What if.... we could automatically test our textual instructions?

Crash-test-virtual-machines

Pablo Duboue\textsuperscript{1,2}

DebConf10, NYC

\textsuperscript{1}pablo.duboue@gmail.com
\textsuperscript{2}DrDub on OFTC
Outline

Generalities
   The Problem
   Debian

Details
   Non-overwhelming amount of details

Even More Natural Language Processing
   More NLP
Outline

Generalities

The Problem
Debian

Details
Non-overwhelming amount of details

Even More Natural Language Processing
More NLP
Branavan et al. (2009)

- This talk is an exercise in brainstorming about how a very interesting paper can be of use within Debian.
  - Reinforcement Learning for Mapping Instructions to Action
    - Computer Science and Artificial Intelligence Laboratory
    - Massachusetts Institute of Technology
  - Presented at the Annual Meeting of the Association for Computational Linguistics (ACL ’2009)
    - Best paper award.
The Problem

- Instructions from the Microsoft KB (how to remove the “msdownld.tmp” temporary folder).
  - Click start, point to search, and then click for files or folders.
  - In the search results dialog box, on the tools menu, click folder options.
  - In the folder options dialog box, on the view tab, under advance settings, click show hidden files and folders, and then click to clear the hide file extensions for known file types check box.
  - Click apply, and then click ok.
  - In the search for files or folders named box, type msdownld.tmp.
  - In the look in list, click my computer, and then click search now.
  - In the search results pane, right-click msdownld.tmp and then click delete on the shortcut menu, a confirm folder delete message appears.
  - Click yes.
**Output**

\[ u: \] click **Run**, and press **OK** after typing **secpol.msc** in the open box.

\[ \tilde{a}: \] **left-click** Run \[ c: \] type-into \[ R: \] [open "secpol.msc"]

\[ e: \] 

---

\[ u: \] click **Run**, and press **OK** after typing **secpol.msc** in the open box.

\[ \tilde{a}: \] **left-click** Run \[ c: \] type-into \[ R: \] [open "secpol.msc"

\[ e: \] 

---

\[ u: \] click **Run**, and press **OK** after typing **secpol.msc** in the open box.

\[ \tilde{a}: \] **left-click** Run \[ c: \] type-into \[ open "secpol.msc" \[ c: \] left-click \[ R: \] [OK]

\[ e: \] 

---
## Results

<table>
<thead>
<tr>
<th></th>
<th>Action</th>
<th>Sent.</th>
<th>Doc.</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random baseline</td>
<td>0.128</td>
<td>0.101</td>
<td>0.000</td>
<td>——</td>
</tr>
<tr>
<td>Majority baseline</td>
<td>0.287</td>
<td>0.197</td>
<td>0.100</td>
<td>——</td>
</tr>
<tr>
<td>Environment reward</td>
<td>*0.647</td>
<td>*0.590</td>
<td>*0.375</td>
<td>0.819</td>
</tr>
<tr>
<td>Partial supervision</td>
<td>0.723</td>
<td>*0.702</td>
<td>0.475</td>
<td>0.989</td>
</tr>
<tr>
<td>Full supervision</td>
<td>0.756</td>
<td>0.714</td>
<td>0.525</td>
<td>0.991</td>
</tr>
</tbody>
</table>
The Technology

- Training Data
  - Instructions plus
  - An automatic verifier for step-wise success.

- Output
  - a script (e.g., a shell script) that executes the commands in the text.

- Method
  - Run the OS in a VM, execute first at random, then start informing a model mapping text excerpts into actions.

- Mapping instructions is complicated by out-of-order phrases (“select run after clicking start”), aggregated phrases (“remove everything in the folder”), high level instructions (“start the Web browser”), etc.
Outline

Generalities
  The Problem
  Debian

Details
  Non-overwhelming amount of details

Even More Natural Language Processing
  More NLP
Why Should We Care?

- Users want instructions.
- Maintainers write instructions.
- System changes, instructions become stale.
- Debian has a strong tradition in automating as much as possible processes and tests related to the health of the system.
  - E.g., lintian, piu-parts, mini-dinstall and a large etc.
- This new technology might enable continuous testing of instructions related to packages.
Which Instructions?

- Instructions in man pages (hadoop.1)
  The "/etc/hadoop/conf" link is managed by the `alternatives(8)` command so you should not change this symlink directly.
  To see what current alternative(8) Hadoop configurations you have, run the following command:
  
  ```
  # alternatives --display hadoop
  hadoop - status is auto.
  link currently points to /etc/hadoop/conf.pseudo
  /etc/hadoop/conf.empty - priority 10
  /etc/hadoop/conf.pseudo - priority 30
  Current 'best' version is /etc/hadoop/conf.pseudo.
  ```
  
  (...
  To activate your new configuration and see the new configuration list:
  
  ```
  # /etc/init.d/hadoop-namenode restart
  ```
How Can We Use It?

- Special test target in debian/rules
  - For example, instruction-tests
  - The target produces the training necessary by the system.
  - First upload: debian-wide model is updated.
  - New uploads: if trained model stops working, the maintainer will be informed. This won’t happen any time soon. Indeed, first we can

- Add special mark-up to discussion forums.
  - Enrich the mark up of the forums with tags that signal the problem and the instructions to solve it.
  - We can then automatically warn the readers when a solution no longer works in a newer version of Debian.
More Food For Thought

• [http://forums.debian.net](http://forums.debian.net)?
  • Docs, Howtos, Tips & Tricks: 622 topics

• Debian-installer specific instructions?

• Instructions in debconf templates?

• Multi-linguality?
  • Create a model of text into instructions for two languages.
  • In their paper, they show the model can (with some luck) be used to generate text from the actions.
  • For instructions lacking in one, language, they can be generated from the actions a understood in the first language.
    • Hopefully better quality instructions than using machine translation.
    • Instructions are held up-to-date automatically.
About the Speaker

- CU is my graduate school ’alma mater’
  - NLP group
  - PhD CS in 2005
- IBM researcher since then
  - Part of the DeepQA team
- NLP is starting to work, thanks to increased computational power and advances in machine learning.
- Passionate about Free Software and Debian
  - Interested in getting more NLP into Debian
  - Will have plenty of time in the upcoming months, so looking forward to do something useful within Debian.
Outline

Generalities
- The Problem
- Debian

Details
- Non-overwhelming amount of details

Even More Natural Language Processing
- More NLP
Interesting Idea Beyond NLP

- How reinforcement learning works.
  - Build over time a model of the mapping between actions and the way the environment is affected by the actions.
  - Sample the actions at a given point and see how the environment reacts.

- When mapping instructions into actions:
  - The instructions guide the sampling of actions and compute rewards.
  - A sequence of actions that maximizes rewards is expected to be the sequence of actions corresponding to the instructions.
Formally

- **Input** is a document \( d = (u_1, \ldots, u_l) \), a sequence of sentences (and each sentence is a sequence of words).

- **Output** is a sequence of actions \( \vec{a} = (a_0, \ldots, a_{n-1}) \). Actions are predicted and executed sequentially (and each sentence is consumed in order, but its words are in any order).

- An action \( a = (c, R, W') \), contains a command \( c \), its parameters \( R \) and the words it encompasses \( W' \)
  - The parameters refer to object in the environment and/or words in the document.

- The environment \( \varepsilon \) contains the objects and their properties.
  - The env. \( \varepsilon \) changes with a command \( c \) with params \( R \) according to a transition distribution \( p(\varepsilon' | \varepsilon, c, R) \).

- The state of the mapping at a given point is given by a mapping state \( s \), a tuple \( (\varepsilon, d, j, W) \).
Output

\textbf{u:} click Run, and press OK after typing \texttt{secpol.msc} in the open box.

\textbf{ā:} \texttt{C:\left-click Run [Run...]}
Reward Function

- After each sentence, check whether any of the words in the remaining of the instructions appear in the screen.
  - If you were expected to open the control panel up to this point, then the label “control panel” should be visible.
  - If none is visible, reward is -1
  - If some are visible, the reward is weighted between 0 to 1 depending on how many matches appear in the screen.

- The main drawback of the technique, without this approximation it will require unreasonable human intervention.
Environment

- The environment are the UI objects in the screen and object properties such as label, location, and parent window.
- Commands include click (left, right, double) and type-into
  - Take a UI object as a parameter
  - type-into also takes as a parameter the input text.
- A total of 4,000+ features represent the environment.
Training Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total # of documents</td>
<td>128</td>
</tr>
<tr>
<td>Total # of words</td>
<td>5562</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>610</td>
</tr>
<tr>
<td>Avg. words per sentence</td>
<td>9.93</td>
</tr>
<tr>
<td>Avg. sentences per document</td>
<td>4.38</td>
</tr>
<tr>
<td>Avg. actions per document</td>
<td>10.37</td>
</tr>
</tbody>
</table>
Training Algorithm

Input: A document set $D$, 
Feature representation $\phi$, 
Reward function $r(h)$, 
Number of iterations $T$

Initialization: Set $\theta$ to small random values.

1. for $i = 1 \ldots T$ do 
2.   foreach $d \in D$ do 
3.     Sample history $h \sim p(h|\theta)$ where 
3.6.       $h = (s_0, a_0, \ldots, a_{n-1}, s_n)$ as follows:
3.6.6.         for $t = 0 \ldots n-1$ do
3.6.6.6.             Sample action $a_t \sim p(a|s_t; \theta)$
3.6.6.7.             Execute $a_t$ on state $s_t$: $s_{t+1} \sim p(s|s_t, a_t)$
3.6.   end
3.8.   $\Delta \leftarrow \sum_t \phi(s_t, a_t) - \sum_a \phi(s_t, a) p(a | s_t; \theta)$
4.   $\theta \leftarrow \theta + r(h)\Delta$
5. end
end

Output: Estimate of parameters $\theta$
## Results

<table>
<thead>
<tr>
<th></th>
<th>Action</th>
<th>Sent.</th>
<th>Doc.</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random baseline</td>
<td>0.128</td>
<td>0.101</td>
<td>0.000</td>
<td>——</td>
</tr>
<tr>
<td>Majority baseline</td>
<td>0.287</td>
<td>0.197</td>
<td>0.100</td>
<td>——</td>
</tr>
<tr>
<td>Environment reward</td>
<td>∗0.647</td>
<td>∗0.590</td>
<td>∗0.375</td>
<td>0.819</td>
</tr>
<tr>
<td>Partial supervision</td>
<td>0.723</td>
<td>∗0.702</td>
<td>0.475</td>
<td>0.989</td>
</tr>
<tr>
<td>Full supervision</td>
<td>0.756</td>
<td>0.714</td>
<td>0.525</td>
<td>0.991</td>
</tr>
</tbody>
</table>
Outline

Generalities
- The Problem
- Debian

Details
- Non-overwhelming amount of details

Even More Natural Language Processing
- More NLP
Where To Go From Here

• Throw me tomatoes (save them for lunch!)
• Forget NLP and use reinforcement learning for something else.
• Let’s talk about where we can tinker with these within Debian.
• Other NLP tech I been considering:
  • Mailing lists summaries (GoogleNews style)
  • Automating meetbot "info" entries
  • Indexing the source code of Debian packages (deb-source-index is an IRCbot interface to a proof-of-concept index)
Mailing Lists Summaries

- Some teams (e.g., debian-java) have high traffic mailing lists.
  - Supplement the mailing list with a page with summaries and information relevant to the reader (e.g., automatic messages filtered to only certain packages).
- Debian-devel mailing list would profit from summaries in the style of the (now defunct) kernel-traffic summaries.
  - Been using the kernel-traffic as training material.
  - Can help complement the submitter-driven news services in the project, like Debian-News.
Automating MeetBot #info Entries

• MeetBot is an excellent bot to conduct meetings in IRC.
  • Written by Richard Darst in pure python.
  • Used heavily by the DebConf-team.

• MeetBot has a number of commands.
  • Many of these commands are used to make a succinct summary of what topics were discussed through the meeting, which action items were assigned and which topics were agreed upon.
    • It is possible to include extra information into these summaries by issuing a general ’#info’ command, but many times the participants forget to do so, producing lower quality summaries.

• I am interested in producing a summary including “also of note” items, in the vein of “an automatic model believes this messages should have been #info”.
NLP and Free Software

- NLP requires many, many eyes.
  - Plenty of errors in current systems.
  - But perfection is possible, just painful.

- I am hoping to contribute to the maintenance of NLP tools within Debian

- And also to get other contributors interested in NLP.
  - Considering offering an on-line course on NLP for Free Software contributors (ping me if you would be interested).
Summary

- Some cool new technology is becoming available.
  - Debian can take advantage of it quickly
- Doing reinforcement learning over full O/S VMs worked for some particular problem with some particular data.
  - Yes, it is a particular case, but it should tempt others to try similar things.
- Let’s add some NLP goodies to the Debian tooling.
- Keep in touch!
  - DrDub in OFTC / pablo.duboue@gmail.com