IRC QA Mining

Mining Question/Answer pairs from #tikiwiki history

This page describes the progress on distilling an usable resource from

http://irc.tiki.org/irclogger_logs/tikiwiki

The process involves:

- Downloading the logs
- Normalizing the IRC logs across different IRC logging clients
- Identifying users that are likely to have asked questions
- Identify the threads where said users participated
- Assembling the final corpus
- Indexing

To avoid having to annotate training data for the first process, DrDub used an approximation of finding IRC nicks that have only said 2 to 10 things in the whole logging history. The expectation is that the said users appear on #tikiwiki, ask a question receive an answer and left.

For the last step, DrDub used

http://www.ling.ohio-state.edu/~melsner/resources/chat-manual.html

a publically available implementation of Disentangling chat (2010) by Elsner and Charniak (http://dl.acm.org/citation.cfm?id=1950492)

Downloading the logs

```
$ curl http://irc.tiki.org/irclogger_logs/tikiwiki > tikiwiki.html
$ mkdir tiki_logs
$ cat tikiwiki.html |grep "raw text"|perl -pe 's/^(.*?)\<a href=\"//;s/\".*/\&raw=on/' > urls
$ tac urls | perl -ne 'chomp; $c++; print "curl \"$_\" > tiki_logs/$c.log\nsleep 1s\n";' > download.sh
$ chmod +x ./download.sh
$ ./download.sh
$ tac urls | perl -ne 'chomp; $c++; s/^.*date\=//; s/\,.*//; $d=$_; open(L,"tiki_logs/$c.log");
while(<L>){s/^\[/\[$d /; print;}; close L;' > ./tiki.log
```

Normalizing the logs

The chat disentangler code expects Pidgin logging format, which is very different from the formats available in the logs. The majority of the text seems to be in xchat log format, to convert the rest of the entries to that format use:

```
$ cat ./tiki.log | perl -pe 's/\]\-\!\- /\] \*\*\* /; s/\]\< /\] \</; if(!m/^\[[^\]]+\] \</ && !m/^\[[^\]]+\] \*/ && m/^\[/){if(m/^\[[^\]]+\] .*\: /){s/\] ([^\:]+)\: (.*)/\] \*\*\* $2 $1/;}elsif(m/^\[[^\]]+\] /){ s/\] /\] \*\*\* /}elsif(m/^\[[^\]]+\]\</){s/\]\</\] \</}else{s/^(\[[^\]]+\]) ([^ ]+) (.*)$/$1 <$2> $3/}}' > ./tiki2.log
```

Then extract the processed file using stripChatTikiNonAnon.py, this file is ready to be used by the disentangler code (put it in the `preprocess/` folder):

```
$ python ./preprocess/stripChatTikiNonAnon.py ./tiki2.log > tiki.preprocessed
```

(this will take a long long time)

This file is available for download at http://duboue.net/download/tiki.processed.bz2 (9.1Mb)

Identifying potential askers

We use the heuristic that any nick with 2 to 20 interactions of the history of the logs is a person that potentially asked a question (note, this is done on the output of the preprocessed file, so IRC nick changes are factored in).

```
$ cat ./tiki.processed | perl -e '%c=();while(<STDIN>){next if(m/^[0-9]+ [^ ]+ \*/); ($t,$n)=m/^([0-9]+) \<([^ ]+)\> /;$c{$n}++};foreach$n(keys %c){print "$n\n" if($c{$n}>=2 && $c{$n}<20)}' > askers
```

Disentangling the logs

Besides downloading the disentangler python scripts, you'll need to download the maximum entropy `megam` program (DrDub used the binaries as the source OcamL code doesn't seem to build on Debian Wheezy, if you know how to fix that, please update this page).

The disentangler assumes the evaluation data to be also annotated, so we add fake annotations:

```
$ cat tiki.processed | perl -ne 'if($c % 10 == 0){$i++;$c=1}else{$c++}; print "T$i $_"' > tiki.annot
```

```
$ PYTHONPATH=$PWD/utils python model/classifierTest.py IRC/dev/linux-dev-0X.annot tiki.annot data/linux-unigrams.dump data/techwords.dump tiki
```

(you might need to symlink `analysis` into the `model` folder and do other changes to get the provided code to work, besides the `PYTHONPATH` hack.)

Running the classifierTest.py will take a long long time. It will take even longer if you wait for the evaluation and description code to finish running (which is not needed for this). You can comment it out on the `classifierTest.py` script or cancel it once `tiki/129/predictions` is complete.

Once the predictions are available do

```
$ PYTHONPATH=$PWD/utils python model/greedy.py tiki.annot ./tiki/129/predictions ./tiki/129/devkeys > tiki.predicted
```

here you really want to modify greedy.py to remove the evaluation / description code at the bottom of the file. Also, check the output file for spurious extra text at the beginning / end of the file due to the output redirection.

Assembling the final corpus

The faster way to find the relevant threads (in terms of human effort) was to materialize all the different threads as separate files:

```
$ cat tiki.predicted | perl -ne 'if(m/^T[0-9]+/){($d)=m/^T([0-9]+) /;open(T,">>tiki/$d.txt"); s/^T$d //; print T $_;
close T}'
```

(note: this will create 200k+ files in that folder, you might ran out of inodes in your file system)

Now we want to keep the threads with a potential asker at the top three interactions and with at least 5 lines on it"

```
$ mkdir tiki/interesting
$ perl -e 'open(N,"askers");@n=<N>;chomp @n; %n=map{$_=>1}@n;close N;
opendir(D,"tiki");@d=readdir(D);closedir(D); foreach$f(@d){next unless $f=~m/\.txt/;open(F,"tiki/$f"); $good=0;
$l=0; while(<F>){ ($n,$t)=m/^[0-9]+ ([^ ]+) (.)/; if($t eq ":"){$l++; $n=~s/\<//;$n=~s/\>//; if($l<3 &&
$n{$n}){$good=1}}}; close F; if($good && $l>5){`cp tiki/$f tiki/interesting`;}}'
```

This populates a folder `tiki/interesting` with the potential QA threads (about 600 threads).

Now, the chat disentangling process is of course imperfect and it is made worse by using their model of linux discussions to differentiate normally occurring text vs. technical text. In practice, many of the identified threads are fairly small. To obtain a more usable resource, it is convenient to augment each of the identified threads with any other interaction involving the target potential asker:

```
$ mkdir tiki/augmented
$ perl -e 'open(N,"askers");@n=<N>;chomp @n; %n=map{$_=>1}@n;close N;
opendir(D,"tiki");@d=readdir(D);closedir(D); %f=(); foreach$f(@d){next unless $f=~m/\.txt/;open(F,"tiki/$f");
$good=0; $l=0; while(<F>){ ($n,$t)=m/^[0-9]+ ([^ ]+) (.)/; if($t eq ":"){$l++; $n=~s/\<//;$n=~s/\>//;
if($n{$n}){if($f{$n}){push@{$f{$n}}, $f}else{$f{$n}=[ $f ];}}}}}; close F; foreach$n(keys %f){print
"$n\t".join("\t",@{$f{$n}})."\n"}' > threads
$ perl -e 'open(T,"threads");@t=<T>;chomp @t; %t=map{@x=split(/\t/,$_); $x=shift@x; $x=> [ map { "tiki/$_"
} @x ]}@t;close T; opendir(D,"tiki/interesting");@d=readdir(D);closedir(D); %f=(); foreach$f(@d){next unless
$f=~m/\.txt/;open(F,"tiki/interesting/$f"); $l=0; @a=(); while(<F>){ ($n,$t)=m/^[0-9]+ ([^ ]+) (.)/; if($t eq
":"){$l++; $n=~s/\<//;$n=~s/\>//; if($t{$n}){push@a,$n}}}; close F; system("cat tiki/interesting/$f ".join("
",map {join(" ",@{$t{$_}}) } @a) . " | sort -n | uniq > tiki/augmented/$f");}'
```

(note, this works because the preprocessing assigns a unique, fictitious "second" in time to each interaction.)

Neither of these approaches is yet producing optimal results (you can take a look for yourself:

- interesting folder, in tar.bz2 format (160k)
- augmented folder, in tar.bz2 format (332k)

Indexing

For indexing, we want to identify the first 5 lines of each meaningful interaction. For that we take the first line by a potential asker in `interesting` and the next 5 lines to that line in `augmented`.

$ perl -e 'open(N,"askers");@n=<N>;chomp @n; %n=map{$_=>1}@n;close N; opendir(D,"tiki/interesting");@d=readdir(D);closedir(D); %f=(); foreach$f(@d){next unless $f=~m/\.txt/;open(F,"tiki/interesting/$f"); $tt=0; while(<F>){ ($d,$n,$t)=m/^([0-9]+) ([^]+) (.)/; if($t eq ":"){$n=~s/\</;$n=~s/\>//; if($n{$n}){$tt=$d; last;}}}; close F; $p=0;$e=0;@l=();open(F,"tiki/augmented/$f");while(<F>){($d,$n,$t,$w)=m/^([0-9]+) ([^]+) (.)(.*)/; next unless($t eq ":"); $e=$d; if($d==$tt || ($p>0&&$p<5)){$w=~s/[^a-zA-Z0-9]+/ /g;push@l,$w;$p++}}; close F;$f=~s/\.txt//;print ("$f $tt $e\n\t".join("\n\t",@l)."\n");}' > questions.txt

You can see that file here: http://duboue.net/download/tiki-questions.txt (220k)

Pumping it into elasticsearch:

$ perl -e 'use JSON; open(Q,"questions.txt");@q=();while(<Q>){chomp;if(m/^\t/){s/\s+/ /g;$q[-1]->{question} .= $_;}else{@s=split(/\s+/,$_);push@q,{ thread=>$s[0], postDateEpoch=>int($s[1]), postDate=>scalar(localtime(int($s[1]))), threadEndDate=>scalar(localtime(int($s[2]))), threadEndDateEpoch=>int($s[2]), question => ''};}}; foreach$q(@q){ open(C, "|curl -XPUT http://localhost:9200/tiki/thread/$q->{thread} --data-binary \@/dev/stdin"); print C to_json($q); close C;}'

Each JSON object will look like this:

```
{"question":
" Hi is there a setting on tikiwiki to get a description picked up by FB when you add a link to Facebook FB picks up the metadata description of the site not the description of the page blog luciash you have FB just attach this link http www avonsys com Firefox FJ",
"threadEndDate":"Thu Nov 12 23:33:01 2009",
"thread":"31814",
"postDate":"Thu Nov 12 23:24:00 2009",
"postDateEpoch":1258086240,
"threadEndDateEpoch":1258086781
}
```

(more fields, like the nicks of the people involved in the thread can be added in some moment.)

Querying it:

$ curl -XGET http://localhost:9200/tiki/thread/_search?pretty=true -d '{ "query" : { "text" : { "text" : "htacces not found in facebook app" } } }'

This query returns 45 hits, here are the first three:

- "_score" : 0.10639295, "_source" : {"question":" Maybe u should check the permission on ur htaccess too Did you run the installer setup sh never understand how to run setup sh and the htacces are chmod 777 http doc tikiwiki org tiki index php page ref id 7 hmm 777 is full access to everyone ","threadEndDate":"Tue Dec 16 16:15:00 2008","thread":"8910","postDate":"Mon Dec 15 18:13:00 2008"}
- "_score" : 0.09947969, "_source" : {"question":" Hi is there a setting on tikiwiki to get a description picked up by FB when you add a link to Facebook FB picks up the metadata description of the site not the description of the page blog luciash you have FB just attach this link http www avonsys com Firefox FJ","threadEndDate":"Thu Nov 12 23:33:01 2009","thread":"31814","postDate":"Thu Nov 12 23:24:00

2009"}
- "_score" : 0.081796534, "_source" : {"question":" hi all anyone from brasil need developers from brasil ok i found the email and will send ","threadEndDate":"Mon Mar 15 14:34:01 2010","thread":"40071","postDate":"Mon Mar 15 14:31:01 2010"}

The index can be currently accessed at: http://tikiqa.duboue.net

(for example, try I have this problem with tikiwiki not opening my links in a new browser.)

A raw access is available through http://tikiqa.duboue.net/?raw=1&query=htaccess+FB (returns the underlining elasticsearch JSON response), this is useful to integrate this in tiki.org or where people see fit.

The source for that page is in GitHub.

Next steps

Potential next steps:

- Integration within http://tiki.org help
- Use the extracted questions to train a classifier, so to extract less obvious pairs
- Move http://tikiqa.duboue.net to a tiki.org server and make it more visually appealing

You can ping DrDub on FreeNode (usually at ##foulab) or on Twitter.

See also: http://duboue.net/blog8.html

Related

- Natural language processing