Scripting, Plotting, Latex

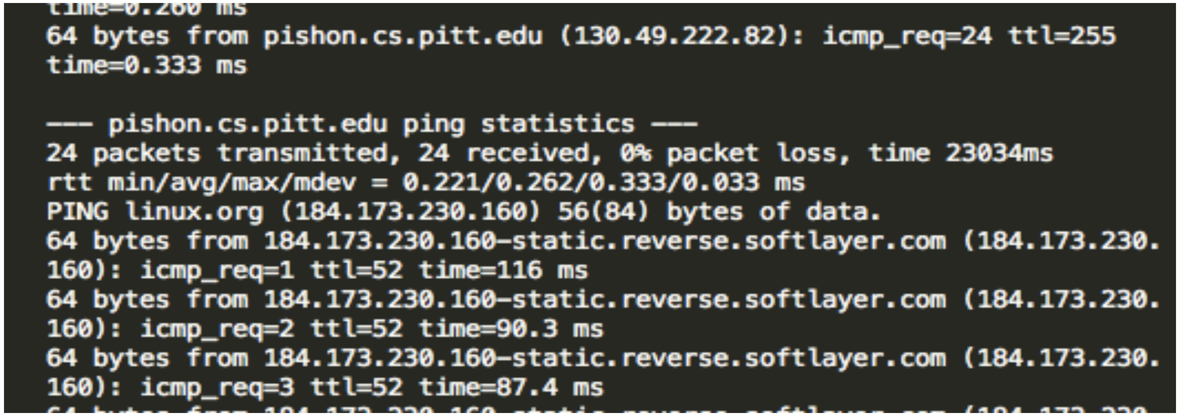
Fan Zhang

Outline

- Learn to do your work in scripts with one example
    - Major commands that are covered (Examples borrowed from Bill Garrison’s materials)
        - grep, sed, awk
        - plot
        - latex
- Some useful links

Suppose we have this task

- Given a file with different lines of information
- Generate a report about it
  - See example: bad_name.bones
  - Generate a report about the connection time

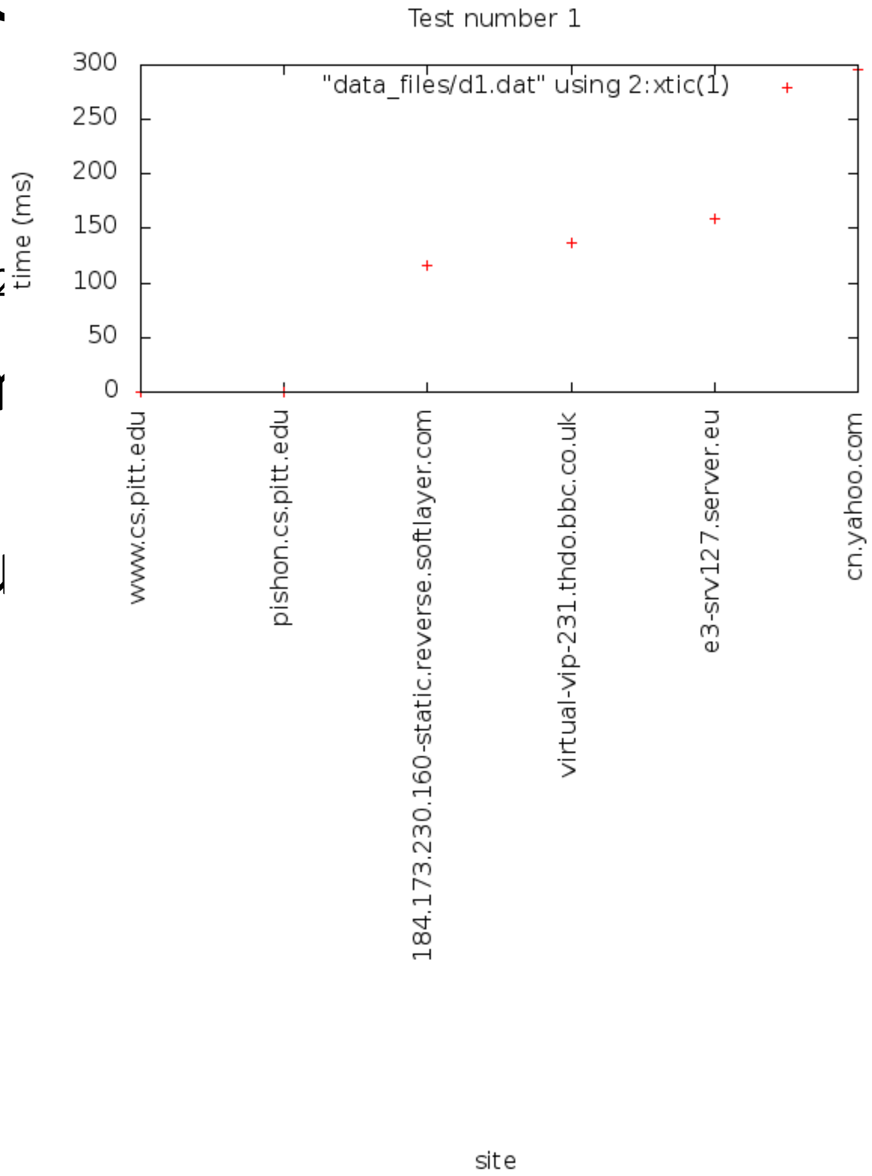


```
time=0.260 ms
64 bytes from pishon.cs.pitt.edu (130.49.222.82): icmp_req=24 ttl=255
time=0.333 ms

--- pishon.cs.pitt.edu ping statistics ---
24 packets transmitted, 24 received, 0% packet loss, time 23034ms
rtt min/avg/max/mdev = 0.221/0.262/0.333/0.033 ms
PING linux.org (184.173.230.160) 56(84) bytes of data.
64 bytes from 184.173.230.160-static.reverse.softlayer.com (184.173.230.
160): icmp_req=1 ttl=52 time=116 ms
64 bytes from 184.173.230.160-static.reverse.softlayer.com (184.173.230.
160): icmp_req=2 ttl=52 time=90.3 ms
64 bytes from 184.173.230.160-static.reverse.softlayer.com (184.173.230.
160): icmp_req=3 ttl=52 time=87.4 ms
64 bytes from 184.173.230.160-static.reverse.softlayer.com (184.173.230.
```

Steps we ne

- Extract the data lines
  - 64 bytes from pishon.cs.pitt.edu (130.49.22
- Extract the detailed nun
  - pishon.cs.pitt.edu 0.239ms
- Plot about each individu
- Generate a PDF report



Extracting the data lines

- Extract the lines
  - Use grep
    - grep icmp_req bad_name.bones > lines.dat

- Grep cheatsheet
  - http://www.ericagamet.com/wp-content/uploads/2011/11/Erica-Gamets-GREP-Cheat-Sheet.pdf

Get the figures out

- Use sed
  - sed 's/64 bytes from \([A-Za-z0-9\.-]*\) ([0-9\.]*): icmp_req=\([0-9]*\) ttl=[0-9]* time=\([0-9\.]*\) ms/\2,\1,\3/' < lines.dat > parsed.dat
- sed tutorial:
  - http://www.grymoire.com/Unix/sed.html
- Essential:
  - Sed ‘s//’ < old > new

Split the data

- Use awk

```
for i in {1..24}
        do
        awk "/^$i,/" < parsed.dat | sed "s/^$i,\([A-Za-z0-9\.-]*\),\([0-9\.]*\)/\1 \2/" >> data_files/d$i.dat
done
```

- Awk tutorial
  - http://www.grymoire.com/Unix/Awk.html

More about awk

- Counting the size of files
  ```
  ls –l *.* | awk ‘{sum+=$5} END {print sum}’
  ```

- Counting the ram used by each user
  ```
  ps aux | awk ‘NR!=1{a[$1]+=$6;} END {for (i in a) print I “, “ a[i] “KB”;}’
  ```

More about AWK

- BEGIN { }
  - Before processing
- {}
  - Processing
- END {}
  - After processing

Plot the data

- Plot one file

```
set title "Test number **i**"
set xlabel "site"
set ylabel "time (ms)"
set term png size 600, 800 enhanced font "Vera,12"
set output "pngs/plot_**i**.png"
set xtics rotate
plot "data_files/d**i**.dat" using 2:xtic(1)
```

- Gnuplot tutorial
  - http://people.duke.edu/~hpgavin/gnuplot.html

Basics of gnu plot

- Customizing scales
  - xtic, ytic
- Specifying columns
  - Using 2:1
- Set style
  - set style data histogram

Generate plot files for different data files

- Using what we have learned

```
for i in {1..24}
do
sed "s/\*\*i\*\*/$i/" < plot_i.gp > gp_scripts/plot_$i.gp
done
```

Calculate the average and plot

-
  ```
  sed "s/\*\*i\*\*/AVG/" < plot_i.gp > gp_scripts/plot_AVG.gp
  ```

-
  ```
  gnuplot gp_scripts/plot_AVG.gp
  ```

Generate the report

- Using latex
  - # create report.tex, a latex file with all plots
  - echo '\documentclass{article}' > report.tex
  - echo '\usepackage{graphicx}' >> report.tex
  - echo '' >> report.tex
  - echo '\begin{document}' >> report.tex
  - echo '' >> report.tex

  - for i in {1..24}
  - do
  -     echo "\\includegraphics[width=\\textwidth]{pngs/plot_$i.png}" >> report.tex
  - done

  - echo "\\includegraphics[width=\\textwidth]{pngs/plot_AVG.png}" >> report.tex
  - echo '' >> report.tex
  - echo '\end{document}' >> report.tex

  - # compile latex file into pdf
  - pdflatex report

Basic knowledge of latex

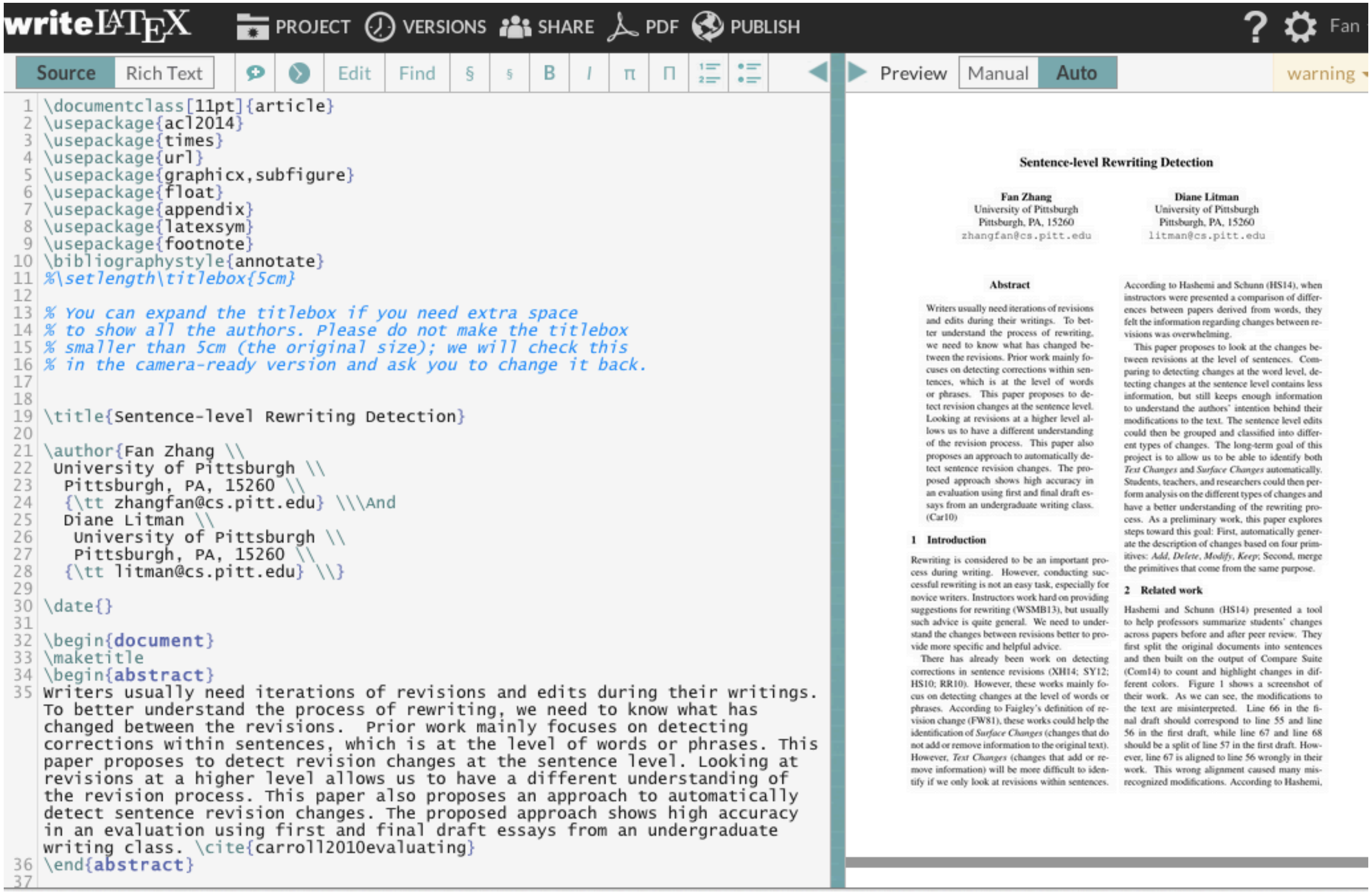
- \begin{} \end{}
  - Document, section, paragraph
- Introducing packages
  - \usepackage{}
- Insert graph,table
  - \includegraphics{}, \begin{tabular}…
- Math formulas
  - $, $$

Latex

- Latex tutorial:
  - http://ece.uprm.edu/~caceros/latex/introduction.pdf
- Writing papers with latex
- Tools that make latex easier:
  - texmaker
  - http://www.xm1math.net/texmaker/

Putting things in the cloud

- www.writelatex.com



```
\documentclass[11pt]{article}
\usepackage{acl2014}
\usepackage{times}
\usepackage{url}
\usepackage{graphicx,subfigure}
\usepackage{float}
\usepackage{appendix}
\usepackage{latexsym}
\usepackage{footnote}
\bibliographystyle{annotate}
%\setlength\titlebox{5cm}

% You can expand the titlebox if you need extra space
% to show all the authors. Please do not make the titlebox
% smaller than 5cm (the original size); we will check this
% in the camera-ready version and ask you to change it back.


\title{Sentence-level Rewriting Detection}

\author{Fan Zhang \\
 University of Pittsburgh \\
  Pittsburgh, PA, 15260 \\
  {\tt zhangfan@cs.pitt.edu} \\\And
  Diane Litman \\
   University of Pittsburgh \\
   Pittsburgh, PA, 15260 \\
  {\tt litman@cs.pitt.edu} \\}

\date{}

\begin{document}
\maketitle
\begin{abstract}
Writers usually need iterations of revisions and edits during their writings.
To better understand the process of rewriting, we need to know what has
changed between the revisions.  Prior work mainly focuses on detecting
corrections within sentences, which is at the level of words or phrases. This
paper proposes to detect revision changes at the sentence level. Looking at
revisions at a higher level allows us to have a different understanding of
the revision process. This paper also proposes an approach to automatically
detect sentence revision changes. The proposed approach shows high accuracy
in an evaluation using first and final draft essays from an undergraduate
writing class. \cite{carroll2010evaluating}
\end{abstract}

```

Sentence-level Rewriting Detection

Fan Zhang
University of Pittsburgh
Pittsburgh, PA, 15260
zhangfan@cs.pitt.edu

Diane Litman
University of Pittsburgh
Pittsburgh, PA, 15260
litman@cs.pitt.edu

Abstract

Writers usually need iterations of revisions and edits during their writings. To better understand the process of rewriting, we need to know what has changed between the revisions. Prior work mainly focuses on detecting corrections within sentences, which is at the level of words or phrases. This paper proposes to detect revision changes at the sentence level. Looking at revisions at a higher level allows us to have a different understanding of the revision process. This paper also proposes an approach to automatically detect sentence revision changes. The proposed approach shows high accuracy in an evaluation using first and final draft essays from an undergraduate writing class. (Car10)

1 Introduction

Rewriting is considered to be an important process during writing. However, conducting successful rewriting is not an easy task, especially for novice writers. Instructors work hard on providing suggestions for rewriting (WSMB13), but usually such advice is quite general. We need to understand the changes between revisions better to provide more specific and helpful advice.

There has already been work on detecting corrections in sentence revisions (XH14; SY12; HS10; RR10). However, these works mainly focus on detecting changes at the level of words or phrases. According to Faigley's definition of revision change (FW81), these works could help the identification of *Surface Changes* (changes that do not add or remove information to the original text). However, *Text Changes* (changes that add or remove information) will be more difficult to identify if we only look at revisions within sentences. According to Hashemi and Schunn (HS14), when instructors were presented a comparison of differences between papers derived from words, they felt the information regarding changes between revisions was overwhelming.

This paper proposes to look at the changes between revisions at the level of sentences. Comparing to detecting changes at the word level, detecting changes at the sentence level contains less information, but still keeps enough information to understand the authors' intention behind their modifications to the text. The sentence level edits could then be grouped and classified into different types of changes. The long-term goal of this project is to allow us to be able to identify both *Text Changes* and *Surface Changes* automatically. Students, teachers, and researchers could then perform analysis on the different types of changes and have a better understanding of the rewriting process. As a preliminary work, this paper explores steps toward this goal: First, automatically generate the description of changes based on four primitives: *Add*, *Delete*, *Modify*, *Keep*; Second, merge the primitives that come from the same purpose.

2 Related work

Hashemi and Schunn (HS14) presented a tool to help professors summarize students' changes across papers before and after peer review. They first split the original documents into sentences and then built on the output of Compare Suite (Com14) to count and highlight changes in different colors. Figure 1 shows a screenshot of their work. As we can see, the modifications to the text are misinterpreted. Line 66 in the final draft should correspond to line 55 and line 56 in the first draft, while line 67 and line 68 should be a split of line 57 in the first draft. However, line 67 is aligned to line 56 wrongly in their work. This wrong alignment caused many misrecognized modifications. According to Hashemi,

- Yay!