The KrdWrd CANOLA Corpus – Gathering Training Data for Sweeping Web Pages

November 25, 2010

Abstract

This document describes the KrdWrd CANOLA Corpus.

The CANOLA Corpus is a visually annotaded English web corpus for training the KrdWrd classification engine to remove boiler plate on unseen web pages. It was harvested, annotaded and evaluated by the tools and infrastructur of the KrdWrd Project.

Contents

The KrdWrd Project[@KRDW] deals with the design of an abstract architecture for A) the unified treatment of Web data for automatic processing, *without* neglecting visual information, on annotation and processing side and B) the appropriate annotation tool to gather data for supervised processing of such data.

The Project comprises an implementation appropriate for pre-processing and cleaning of Web pages, where users are provided with accurate Web page presentations and annotation utilities in a typical browsing environment, while machine learning (ML) algorithms also operate on representations of the visual rendering of Web pages. The system also preserves the original Web documents and all the additional information contained therein to make different approaches comparable on identical data.

The system is sketched in [SS09].

For training the KrdWrd ML Engine, a substantial amount of hand-annotated data, viz. Web pages, are needed. Following, we present the parts of the system that cover the acquisition of training data, i.e. the steps before training data can be fed into a ML Engine.

Then, after an overview of the sequence of steps needed to gather new training data in 1, an in-depth description of the processing steps *before* Web pages can be presented to annotators in 2, presentation of the actual tool annotators use in 3, and the compilation of their submitted results in 4, we will be ready to feed the KrdWrd Gold Standard to a ML Engine.

1 System Overview

Two fundamental ideas behind this part of the system are: firstly, Web pages have a textual representation, namely the text they contain, a structural representation, namely their DOM tree, and a visual representation, namely their rendered view – all representations should be considered when automatically cleaning Web pages, and consequently, all should be annotated during acquisition of training data for ML tasks. Secondly, data acquisition for training of supervised ML algorithms should preserve pristine, unmodified versions of Web pages – this will help to reproduce results *and* to compare those of different architectures.

1.1 Functional Walk-Through

Gathering a set of sample pages is at the beginning of tagging new data. The process needs to be coordinated by the administrators of the system, i.e. server level access is needed to make new corpora available for later tagging by users. The Process starts with a list of seed terms which are used to construct an ad-hoc corpus of Web pages where the result is a list of Uniform Resource Locators (URL[1]).

The URL list is then *harvested*, i.e. the according Web pages are downloaded and saved for further processing. This process is coordinated by the administrators of the system and is started as automated batch-job on the server where its input is the URL List and the result is the set of downloaded Web pages and their content.

These Web Pages are then available online to users for tagging, i.e. there are no constraints on who is able to access these pages; however, keeping track of *who tagged what* requires to differentiate between users, and hence, registration with the system, viz. logging in. The Web pages are accessible via the KrdWrd Add-on in combination[2] with the Web Services hosted on [@KRDW, Web Site].

[1]see [@URL] for details – but also [@ADDR].

[2]Indeed, the data is accessible with *any* browser – but the KrdWrd Add-on enhances the experience.

Users can tag new, alter or redisplay formerly tagged Web pages with the help of the KrdWrd Add-on. The KrdWrd Add-on builds upon and extends the functionality of the Firefox [@FF] browser and facilitates the visual tagging of Web pages, i.e. users are provided with an accurate Web page presentation and annotation utility in a typical browsing environment. Readily (or partly) tagged pages are directly sent back to the server for storage in the KrdWrd Corpus data pool and for further processing.

Updated or newly submitted tagging results are regularly merged, i.e. submitted results from different users for the same content are processed and compiled into a majority-driven uniform view. This automated process uses a *winner takes all strategy* and runs regularly on the server – without further ado. The *merged* content is stored in the KrdWrd data pool and hence, available for browsing, viewing, and analysis by the KrdWrd Add-on[2] and furthermore, it can be used as training data for Machine Learning algorithms.

1.2 Implementation Survey

The KrdWrd Infrastructure consists of several components that bring along the overall functionality of the system. They are run either on the KrdWrd Server or are part of the KrdWrd Add-on and hence, build upon and extend the functionality of the Firefox browser. The Server components are hosted on a Debian GNU/Linux [@DEB] powered machine. However, the requirements[3] are rather limited and many other standard linux - or linux-like - systems should easily suffice, and even other platforms should be able to host the system. Nevertheless, the KrdWrd Add-on strictly runs only as an extension of the Firefox browser, version 3[4].

Access to the system is given as HTTP Service hosted on `krdwrd.org`, an SSL-certified virtual host running on an Apache Web Server [@HTTP] accompanied by mailing services, a dedicated trac as Wiki and issue tracking system for software development (extended with a mailing extension), and subversion [@SVN] as version control system. The interfacing between the KrdWrd Add-on and the Web Server is done via CGI [@CGI] scripts, which itself are mostly written in the Python programming language [@PYTH].

2 Pre-Processing: Harvesting Web Pages

Generally, pre-processing is the first step to streamline external data for further processing in a customised data-processing pipeline, and in the KrdWrd System it constitutes harvesting data, i.e. grab Web pages off the web, convert them into UTF-8 encoding [@UNIC], make links on these pages relative [@W3ba], and compile them into a corpus that can be tagged by users.

2.1 URL List Generation

For downloading pages off the Web the KrdWrd System needs to be told *which* pages to grab. But because we are interested in a wide spectrum of layouts we need to scatter the URLs to be fetched over different web sites, i.e. we are not interested in having a small number of site URLs and then recursively grab these sites but we want a large number of URLs from different sites.

To this end we utilise the BootCaT toolkit[BB04] to construct an ad-hoc URL list: a set of seed terms is used for automated queries against a Web search engine, the top results for querying random combinations of the terms are downloaded and analysed, i.e. unigram term counts from all

[3]These include sed, awk, python, bash, subversion, XULRunner, wwwoffle, apache, R.
[4]But it could be converted into a self-contained XULRunner application.

retrieved Web pages are compared with the corresponding counts from a reference corpus. In the last step[5] multi-word terms are extracted and used a seed terms for the query process. However, we used the top results from these last multi-word queries as URL List.

en détail: We used the BootCaT installation of the Institute of Cognitive Science's Computational Linguistics group at the University of Osnabrück[@CL] – at the time of writing this was the initial version with the updates from February 2007.

The topic of the seed terms for the BootCaT procedure was "Nuremberg in the Middle Ages", the terms were: *history, coffee, salt, spices, trade road, toll, metal, silk, patrician, pirate, goods, merchant*, the Internet search engine BootCaT used was Yahoo![@YAHO], the reference corpus was the British National Corpus (BNC)[6], and the procedure resulted in 658 URLs from unique domains; note that we departed from the original BootCaT recipe and only allowed one URL per domain. This URL list was passed on to the KrdWrd Harvester – but, of course, *any* URL list can be feed to the Harvester.

The seed terms, the command sequence, and the URL list can be found at https://krdwrd.org/trac/browser/tags/harvest/canola.

2.2 The KrdWrd App: Harvesting Mode

The automated downloading of Web content is done by the KrdWrd App in harvesting mode, namely by feeding the App an URL list as input and have it then fetch and store downloaded content for further processing. Moreover this process resolves three significant concerns:

Enforce UTF-8 Character Encoding for grabbed documents – character encoding has been the cause for much hassle in data processing, and to eliminate it – or at least reduce it to a minimum – we transform *every* document into UTF-8 encoding [@UNIC] and make sure that successive processing steps are UTF-8 aware.

Change the <BASE> Element for grabbed documents (or insert one) [@W3ba, @ADDR] – for smooth integration into the KrdWrd system we change this attribute such that relative URIs are resolved relative to our system.

Surround Text with <KW> Elements in grabbed documents – these additional elements splits up text: when large amounts of text fall under a single node in the DOM tree, i.e. when the whole text can only be selected as a whole, these elements loosen this restriction but, on the other hand, do not affect the rendering of the Web page or other processing steps.

Finally, the System extracts the textual content of each page and only considers documents of certain text length as appropriate for further processing and discards all others. The rational is that *very* short and *very* long web pages rarely contain useful samples of interesting running text.

en détail: We used the previously generated URL list and fed it to the KrdWrd App in harvesting mode, which then retrieved Web pages via the KrdWrd Proxy (see 2.3) just as if someone operating a Firefox browser had viewed them. The textual length restriction was set to only allow for a

[5]Though, this loop can be repeated multiple times with unigram term counts until the corpus of retrieved Web pages reaches a certain size or matches other characteristics.

[6]The data was obtained under the terms of the BNC End User Licence. For information and licensing conditions relating to the BNC, please see the web site at http://www.natcorp.ox.ac.uk

decent amount of text, which we thought holds for documents consisting of 500 to 6,000 words[7]. Finally, we manually inspected the remaining grabbed pages for problems arising from limitations – and had to discard two files. Overall, the process resulted in 228 pages that were considered for further processing.

The currently used 'harvester' can be found at `https://krdwrd.org/trac/browser/trunk/src/app/harvest.sh`.

2.3 The KrdWrd Proxy

The KrdWrd Harvester and the KrdWrd Add-on make all Internet connections through the KrdWrd Proxy. This storage fills up with the harvested Web pages but also with all directly-linked material, which is included via absolute or relative links, or e.g. *generated* by scripts. Often, this 'additional' material is considered superfluous and therefore discarded; moreover, the non-textual content of Web pages is often stripped off – or the textual or structural content altered. See e.g. [@POTA, @CEan], or more generally [KB06, FNKdS07, EKS08].

Unfortunately, this renders it very difficult – or even impossible – to compare work in cases where one utilises data that is not available any more or only in an altered form. This is to say indeed, in the end we *also* want text but with different requirements of competing systems the base material must be pristine, i.e. the most 'natural' and the least modified version of data should be conserved. To this end, we utilise the World Wide Web Offline Explorer (wwwoffle)[@WOFF] as a proxy, which can be operated in two modes: *online* and *offline*.

wwwoffle Online Mode allows for caching of pages that are downloaded for later review, use with one or more external proxies, control over which pages cannot be accessed and which pages are not to be stored in the cache.

wwwoffle Offline Mode allows for use of normal browser to follow links, control which pages can be requested, and for non-cached access to Intranet servers.

wwwoffle generally allows for a searchable cache index with the addition of included programs, and viewable indexes sorted by name, date, server domain name, type of file. The configuration is done in a single configuration file, which can be accessed via an interactive web page to allow editing; user customisable error and information pages are also easily configurable.

During pre-processing the KrdWrd Online Proxy is used; it runs as a daemon and responds only to internal requests but material that is downloaded in online mode will be available for requests in offline mode.

The KrdWrd Offline Proxy runs as a daemon and responds to network requests from the Internet, it is publicly available[8], and can be accessed via `proxy.krdwrd.org:8080`. This proxy does not fetch new pages into the KrdWrd Cache, i.e. all Web page request coming from the client computer, e.g. from a user surfing the net with an installed and enabled KrdWrd Add-on, will be filtered and only requests for content that had previously been downloaded in online mode will be allowed. The offline mode is automatically configured by the KrdWrd Add-on .

[7]We used the linux `wc`[@WC] command, i.e. a word is a string of characters delimited by white space characters.
[8]with the exception that there exists a *dummy* login…

The Proxy data pool holds unmodified, re-loadable (near[9]) copies of all the Web pages from within the KrdWrd Corpus.

en détail: We set-up and configured two instances of wwwoffle on the KrdWrd Host; one publicly available, operating in offline mode and constituting the KrdWrd Offline Proxy, and one for use by the KrdWrd Harvester, operating in online mode and constituting the KrdWrd Online Proxy. The two instances are operational at the same time and they share the same data pool; this is easily possible and does not result in data inconsistencies because the offline proxy only reads data from the pool – it never writes data to the pool. Additionally, we configured the online proxy to *never* re-grab material, i.e. the first encounter of new content will be the one the systems keeps.

The currently used configuration can be found at `https://krdwrd.org/trac/browser/trunk/src/utils/wwwoffle`.

3 Manual Annotation: Classification of Web-Page Content by Human Annotators

The pre-processed data is now ready to be processed by annotators, and we will present the setting in which the annotated data, the foundation for the gold standard, was acquired.

The KrdWrd System incorporates the KrdWrd Add-on, an extension for the Firefox browser, which facilitates the visual tagging of Web pages. However, users also need to be told *what* to tag *how* – therefore, a refined version of the official 'CLEANEVAL: Guidelines for annotators' [@CEan] is provided, and – additionally – users are encouraged to work through a small tutorial to get acquainted with different aspects of how to apply the guidelines to real-world Web pages. The snag of finding people to actually put the system into use was kindly solved by the lecturers of the *Introduction to Computational Linguistics* class of 2008 from the Cognitive Science Program at the University of Osnabrück by means of an homework assignment for students.

3.1 The KrdWrd Add-on: An Annotation Platform

The KrdWrd Add-on receives data from the server, modifies the rendering of Web pages by highlighting selected text, supports the tagging of different parts of a page differently, and finally, sends an annotated page back to the server for storage and subsequent processing.

It extends the functionality of the Firefox browser with a status-bar menu where – beside some administrative tasks – the user may choose to put the current browser tab into *tracking mode*. In this mode pre-defined colour coded tags are integrated into the familiar view of a Web page A) to highlight the part of the page where the mouse is hovering over and thereby, is subject to tagging, and B) to highlight the already tagged parts of the page.

The annotation process is straightforward (cf. figure 1 for a partly annotated page):

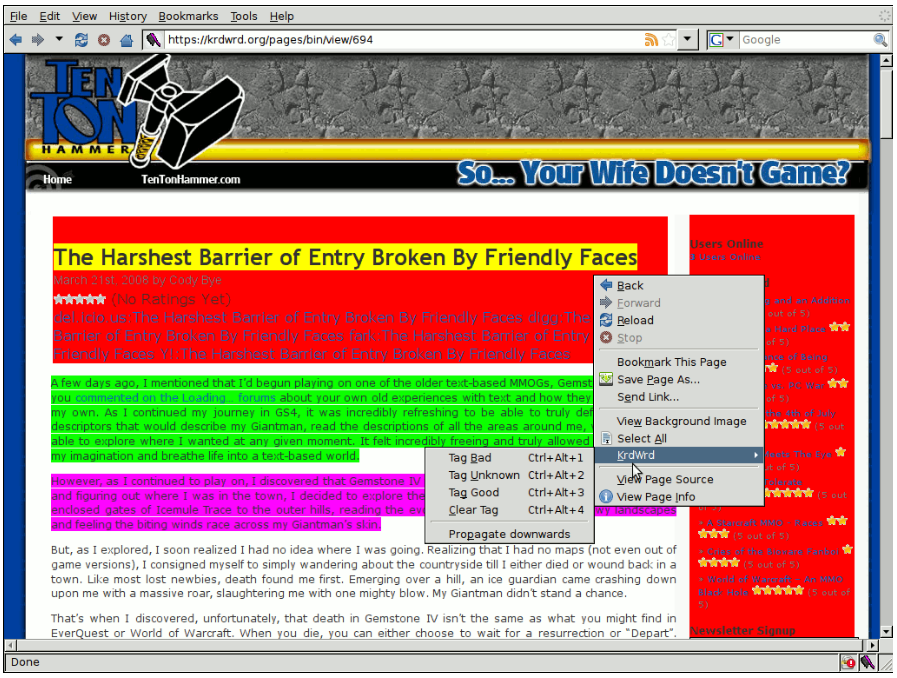
1. Users move the mouse over the Web page and the block of text *under* the mouse pointer is highlighted (Sometimes this block will be rather small, sometimes it may cover large portions of text),

2. Users assign tags to the highlighted blocks by either using assigned keyboard shortcuts or via entries in the context menu (Afterwards, these blocks stay coloured in the respective colours of the assigned tags),

[9]Dynamically generated links are challenging and may lead to missing content.

3. Users submit the page, i.e. the Web page *and* the incorporated tags are transfered to the server – this is done by pressing a shortcut or via an entry in the status-bar menu (The tagged page, or a partly tagged page, for that matter, can be re-submitted to the server), and

4. The KrdWrd System serves a new, untagged page for tagging[10].



The Harshest Barrier of Entry Broken By Friendly Faces

March 21st, 2008 by Cody Bye

(No Ratings Yet)

del.icio.us:The Harshest Barrier of Entry Broken By Friendly Faces digg:The Barrier of Entry Broken By Friendly Faces fark:The Harshest Barrier of Entry Friendly Faces Y!:The Harshest Barrier of Entry Broken By Friendly Faces

A few days ago, I mentioned that I'd begun playing on one of the older text-based MMOGs, Gemst you commented on the Loading... forums about your own old experiences with text and how they my own. As I continued my journey in GS4, it was incredibly refreshing to be able to truly def descriptors that would describe my Giantman, read the descriptions of all the areas around me, able to explore where I wanted at any given moment. It felt incredibly freeing and truly allowed my imagination and breathe life into a text-based world.

However, as I continued to play on, I discovered that Gemstone IV and figuring out where I was in the town, I decided to explore the enclosed gates of Icemule Trace to the outer hills, reading the ev wy landscapes and feeling the biting winds race across my Giantman's skin.

But, as I explored, I soon realized I had no idea where I was going. Realizing that I had no maps (not even out of game versions), I consigned myself to simply wandering about the countryside till I either died or wound back in a town. Like most lost newbies, death found me first. Emerging over a hill, an ice guardian came crashing down upon me with a massive roar, slaughtering me with one mighty blow. My Giantman didn't stand a chance.

That's when I discovered, unfortunately, that death in Gemstone IV isn't the same as what you might find in EverQuest or World of Warcraft. When you die, you can either choose to wait for a resurrection or "Depart".

Figure 1: We used the lovely colour fuchsia to highlight the part of the page where the mouse is hovering over, and the colours red, yellow, and green[11] for the already tagged parts, where red corresponded to *Bad*, yellow to *Unknown*, and green to *Good* (cf. 3.2 for details).

Furthermore, the KrdWrd Add-on is accompanied by a manual [Krd]. It explains how to install the Add-on, get started with tagging pages, how to actually tag them, i.e. it includes the annotation guidelines, and also gives some tips & tricks on common tasks and problems.

The Add-on is available from `https://krdwrd.org/trac/wiki/AddOn`.

3.2 The KrdWrd Annotation Guidelines

The KrdWrd Annotation Guidelines specify which tag should be assigned to particular kinds of text. We used the CleanEval (CE) annotation guidelines as a start (cf. [@CEan]), and made a few substantial changes however, because we realised that there were several cases in which their guidelines were insufficient.

The most important change we made was the addition of a third tag 'uncertain' whereas originally, only the two tags 'good' and 'bad' were available. It had soon become apparent that

[10]This new page is randomly selected among the set of pages with the lowest count of aggregated submissions per user, i.e. at large, the submissions will be evenly distributed over the corpus – but cf. 3

[11]fuchsia – there is a short story behind this color: `https://krdwrd.org/trac/wiki/KrdWrd` – red, yellow, and green, respectively

on some Web pages there were passages that we did not want to be part of a corpus (i.e. that we did not want to tag 'good') but that we did not want to throw out altogether either (i.e. tag them as 'bad'). We also decided to tag all captions as 'uncertain'.

Another rationale behind this introduction of this third tag was that we might want to process this data at a later stage. Also note that in [SMP08] other CE participants also used a three-element tag set.

We adopted the following guidelines from the CE contest, and all of these items were supposed to be tagged 'bad':

- Navigation information
- Copyright notices and other legal information
- Standard header, footer, and template material that are repeated across (a subset of) the pages of the same site

We modified the requirement to clean Web pages of internal and external link lists and of advertisement slightly: The KrdWrd Guidelines state that *all* (hyper-)links that are *not* part of the text are supposed to be tagged as 'bad'. This, of course, includes link lists of various kinds, but preserves links that are grammatically embedded in 'good' text. We also restricted ourselves as to discard advertisement from *external* sites only. Some of the pages were pages about certain products, i.e. advertisement, but we did not want to exclude these texts (if they fulfilled our requirements for 'good' text, as defined below).

The two sorts of text, we did not exclude specifically (as the CE guidelines did), were Web-spam, such as automated postings by spammer or blogger, and cited passages. Instead, we required 'good' text to consist of complete and grammatical English sentences that did not contain 'non-words' such as file names. That way, we filter out automatically generated text *only if* it is not grammatical or does not make up complete sentences, and keep text that can be useful for information extraction with statistical models.

Our refined annotation guidelines still leave some small room for uncertainties (but probably *all* such guidelines suffer from this problem). We are optimistic, however, that they are a clear improvement over the original CE guidelines and that our Web corpus will only contain complete and grammatical English sentences that contain 'normal' words only.

The annotation guidelines are available from `https://krdwrd.org/manual/html/`.

3.3 The KrdWrd Tutorial: Training for the Annotators

For initial practice, we developed an interactive tutorial that can be completed online (as feature of an installed Add-on).

The interactive tutorial can be accessed from the status bar by clicking 'Start Tutorial', and is designed to practice the annotation process itself and to learn how to use the three different tags correctly. Eleven sample pages are displayed one after another, ranging from easy to difficult (these are the same samples as in the 'How to Tag Pages' section of the manual).

The user is asked to tag the displayed pages according to the guidelines presented in the manual. We inserted a validation step between the clicking of 'Submit' and the presentation of the next page, giving the user feedback on whether or not she used the tags correctly. Passages that are tagged in accordance with our annotations are displayed in a light-coloured version of the original tag, i.e. text correctly tagged as 'bad' will be light-red, 'good' text will be light-green, and text that was tagged correctly as ‘uncertain’ will be light-yellow. The passages with *differing* annotations are displayed in the colour in which they should have been tagged, using the normal colours, i.e. saturated red, green, and yellow. After clicking ‘Next Page’ on the right top of the screen, the next page will be shown.

If a user should decide to quit the interactive tutorial before having tagged all eleven sample pages, the next time she opens the tutorial, it will begin with the first of the pages that have not been tagged, yet. And should a user want to start the tutorial from the beginning, she can delete previous annotations via ‘My Stats’ in the status bar. Then, the next time the tutorial is opened it will start from the very beginning. By pressing ‘Start Tutorial’ in the status bar during the practice and *before* the submission of the current page, that same page will be displayed again, un-annotated. When using ‘Start Tutorial’ *after* a page’s submission and before clicking ‘Next Page’ in the notification box at the top, the next page of the tutorial will be shown.

As stated above, it is our goal that the interactive tutorial will help users getting used to the annotation process, and we are also optimistic that it helps understanding and correctly applying the tagging guidelines as presented in the manual.

3.4 The KrdWrd Assignment: A Competitive Shared Annotation Task

Finally, our efforts were incorporated into an assignment for the class ‘Introduction to Computational Linguistics’ where – from a maximum number of 100 students – 68 completed the assignment, i.e. their effort was worth at least 50% of the assignment’s total regular credits. The assignment was handed out 7, July, was due 18, July 2008, and consisted of two exercises:

1. The first task was to complete the interactive online tutorial, i.e. the students had to go through the eleven sample pages, annotate them, and – ideally – think about the feedback. This task was worth 20% of the credits.

2. The second task was to tag pages from our assembled corpus; 15 tagged pages were worth 80% of the credits and 10 additional pages were worth an extra that was counted towards the credits of all other homework assignments, i.e. students could make up for ‘lost’ credits[12].

4 The Gold Standard: Compilation and Analysis of manually annotated Data

The data for the gold standard was collected via the KrdWrd Add-on (cf. 3.1) as an homework assignment (cf. 3.4) for a Computational Linguistics class, which is a second year undergraduate Cognitive Science class at the University of Osnabrück. The Annotators were introduced to the KrdWrd Annotation Guidelines (cf. 3.2) by means of the KrdWrd Tutorial (cf. 3.3), and were supposed to work independently (e.g. from their home PC) though, could have sat near each other. However, we did take precautions against naïve copying by enforcing authentication for the users with their student accounts, hiding other users’ results, and serving random pages for tagging – thus, even if students were exchanging information it could rather have been about the assignment and the tagging in general than a specific Web site in particular.

[12]As a matter of fact, 43 students received the total of 100% regular credits + 100% extra credits.

4.1 Initial Observations

From 100 student subscribed to the class 69 installed the Add-on and submitted at least one page (not necessarily a tagged one, though…). This was also about the ratio of students who took the final exam for this course hence, we can say that almost every students seriously interested in finishing this class also took the homework assignment.

The majority of submissions came within the last 4 days of the period of time they were granted to finish the assignment – with a major peak at the last day; which, according to all we know, is quite common. This has probably also led to only very few people making use of the re-submit feature, i.e. continuing or modifying an already submitted page.

The possibility to interact with the KrdWrd Team, e.g. to solve installation problems, or to exchange information via an e-Mail list we had set up for this purpose was rarely used (cf. `https://krdwrd.org/trac/mail/threads`). The few reported problems however, led to some beneficial improvements of the documentation.

Our initial Data Set (before further clean-up): 228 Web pages, consisting of almost 440,000 words and over 2.6 million characters, were independently processed by 69 users who submitted 1767 results (re-submits for a page counted only once), which is an average of 7.75 submissions per page.

4.2 The KrdWrd App: Annotations Merging Mode

The KrdWrd App in merging mode compares the initially grabed *master* with the user submitted results and, for every text node in the DOM tree, computes a majority vote and assigns this as the gold standard tag to the node of a newly created document.

The process is carried out offline on the server: the input is one URL of a master document and the URLs of the respective user submitted results. After reading all documents the DOM trees of all documents are traversed top-down and tags along the traversal are propagated further down as long as, no more specific tag is encountered, i.e. a tag can not overwrite another one further down the path but is *pushed down* as far as possible (cf. figure 2 for an illustration). At the end of each path in the tree the assigned tags are counted. After having traversed all documents a sanity check is carried out[13] [14] namely, are there documents which still have unseen nodes, or are there documents which had less nodes than the master document? In either case, these submissions are discarded from further processing.

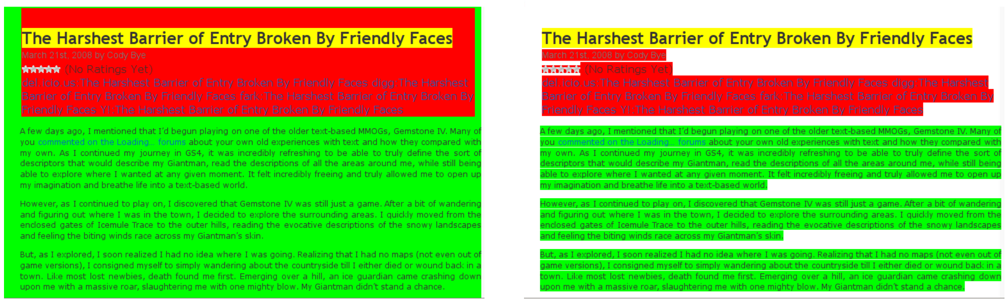
The remaining submissions are taken into account for the majority vote on each node of the master document. Another document is generated, which includes the resulting tags.

4.3 Merge Analysis

Before we started to analyse the results of the merging process we excluded the results of one user who had only submitted one page. Then, the merging process revealed the following problematic cases (usually by rejecting user results on grounds of the sanity check): 2 pages with no results

[13]We also implemented another sanity check namely, to check whether the textual content in the nodes is identical but dropped this condition – mainly because the few encounters were false positives and it had negative impact on performance as well.

[14]The overall handling of JavaScript is not satisfactory. To address the diversions between submits occurring after dynamic client-side JavaScript execution on different clients, the Add-on could hook into the node creation and clone processes. They could be suppressed entirely or newly created nodes could grow a special id tag to help identifying them later.



The Harshest Barrier of Entry Broken By Friendly Faces

March 21st, 2008 by Cody Bye

(No Ratings Yet)

del.icio.us:The Harshest Barrier of Entry Broken By Friendly Faces digg:The Harshest Barrier of Entry Broken By Friendly Faces fark:The Harshest Barrier of Entry Broken By Friendly Faces Y!:The Harshest Barrier of Entry Broken By Friendly Faces

A few days ago, I mentioned that I'd begun playing on one of the older text-based MMOGs, Gemstone IV. Many of you commented on the Loading... forums about your own old experiences with text and how they compared with my own. As I continued my journey in GS4, it was incredibly refreshing to be able to truly define the sort of descriptors that would describe my Giantman, read the descriptions of all the areas around me, while still being able to explore where I wanted at any given moment. It felt incredibly freeing and truly allowed me to open up my imagination and breathe life into a text-based world.

However, as I continued to play on, I discovered that Gemstone IV was still just a game. After a bit of wandering and figuring out where I was in the town, I decided to explore the surrounding areas. I quickly moved from the enclosed gates of Icemule Trace to the outer hills, reading the evocative descriptions of the snowy landscapes and feeling the biting winds race across my Giantman's skin.

But, as I explored, I soon realized I had no idea where I was going. Realizing that I had no maps (not even out of game versions), I consigned myself to simply wandering about the countryside till I either died or wound back in a town. Like most lost newbies, death found me first. Emerging over a hill, an ice guardian came crashing down upon me with a massive roar, slaughtering me with one mighty blow. My Giantman didn't stand a chance.

The Harshest Barrier of Entry Broken By Friendly Faces

March 21st, 2008 by Cody Bye

(No Ratings Yet)

del.icio.us:The Harshest Barrier of Entry Broken By Friendly Faces digg:The Harshest Barrier of Entry Broken By Friendly Faces fark:The Harshest Barrier of Entry Broken By Friendly Faces Y!:The Harshest Barrier of Entry Broken By Friendly Faces

A few days ago, I mentioned that I'd begun playing on one of the older text-based MMOGs, Gemstone IV. Many of you commented on the Loading... forums about your own old experiences with text and how they compared with my own. As I continued my journey in GS4, it was incredibly refreshing to be able to truly define the sort of descriptors that would describe my Giantman, read the descriptions of all the areas around me, while still being able to explore where I wanted at any given moment. It felt incredibly freeing and truly allowed me to open up my imagination and breathe life into a text-based world.

However, as I continued to play on, I discovered that Gemstone IV was still just a game. After a bit of wandering and figuring out where I was in the town, I decided to explore the surrounding areas. I quickly moved from the enclosed gates of Icemule Trace to the outer hills, reading the evocative descriptions of the snowy landscapes and feeling the biting winds race across my Giantman's skin.

But, as I explored, I soon realized I had no idea where I was going. Realizing that I had no maps (not even out of game versions), I consigned myself to simply wandering about the countryside till I either died or wound back in a town. Like most lost newbies, death found me first. Emerging over a hill, an ice guardian came crashing down upon me with a massive roar, slaughtering me with one mighty blow. My Giantman didn't stand a chance.

Figure 2: On the left is the un-propagated page with major parts having been tagged green and red. On the right is the propagated version where the green has been pushed down into the text nodes; the same holds for red but note that the heading in yellow has not been overwritten.

left to merge, 3 pages with only one result to merge, 2 more pages with only two result to merge, 1 page with four results to merge, and 1 page that could not be merged due to an error in our application[15]. We also excluded all these cases from further processing (cf. figure 3).

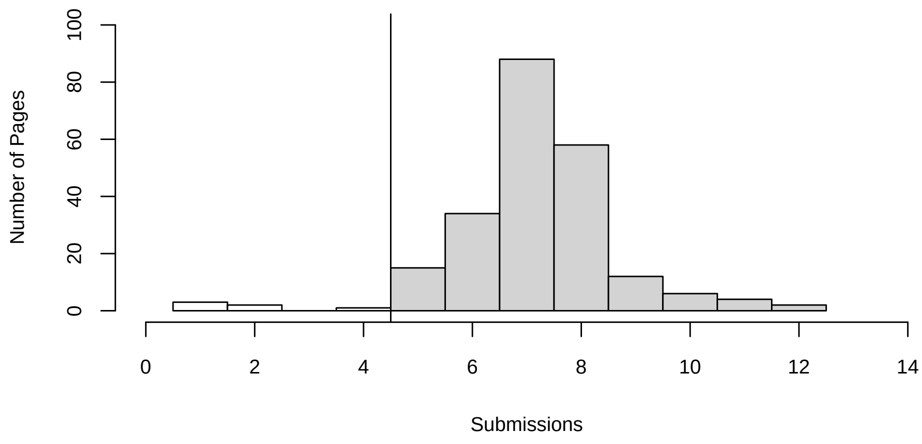


Figure 3: Number of Pages with x Submissions - the dividing line at *5 Submissions* shows the cut-off, i.e. pages with less then 5 Submissions were excluded from further processing. The observant reader may notice that we said the annotations were evenly distributed: this is the case, now. We had not turned on this feature when we started collecting the data, however.

We continued to do a plausibility check for the submitted results: we computed a *tag-bias* for each user, where we compared each user's tendency to chose a tag for a node with the actual winning tags for nodes. This computation revealed 4 cases in which users showed strong biases towards certain tags[16]. We also excluded all results from these users.

[15]We fixed the error but this rendered the submitted pages unusable – newly submitted pages will be mergable.

[16]Manual inspection of these cases showed that the users obviously only wanted to raise their tagged-pages count

The resulting and final Data Set: 219 Web pages, consisting of more than 420,000 words and over 2.5 million characters, were independently processed by 64 users who submitted 1595 results (re-submits for a page counted only once), which is an average of 7.28 submissions per page.

We continued our analyses of this new data at hand, and looked into the timestamps we collected for the submissions: therefore, we summed up all the deltas between two submissions for each user and calculated the duration each user *saw* a single page; then, we computed a reasonable upper bound for how log a submit action might take, i.e. the hypothesis was that page-view times longer than a certain amount of time were actually breaks. To this end, we detected outliers[17] and discarded *all* respective submissions (the calculated[R D08] result was 700s).

The calculated time data suggests that:

- the majority of users spent between 56 and 88 minutes on the assignment with an average of 71 minutes (cf. figure 4 for details),
- average per-page annotation time drops below three minutes (cf. figure 5), and
- the first pages after the tutorial are still more challenging than later ones (cf. 6).

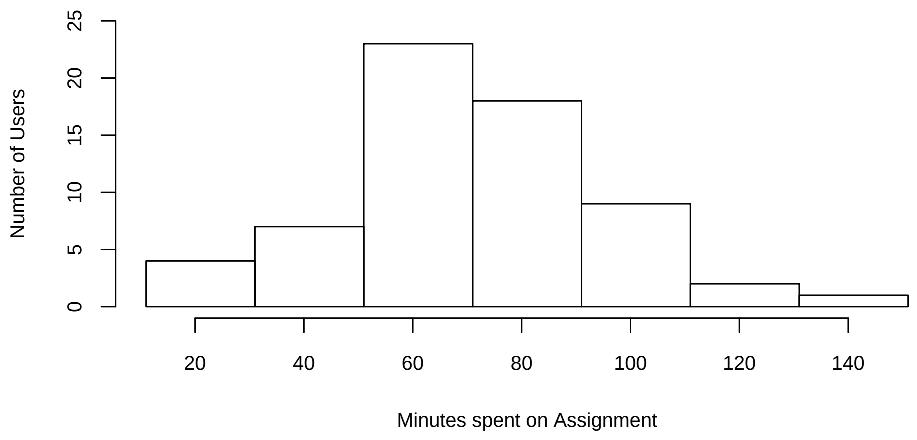


Figure 4: Time in Minutes spent by y Users on the Assignment, i.e. how much Time did a User interact with the Add-on to tag her share of the Canola Corpus.

For the overall inter-coder agreement of the remaining submissions we calculated Fleiss's multi-π as layed out in [AP08]: for each Web page the remaining submissions were set as coders, and the tagged DOM nodes as items – the three categories were fixed. This resulted in an average inter-coder agreement over all pages of 0.85 (cf. 8), which we think is – at least – substantial. Considering that these submissions were the basis for the merge process we believe that the Canola

and therefore, just tagged very few nodes – typically high up in the DOM tree – which were then propagated downwards.

[17]This is quite standard: x values outside the range $Q1 - 1.5 * IQR < x < 1.5 * IQR + Q3$ were considered outliers.

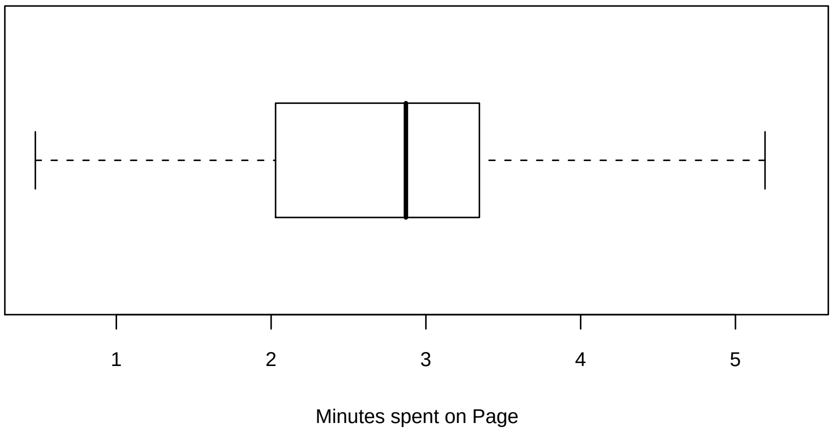


Figure 5: Minutes spent on a single Page accross all annotations of the Canola corpus.

Gold Standard Corpus is a solid basis for further processing. Furthermore, this metric could be used for comparison of cleaning results in general – maybe normalised for the number of words or characters per DOM node.

Remark: we looked into the pages at the lower end of the agreement spectrum and found that they tended to be quite long and were often discussion forum pages, i.e. with many alterations in the tags that were to assign. Given that similar shorter pages achieved better results, it seems that even our already quite low boundary of 6,000 words per page resulted in pages that were frustrating to process.

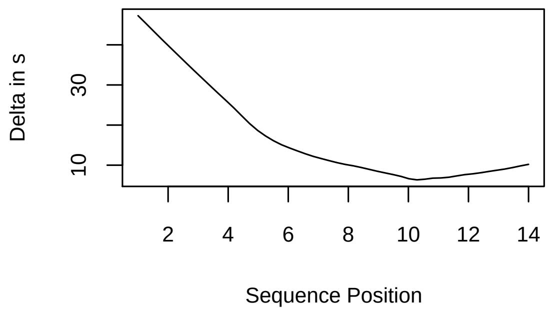


Figure 6: Smoothened average of differences in seconds between annotation times of all users at Position x in their specific sequences of Web Pages and the mean of all other users who processed identical pages at a later time in their respective sequences.

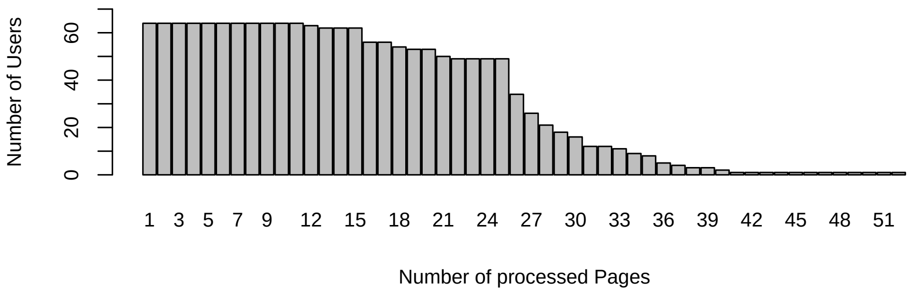


Figure 7: Aggregated counts for the Number of Users who processed *at least* x Number of Pages. Note the two steps at 15 and 25 pages, which correspond to the obligatory and the optional number of pages in the assignment. Also note that there were quite many students who went far beyond the requirement for the assignment.

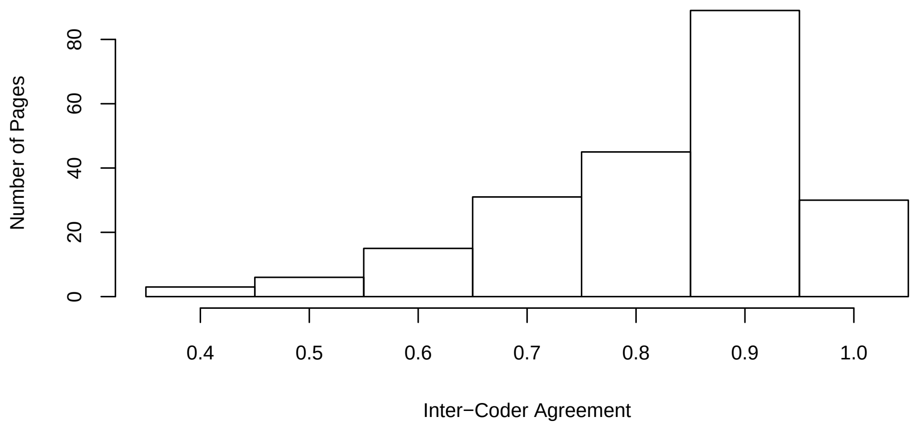


Figure 8: Inter-coder agreement between submissions for pages over the Canola corpus.

References

The Web sites referred to below were last accessed on March 20, 2009. In case of unavailability at a later time, we recommend visiting the Internet Archive.

[AP08] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008. Available from: http://www.mitpressjournals.org/doi/abs/10.1162/coli.07-034-R2.

[BB04] Marco Baroni and Silvia Bernardini. BootCaT: Bootstrapping corpora and terms from the web. pages 1313–1316, Lisbon, Portugal, May 2004. Available from: http://sslmit.unibo.it/~baroni/publications/lrec2004/bootcat_lrec_2004.pdf.

[EKS08] Stefan Evert, Adam Kilgarriff, and Serge Sharoff, editors. *Can we beat Google? (WAC4 - 2008) – Proceedings of the 4th web as corpus workshop*, 06 2008. Available from: http://webascorpus.sourceforge.net/download/WAC4_2008_Proceedings.pdf.

[FNKdS07] Cédrick Fairon, Hubert Naets, Adam Kilgarriff, and Gilles-Maurice de Schryver, editors. *Building and Exploring Web Corpora (WAC3 - 2007) – Proceedings of the 3rd web as corpus workshop, incorporating CLEANEVAL*, Louvain-la-Neuve, July 2007. Presses universitaires de Louvain.

[KB06] Adam Kilgarriff and Marco Baroni, editors. *11th Conference of the European Chapter of the Association for Computational Linguistics – Proceedings of the 2nd International Workshop on Web as Corpus*, Trento, Italy, April 2006. Available from: http://www.aclweb.org/anthology-new/W/W06/W06-1700.pdf.

[Krd] The KrdWrd Project. *Add-on Manual*. Available from: https://krdwrd.org/manual/manual.pdf. Online Version at https://krdwrd.org/manual/html/.

[R D08] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. Available from: http://www.R-project.org. ISBN 3-900051-07-0.

[SMP08] Miroslav Spousta, Michal Marek, and Pavel Pecina. Victor: the web-page cleaning tool. In Evert et al. [EKS08]. Available from: http://webascorpus.sourceforge.net/download/WAC4_2008_Proceedings.pdf.

[SS09] Johannes M. Steger and Egon W. Stemle. KrdWrd Architecture for Unified Processing of Web Content. In *Proceedings of the Fifth Web as Corpus Workshop (WAC5)*, Donostia-San Sebastian, Basque Country, 2009.

[@ADDR] W3C. Naming and addressing: URIs, URLs, . . . [online, cited 032009]. Available from: http://www.w3.org/Addressing/.

[@CEan] Marco Baroni and Serge Sharoff. CLEANEVAL: Guidelines for annotators [online, cited 032009]. Available from: http://cleaneval.sigwac.org.uk/annotation_guidelines.html.

[@CGI] The Common Gateway Interface (CGI) – a standard for external gateway programs to interface with information servers such as http servers [online, cited 032009]. Available from: http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

[@CL] The computational linguistics group of the Institute of Cognitive Science at the University of Osnabrück [online, cited 032009]. Available from: http://www.ikw.uni-osnabrueck.de/CL/.

[@DEB] Debian GNU/Linux: The free operating system for your computer [online, cited 032009]. Available from: http://www.debian.org/.

[@FF] Firefox: The browser that has it all [online, cited 032009]. Available from: http://www.mozilla.com/firefox/.

[@HTTP] The apache HTTP server project [online, cited 032009]. Available from: http://httpd.apache.org/.

[@KRDW] Johannes M. Steger and Egon W. Stemle. The KrdWrd project web site [online, cited 032009]. Available from: https://krdwrd.org/.

[@POTA] A perl module intended to perform “boilerplate” stripping and other forms of filtering [online, cited 032009]. Available from: http://sslmitdev-online.sslmit.unibo.it/wac/post_processing.php.

[@PYTH] Python: An interpreted, interactive, object-oriented programming language [online, cited 032009]. Available from: `http://www.python.org/`.

[@SVN] An open source version control system [online, cited 032009]. Available from: `http://subversion.tigris.org/`.

[@UNIC] Unicode home page [online, cited 032009]. Available from: `http://www.unicode.org/`.

[@URL] URI working Group. Uniform resource locators: A syntax for the expression of access information of objects on the network [online, cited 032009]. Available from: `http://www.w3.org/Addressing/URL/url-spec.txt`.

[@W3ba] HTML 4.01 specification – path information: the BASE element [online, cited 032009]. Available from: `http://www.w3.org/TR/html401/struct/links.html#h-12.4`.

[@WC] The wc command [online, cited 032009]. Available from: `http://www.bellevuelinux.org/wc.html`.

[@WOFF] Andrew M. Bishop. A simple proxy server with special features for use with dial-up internet links [online, cited 032009]. Available from: `http://gedanken.demon.co.uk/wwwoffle/`.

[@YAHO] The Yahoo! internet search engine [online, cited 032009]. Available from: `http://www.yahoo.com`.