

The ILIAD Project: Language Technology Meets Linux Troubleshooting

David Martinez



THE UNIVERSITY OF

MELBOURNE

INTRODUCTION

Imagine ...

- You are a Debian GNU/Linux user and have just updated the local installation of `unstable` on your ThinkPad T41 laptop, only to find that GNOME no longer runs

Imagine ...

- You are a Debian GNU/Linux user and have just updated the local installation of `unstable` on your ThinkPad T41 laptop, only to find that GNOME no longer runs
- Armed with only a web browser and internet connection, how do you solve the problem?

Limitations of Web Search

- Low recall of keyword search
- Difficulties in constraining the search to documents from a fixed time period/relative to a particular spread of versions
- Problems with multi-document threads
- Resolved vs. unresolved threads

Other Complications

- Variable quality of information on different forums
- Variable formatting of data on different forums
- Variable quality of language
- Rants, subjectivity, reproducibility, ...

The Nature of the Beast

dtheorem

This is something that is desirable to me. The following is **not** a typo.

If I use my old `.config` file to compile a new kernel, will it cause problems if the kernel I'm compiling is **older** than the one I got the `.config` file from? Specifically, I want to use a `.config` file from 2.4.22 and use it to compile a 2.4.20.

lupin_the_3rd

You'll run into modules from one not being in the older kernel which may cause problems in the compile but you should be able to remedy that by noting which ones caused the problems and removing them from the config file... any reason you are moving back a stable kernel? 2.4.22 runs sooo much better on my box.

bulliver

My advice: just try it. Copy your `.config` file over to the new(old?) kernel source, then do a 'make oldconfig', it should prompt you for the differences between the `.config` file and changes in the source.

Not sure if it will work with an older kernell source but you might as well try right?

PROPOSED SOLUTION

System Outline

1. Crawl and normalise data from various Linux user forums
2. Pre-process data variously
 - filter out spam/uninformative threads
 - extract basic specifications of original query
 - extract basic specifications of solution
 - rate each thread according to solvedness, quality of solution, ...
3. Query system via (optionally) constrained queries and generate thread ranking in response

Top-level Architecture

- “Interpolate” IR with IE
- IR: simple keyword matching
- IE:
 - ★ thread-level features
 - ★ “fuzzy matching” of constraints in query with specifications of each thread

Why ILIAD?

- ILIAD = Improved Linux Information Access by Data Mining

Component Technologies

- Crawler/data normalisation
- Code/command/config file parser
- Thread-level filters
- Text normaliser and POS tagger
- NER (packages, versions)
- Parser

Crawler and Data Normalisation

- Crawl data from:
 - ★ LinuxQuestions (36,551 threads)
 - ★ Debian mailing lists (9,500 threads)
 - ★ Fedora Forum (900 threads)
 - ★ (Ubuntu Forum, Experts Exchange, ...)
- Varying formats: vBulletin, phpBB, pipermail, mailman
- Data normalised and stored in common mySQL database

Code/command/config File Parser

- Identify chunks of code/commands/config file excerpts for:
 - ★ feature engineering in thread-level processing
 - ★ parser purposes
- Current alpha-version (hack) version based on average word length, relative occurrence of oov words, HTML formatting
- Also sentence, word tokeniser for remainder of text

Thread-level Filters

- Filter out threads that are:
 - ★ too short or too long
 - ★ span too long a time
 - ★ have been soft-deleted from the forum

Text Normaliser and POS Tagger

- TODO: Normalise text (spelling correction, deabbreviation, remove mail headers/footers, punctuation normalisation, ...)
- TODO: develop POS tagger for the Linux user forum domain
 - ★ semi-supervised learning with small-scale annotation of domain data?

NER (packages, versions)

- Currently: identify (version?) numbers and distribution name
- TODO: generate full NER capabilities based on:
 - ★ gazetteer based on (distro-specific) package lists
 - ★ coupling of packages/distros and version numbers based on knowledge of actual package/distro versions
 - ★ semi-supervised learning with small-scale annotation of domain data?

Parser

- Ideally we want to be able to parse (some) of the text to aid in information extraction, ideally using:
 - ★ full ERG (English Resource Grammar)
 - ★ type-level lexical acquisition
 - ★ on-the-fly token-level supertagger
- TODO: treebank selection of (clean) thread data
- TODO: integrate parse selection with lexical acquisition

Supertagging for robust parsing

- Supertagging = POS tagging with a very fine-grained tagset

*How*_[WRB] *does*_[VBZ] *that*_[DT] *sound*_[VB] *for*_[IN] *you*_[PRP] ?_[.]



*How*_[adj-wh-le] *does*_[va-does-le] *that*_[n--pr-dei-sg-le]
*sound*_[v-pp-pp-seq-le] *for*_[p-le] *you*_[n--pr-you-le] ?_[.]

Supertagging: A Shopping List

- We desire a method that:
 - ★ works across different domains
 - ★ can be trained directly from treebank data
 - ★ scales to a large tagset (100s of tags)
 - ★ achieves state-of-the-art accuracy
 - ★ is probabilistic

CURRENT STATE OF PLAY

Thread-level Evaluation

1. task vs. discussion orientation
2. completeness of original problem specification
3. relative “solvedness” of thread

Annotation

- Triple annotation of 250 threads across different forums for the three tasks
- For each thread, annotate on scale of 1–5
- Kappa measure: binarise ($[0, 0.5]$ vs. $(0.5, 1.0]$)

	κ	Spearman's ρ
Task orientation	0.64	0.84
Completeness	0.21	0.77
Solvedness	0.38	0.93

Preliminary Experiments

- Attempt to model tasks as three independent binary classification ($[0, 0.5]$ vs. $(0.5, 1.0]$) or regression task
- Highly skewed: most threads are task oriented (81.6%), well-specified (94.8%), and solved (76.5%)
- Baseline: Bayesian text classification methods (rainbow)
- Learning algorithms currently targeted: SVMs, maxent, memory-based learning, perceptron, linear regression

Feature Set

- ★ Thread-level features (no. of posts, distribution of posts from initiator, average sentence length, bag-of-words, etc.)
- ★ Post-level features (linux distribution mentions, code-lines, avg. word length, etc.)
- ★ Post-level features for: initial post, first response, bag of all other responses, final post from initiator
- ★ Common feature set for all three tasks

Preliminary Results

	MC	TiMBL	JRip	DTrees	SVM	Perceptron	Linear Reg.
Task orientation	81.6	76.2	80.7	71.7	78.3	72.9	79.1
Completeness	94.8	91.4	94.8	94.8	94.0	90.9	94.4
Solvedness	76.5	65.3	77.0	68.5	76.5	72.1	73.9

- ★ MC (Majority Class) best in almost all classes.
- ★ Classifiers and linear regression similar performance.
- ★ Learning curves do not show clear progression.

Next Steps

- ★ Evaluate regression
- ★ More feature engineering
- ★ Hierarchical models, Boosting
- ★ Semi-supervised approaches
- ★ Active learning for more efficient annotation

Next Big Steps

- ★ Generation of IR dataset (queries and relevance judgements)
- ★ Domain tuning of NER/POS tagger
- ★ Experiments with robustness in the ERG
- ★ Generation of thread-level “summaries”

Overall Big Questions

- ★ Can we get the parser to do sensible things with very noisy data?
- ★ Can IR+NLP outdo simple IR?

Summary

- ★ ILIAD = I mproved Linux Information Access by Data Mining
- ★ Investigation of the applications of NLP to web data
- ★ (Hard but) nice application to showcase NLP competencies