform</td>
<td>0.60037</td>
<td>editor, email, desktop, apple, interface, sun, gui, firewall</td>
<td>equipment, sea, site, lift, construction, level</td>
</tr>
<tr>
<td>tool</td>
<td>0.55951</td>
<td>application, database, protocol, source, web, cloud, library</td>
<td>injection, hole, drill, perforation, valve, pump</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Words</th>
<th>Similarity Score</th>
<th>Most Similar Words (CS)</th>
<th>Most Similar Words (CERE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>0.47303</td>
<td>tool, user, suite, platform, microsoft</td>
<td>water, cement, bone, steel, insulator, chemical, powder</td>
</tr>
</tbody>
</table>

### Example Sentences

- **platform (CS)**: HoneyC is a platform independent open source framework written in Ruby

- **platform (PTEE)**: The first tower emerged in the early 1980s with the installation of Exxon’s Lena oil platform
# Table of Contents

1. Motivation
2. Preliminaries
3. Our Approach
4. Results & Findings
5. Related Work
6. Conclusions & Future Work
Related Work


Table of Contents

1 Motivation
2 Preliminaries
3 Our Approach
4 Results & Findings
5 Related Work
6 Conclusions & Future Work
Conclusions

- demonstrated the applicability of word2vec algorithm for detecting domain specific ambiguity
- demonstrated its applicability in both small and large software projects
- similarity threshold to detect ambiguous words

Future Work

- investigate applicability for large scale requirements specification
- detect similarity between natural language requirements in software product lines
- compare word2vec with other word embedding techniques, e.g., GloVe, fastText etc
Thank You