Inferring semantically related words from software context

Jinqiu Yang, **Lin Tan**
University of Waterloo

Motivation

I need to find all functions that disable interrupts in the Linux kernel.
Hmmm, so I search for "**disable*interrupt**".

MISSING:

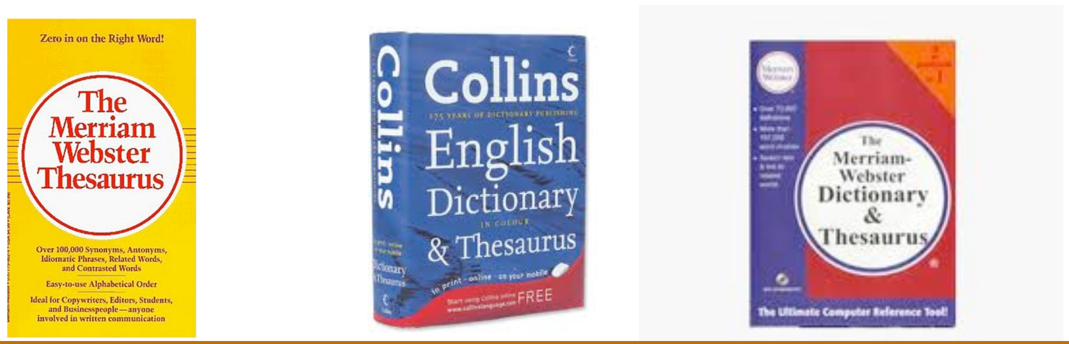
disable_irq(...), mask_irq(...)

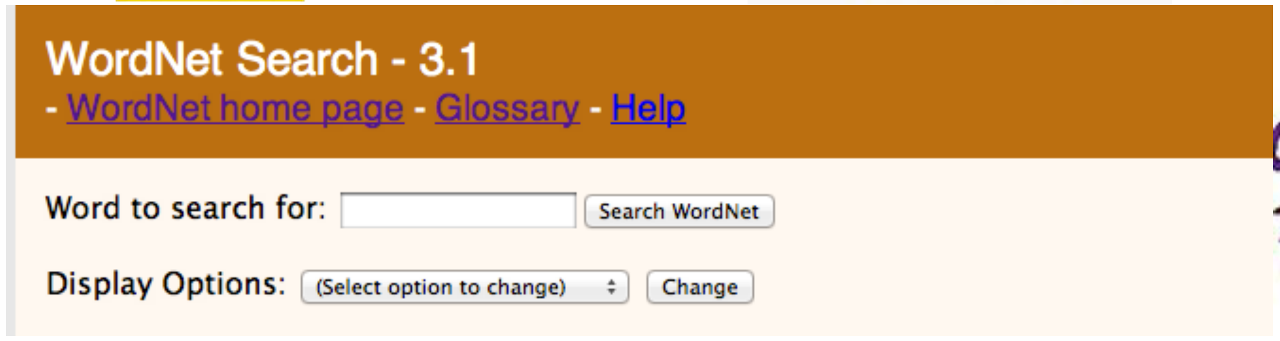
New Search Queries:

"disable*irq", "mask*irq"

BUT how am I supposed to know**???**

How to Find Synonyms or Related Words?

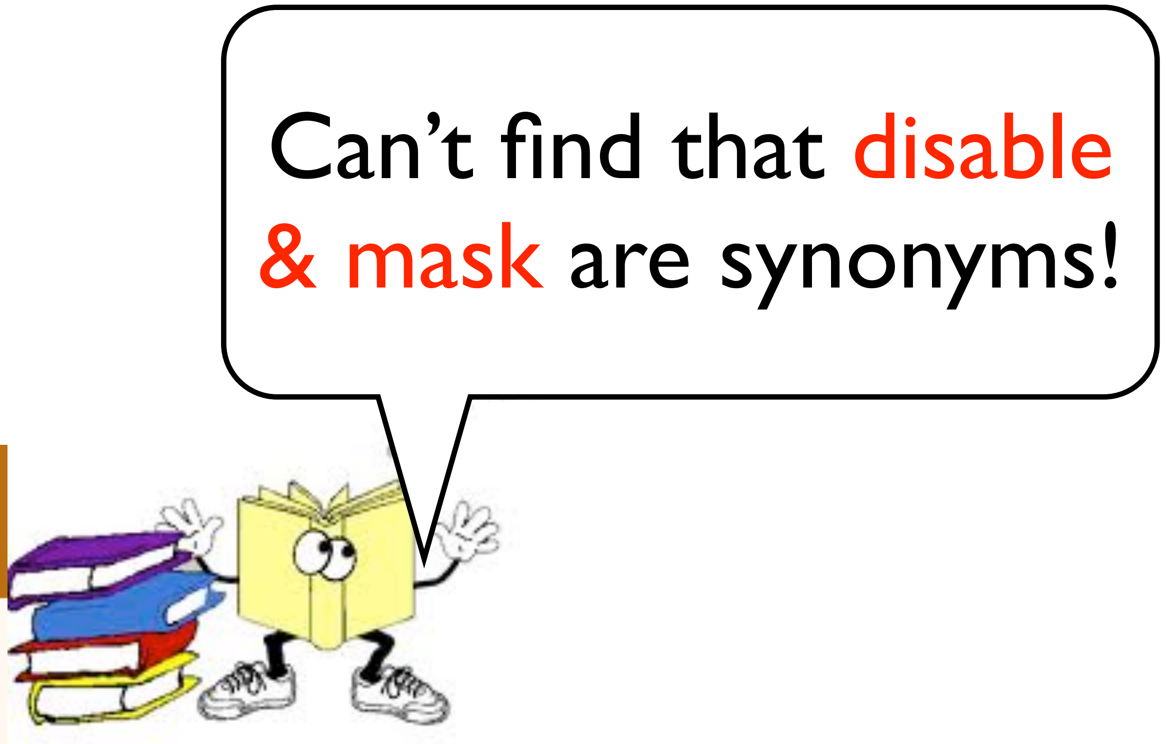

The Merriam Webster Thesaurus
Collins English Dictionary & Thesaurus
The Merriam-Webster Dictionary & Thesaurus



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: Search WordNet

Display Options: (Select option to change) Change



Guess on my own

Ask developers

Our Approach: Leveraging Context

- Comments:

"Disable all ***interrupt*** *sources"*

↕

"Disable all ***irq*** *sources"*

- Identifiers:

void ***mask****_all_interrupts()*

↕

void ***disable****_all_interrupts()*

Real comments and identifiers from the Linux kernel

- We call a pair of such **semantically related words** an **rPair**.

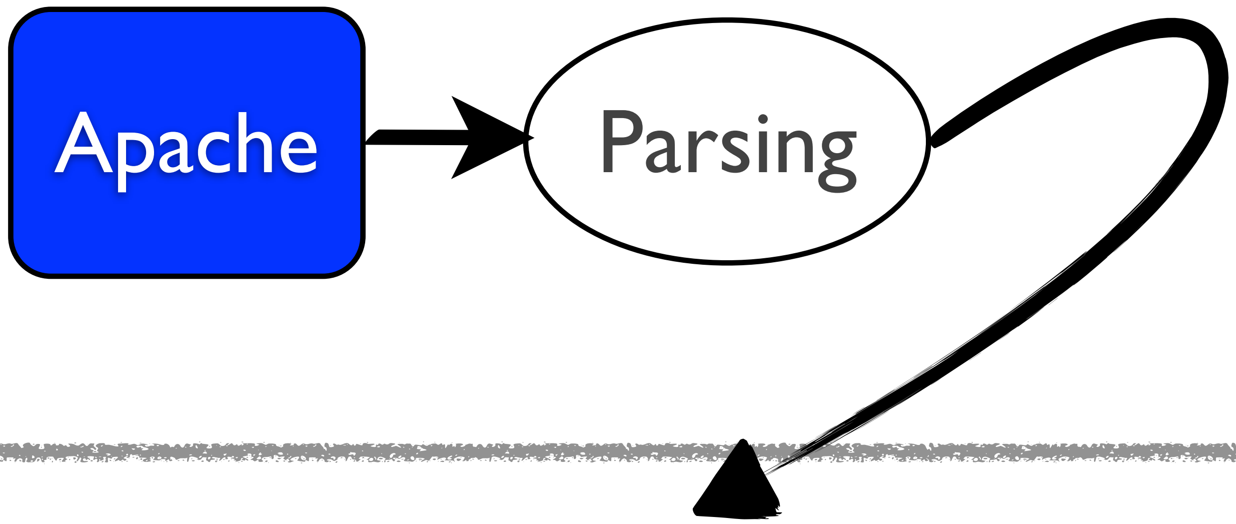
Contributions

- A general context-based approach to automatically infer semantically related words from software context
  - Has a reasonable accuracy in 7 large code bases written in C and Java.
  - Is more helpful to code search than the state of art.

Outline

- Motivation, Intuition and Contributions
- Our Approach
  - A Running Example: Parsing, Clustering, Extracting, Refining
- Evaluation Methods & Results
- Related Work
- Conclusion

A Running Example



maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

Real comments from Apache HTTPD Server

Extracting rPairs

an iovec to store the trailer sent after the file
an iovec to store the headers sent before the file

SimilarityMeasure = *8/10*

the compiled max threads
min of spare threads

SimilarityMeasure = *1/4*

$$SimilarityMeasure = \frac{\text{Number of Common Words in the Two Sequences}}{\text{Total Number of Words in the Shorter Sequence}}$$

$\boxed{\text{threshold} = 0.7}$

You can find how different thresholds affect our results in our paper.

Running Out of Time

- Pairwise comparisons of a large number of sequences is expensive.
- 519,168 unique comments in the Linux kernel → over 100 billion comparisons

Clustering

add

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

daemons

data

iovec

Clustering

add

maybe add a higher-level description

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

daemons

data

iovec

Clustering

add

maybe add a higher-level description

return err maybe add a higher-level desc

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

daemons

min of spare daemons

the compiled max daemons

data

data in the appropriate order

data in the wrong order

if a user manually create a data file

iovec

an iovec to store the headers sent before the file

an iovec to store the trailer sent after the file

The Speedup After Clustering

- Pairwise comparisons of a large number of sequences is expensive.
- <u>519,168</u> unique comments in the Linux kernel → over 100 billion comparisons.

- Clustering speeds up the process for the Linux kernel by almost 100 times.

Refining rPairs

- Filtering:
  - Using stemming to **remove** rPairs that consists of words with the same root, e.g., (called, call).
- Normalization:
  - (threads, daemons) $\longrightarrow$ (thread, daemon).
  - (called, invoked) $\longrightarrow$ (call, invoke)

Outline

- Motivation, Intuition and Contributions
- Our Approach
    - A Running Example: Parsing, Clustering, Extracting, Refining
- Evaluation Methods & Results
- Related Work
- Conclusion

Evaluation Methods

- Extraction Accuracy
  - **7** large code bases, in Java & C, from Comment-Comment, Code-Code, Comment-Code
- Search-Related Evaluation
  - Comparison with SWUM [Hill Phd Thesis] in Code-Code

Comment-Comment Accuracy Results

Software	rPairs	Accuracy	Not in Webster or WordNet
Linux	108,571	47%	76.6%
HTTPD	1,428	47%	93.6%
Collections	469	74%	97.3%
iReport	878	84%	95.2%
jBidWatcher	111	64%	98.4%
javaHMO	144	56%	91.1%
jajuk	203	69%	94.2%
Total/Average	111,804	63%	91.7%

We randomly sample 100 rPairs per project for manual verification (all 111 for jBidWatcher).

- The majority (91.7%) of correct rPairs discovered are not in Webster or WordNet.

Evaluation Methods

- Extraction Accuracy
  - **7** large code bases, in Java & C, from Comment-Comment, Code-Code, Comment-Code
- Search-Related Evaluation
  - Comparison with SWUM [Hill Phd Thesis] in Code-Code

Search-Related Evaluation

In jBidWatcher, **“Add auction”**

Query expansion: **“XXX auction”**

Our approach	SWUM
new	register
register	…
…	

Search-Related Evaluation

In jBidWatcher, **“Add auction”**

JBidMouse.DoAuction(...)
AuctionServer.registerAuction(...)
AuctionManager.newAuctionEntry(...)
FilterManager.addAuction(...)
...

SWUM gold set

add→ register, do, new

our gold set

Search-Related Evaluation

In jBidWatcher, **“Add auction”**

add$\rightarrow$ register, do, new

Our approach
(55 words)

new
register
do
load
...

Precision = 3/55 = 5.5%
Recall = 3/3 =100%

SWUM
(84 words)

register
do
...

Precision = 2/84 = 2.3%
Recall = 2/3 = 67.7%

Search-Related Evaluation

In jBidWatcher, **"Add auction"**

add→ register, do, new

Our approach (55 words)	SWUM (84 words)
load	…
…	

Our approach achieves higher precision and higher/ equal recall for 5 out of 6 rPair groups in the gold set.

Precision = 3/55 = 5.5%
Recall = 3/3 =100%

Precision = 2/84 = 2.3%
Recall = 2/3 = 67.7%

Related Work

- Verb-DO (Direct Object) [Shepherd et al. AOSD] & SWUM - Improved version of Verb-DO [Hill Phd Thesis]
    - Requires Natural Language Processing (NLP) techniques
    - Requires manually generated heuristics

Conclusions

- A simple, general technique to automatically infer semantically related words from software context
  - No Natural Language Processing (NLP) required
  - Reasonable accuracy in 7 large C & Java code bases
  - The majority of rPairs discovered are not in the dictionaries or WordNet.
  - Higher precision & recall than the state of art