Implementing a BNC-Compare-able Web Corpus

William H. Fletcher[1]
United States Naval Academy

Abstract

This paper details the author's plans for and progress with compiling and analyzing a new gigaword English corpus from the web to complement his BNC-based online database "Phrases in English". This new corpus represents the principal English-speaking countries in proportion to their population and will be linguistically annotated with the CLAWS4 tagger using a PoS-tagset comparable to those of the BNC and ANC. Parallel processing on multiple PCs will facilitate reaching the targeted size. This corpus will continue to grow dynamically in response to actual user queries to the author's various web as corpus interfaces, but "snapshots" of each generation of the corpus will be preserved to ensure replicability of results. This report on work in progress will inspire discussion of the underlying concepts and suggestions for improvement.

Keywords : web corpus, corpus annotation, BNC, Phrases in English

1. Concept and Background

1.1. Concept

WebAsCorpus.org (WaC) will provide an exploratory environment to investigate and discover patterns of English words and phrases similar to my BNC-based "Phrases in English" (PIE) site (http://pie.usna.edu and http://phrasesinenglish.org). PIE permits search by word-form, lemma and part-of-speech-tag (PoS), with full support for wildcards and regular expressions. There are database tables of *n*-grams for values of *n* in the range 1-8. Search results display words and phrases matching the user's query along with frequency data and PoS tags; datasets can also be downloaded as tab-separated value text files for direct import into a database or spreadsheet. Users can view concordances of a matching word-form by clicking on it in the search results. Other search interfaces allow one to explore PoS-*n*-grams, character-*n*-grams and phrase-frames, *i.e.*, sets of variants of an *n*-gram identical except for a single word. Drill-down queries show variant phrases of different values of *n* in which the user's search term appears.

---

[1] Language Studies Department, United States Naval Academy, fletcher@usna.edu. The research described in this paper was funded in part by the Naval Academy Research Council.

Initially WaC is aiming for quantity: to achieve the critical mass necessary for investigating the phraseology of contemporary and emerging English it will recreate most of the functionality of PIE with a corpus of Web documents at least ten times larger than the BNC. It will be fully *compare-able with* if not truly *comparable to* the BNC. Since it will not necessarily be a representative sample of English as a whole or even of the language of the Web, frequency data may be only a rough guide to predominant actual usage. In addition, a significant amount of noise may persist in the corpus. Nevertheless, it should excel as an environment for discovering, studying and documenting linguistic innovation and evolving trends in language use. In later iterations it will be refined and made more balanced using genre detection and boilerplate removal techniques developed by others (*e.g.* CLEANEVAL).[2]

A key element of the WaC concept is dynamic expansion weighted toward actual user needs and interests. WaC already provides a query interface to generate concordances directly via Microsoft's Live Search (LS, http://live.com, formerly known as MSN Search) application programming interface (API).[3] As matching documents are retrieved and excerpted, both the original source HTML and the plain-text document derived from it are saved for later analysis and incorporation into the database. In addition, query terms from user searches with WaC, PIE and KWiCFinder serve as seeds to collect new documents for the corpus as described below.

1.2. Precursors

With the AltaVista's (AV) debut in the final days of 1995 I recognized the Web's potential both for consultation directly as a corpus and for machine-readable texts to compile offline corpora. In 1997 I piloted KWiCFinder (Fletcher 2007), a Windows concordancer which queries AV and downloads and excerpts matching documents, optionally saving them locally for further analysis. With KWiCFinder I have compiled several Web corpora ranging in size up to 180 MW (Fletcher 2004b; 2007). The last of these can be queried on the WaC website, with live generation of Web concordances based on queries to LS.

At TALC 2002 I proposed establishing a Web Corpus Archive (WCA) and a Search Engine for Applied Linguists (SEAL). In the conference publication I detail my vision and compare it to alternatives (Fletcher 2004a : 285-291). In a later presentation (Fletcher 2005) I update the concept and sketch both the path and the obstacles to a

---

[2] In view of the ever-dropping cost of storage (one of my hosting providers now allows up to 1 TB webspace for a few dollars a month!) it may be useful to have both "clean" and "dirty" versions of this web corpus, the latter consisting of all raw data as downloaded. Compare *e.g.* the number of results for a search for **bot(s)* in the clean (249) and dirty (425) 1-gram data from my 2006 web corpus (http://webascorpus.org/searchwc.html), especially in the "long tail" of hapaxes.

[3] An API provides a protocol to query the search engine directly, specify the information desired, and receive the results in form easily parsed by a program.

linguistic search engine (SE). When Paul Rayson (UCREL, Lancaster University, UK) proposed an annotated Web as corpus project to implement a similar concept (Rayson *et al.* 2006), I joined enthusiastically in the effort. Since that project was not funded, I now have revived the idea of carrying out a similar plan alone.

2. Implementing WaC

2.1. Data collection

2.1.1. Source of texts

Compilers of large Web corpora (*e.g.* Baroni and Kilgarriff: 2006) have frequently relied on crawling techniques: after identifying a number of entry pages via SE queries, their software retrieves and saves these pages, then extracts and follows links from them, repeating the process until enough useful pages are downloaded. With today's inexpensive bandwidth, storage and processing power, a massive text collection can be accumulated in a matter of hours with this discovery strategy.

In contrast, WaC relies on LS to both find and deliver candidate documents. Several factors played a role in this decision. First, of the SEs which provide free APIs to developers, LS is the most generous by far: it allows 10,000 queries per application id (AppID) *per IP address* per day. In other words, running the query software on multiple machines *multiplies* the daily quota. In contrast, Google is phasing out its API and license conditions that would support such software, and it permits only 1000 queries daily *per application* for those who still have legacy keys. While Yahoo (which has taken over and replaced the AltaVista SE) may agree to grant more than 1000 queries per day upon request, permission is by no means automatic. All three SEs limit search result sets to 1000 items per query.

Moreover, LS provides high-quality search results, with relatively few pages from link farms or "scraper sites", which repeat content from or link to other pages merely for advertising revenue.[4] In addition, one can tweak the search results ranking by adjusting on a scale of 1-100 the relative weight of parameters like exactness of match to the search terms (*e.g.* to block automatic stemming), page popularity and "freshness", *i.e.* how recently a webpage was created or updated. In effect these adjustable parameters permit a single set of query terms to match many times the nominal limit of 1000 items. This feature is particularly valuable to deemphasize page popularity to avoid the bias toward commercial sites so evident on Google.

LS also supports search by location, *i.e.* by country or even latitude and longitude. This is more precise than search by the server's IP address or domain, especially for

[4] Whether this is due to better spam detection or smaller index size overall is unclear (F. McCown, p.c.).

location-independent domains like .COM, .NET and .BIZ.[5] In addition, studies have shown LS to be more responsive to changes on the Web: there is faster turnover in the top hits returned for a given query (McCown and Nelson 2007 a and b) than with Google or Yahoo!, and documents in the cache tend to be "fresher", *i.e.* updated more frequently.

Finally, the LS cache provides quick, reliable access to the original texts. In documents retrieved from the cache, LS generally detects the character set encoding accurately and converts it to UTF-8, thereby eliminating a potential source of variability and errors.

LS also converts Adobe Acrobat PDF documents to HTML which closely reflects the formatting of the original. While PDF does not encode the logical formatting of the text (headings, paragraphs, captions etc.), the structure can be inferred from the converted HTML. Each style within a PDF document is assigned to a CSS class; since the body of the text belongs to the most frequent class, it is easy to distinguish major content from headers and footers. Each line or column is encoded as a <SPAN> element with absolute top and left positioning, which facilitates recognition of new paragraphs and columns (greater left offset for indenting and right-hand column, greater top offset for non-indented paragraph breaks). One problem that plagues all PDF to text converters persists: spaces are occasionally dropped or inserted between or within words, an artifact of the PDF encoding.

LS' API provides direct links to the cache, and the site responds rapidly and at a high transfer rate, permitting very efficient data collection without delays, redirections or dead links.[6] As part of the U.S. Department of Defense my institution is required to block traffic to and from a large number of potentially "hostile" IP addresses (which has even included the Université catholique de Louvain); by retrieving documents from LS' cache I can circumvent these restrictions. One final benefit is that a HEAD request to LS' cache server always returns the CONTENT-LENGTH, which allows one to skip documents too large or too small to be interesting.[7]

Of course, LS does not offer a perfect solution. Apparently its crawls are shallower (*i.e.* include fewer pages from a given site) than those of Google or Yahoo, and the overall size of the index is smaller. Like those of its competitors, LS' hit counts can vary significantly across multiple instances of the same query, even reporting 0 hits for

---

[5] The location appears to be determined by address or other geographical information on the webpage or elsewhere on the website. Consequently sites lacking such information cannot be classified by country and are excluded from country-specific queries.

[6] Intermittently the API search interface was out of synchronization with the cache, so some searches had to be repeated.

[7] Owing to the increasing prevalence of dynamically-generated content, servers now more frequently transfer documents in a number of "chunks". Since each chunk reports only its own length, one must download an entire document to know its total length.

a very frequent phrase; the hit count typically stabilizes after a couple of repetitions of a given query. Sporadically, especially when traffic is heavy, LS rejects an otherwise valid AppID, forcing one to query the Web user interface (WUI) and "scrape" (*i.e.* parse and extract the links from) the results page.[8] Finally, there is a nuisance factor: LS returns at most 50 results per query, but usually the result sets fall short by several documents, necessitating re-querying.

2.1.2. Search strategy

Several approaches to selecting search terms to "seed" data collection have been discussed in the literature (Baroni and Kilgarriff 2006; Sharoff 2005 and 2006). These seed terms can have a crucial bearing on the composition of the resulting corpus (Ueyama 2006). For my earliest Web corpus the 20 most frequent words in the BNC were chosen on the assumption that they would have the least direct influence on the content (Fletcher 2004b:194). Searching for specific content words becomes a self-fulfilling prophecy, skewing the results toward pages in which those words are prominent. For example, a search for morphological variants of *hone* to explore the metaphoric use (*hone one's skills* etc.) leads almost exclusively to websites offering hones for brake cylinders or stone. This danger even lurks in function words: a search for *well* matches primarily pages with pumps and well-drilling equipment and services.

For my 2006 Web corpus I used a different strategy to ensure variety across the semantic spectrum in the hits: I chose "prototypical example" words from each sub-classification of UCREL's USAS semantic categories (Archer, Wilson and Rayson 2002:3) which are used and spelled the same in all Anglophone countries and searched on alternate wordforms (e.g. *result OR results OR resulted OR resulting*). For WaC I am using these content word search terms in conjunction with function words specifying a low value for LS' "exact match" parameter in order to match morphological variants. Search terms are also drawn from my database of actual queries from WaC, PIE and KWiCFinder.[9] Any query that matches fewer than 30,000 HTML pages or 10,000 PDF pages is discarded. Finally, all pages excerpted by my LS-based Web concordancer (http://webascorpus.org/searchwac.html) are being archived for possible inclusion in the WaC database.

Ideally WaC will represent the entire range of native-speaker English found online,[10] but how to ensure or even measure representativeness remains elusive. Toward this goal I aim for proportional geographic representation based on population of the major national variants of native-speaker English weighted to reduce the preponderance of

---

[8] These idiosyncrasies occasioned much frustration while developing LS-based web concordancing for http://webascorpus.org/searchwac.html. Cf. also McCown and Nelson (2007 a and b).

[9] An alternative would be to attempt a near-random sample of the SE indices using techniques like those proposed by Bar-Yossef Z. and Gurevich (2006) or Anagnostopoulos, Broder and Carmel (2005).

[10] Leech (2007) offers a detailed discussion of the concept and challenges of "representativeness" in a web context.

American English on the one hand and to allow sufficient sample size of the smallest English-speaking countries on the other. Initially, for each 100 pages sampled, 10 come from Australia, 13 from Canada, 2 each from Ireland and New Zealand, and 30 from the UK, leaving just 43 for the US.[11] For each set of query terms several back-up documents per country will be downloaded to replace any weeded out by subsequent filtering. In addition to HTML pages, about 10% of the corpus will be from PDF files, which typically have higher-quality text and represent specific genres of interest (*e.g.* scholarly papers, print media and government documents).

2.1.3. Download and analysis

To net a billion words requires downloading and processing on the order of a million webpages. To expedite this phase I ran a suite of programs on multiple PCs. Here I will first describe the original plan, then the procedure that was actually followed. I intended to use the 42 networked PCs running under Windows XP in our language laboratory.[12] The week before the practical phase of this project was to begin I found out that a planned renovation of the facilities was being moved up by six months, so the room had to be vacated immediately.

The original plan is outlined in this section, and the procedure actually followed is described in 2.1.4. To simplify things as well as comply with my institution's security policies, a parallel processing strategy was envisioned instead of a master-slave or peer-to-peer collaborative environment. Querying SEs and fetching and analyzing webpages is well suited for what has been called "naturally" or even "embarrassingly" parallel computation: the only aspects that require coordination are initial tasking, prevention of duplicates, and data aggregation at the end. Each worker PC (wPC) is assigned a set of query terms to fetch and process independently. Since wPCs process a single webpage at a time and keep track of their progress through the tasklist in a local database[13] (LDB), they can be scheduled to (re)start their work after normal

---

[11] These figures represent roughly twice the population percentages of the countries other than the US, taking into account Canada's large Francophone population, but disregarding other linguistic minorities in all countries. With this formula, even NZ and IE will have roughly 20 M words each in the gigaword corpus, a useful starting sample size. One obvious but defensible omission in this stage is the large number of countries in which standard English is the / a national language, but not the first language of the majority. A later iteration of WaC could include a category "commonwealth English".

[12] The most frequent reaction I hear to this plan is "why Windows?" Two important reasons: I have a fleet of Windows machines standing idle 12 or more hours per day (and none running Linux), and I have already programmed and assembled a suite of Windows tools for each phase of the project. As a bonus this model and these tools can be adapted by others with similar computing environments.

[13] I have found SQLite 3 (Owens 2006) an excellent performer for such tasks. It is compact (small memory footprint, occupied only when needed) and fast, outperforming MySQL on some queries even against hundreds of megabytes of data. Moreover, SQLite databases occupy a single file, easily copied to another computer for analysis or uploaded to a server for data aggregation. Weaknesses are gaps in implementation of SQL (not regular expressions!) and poorer performance in a multiuser setting with high concurrency rates.

instructional hours and suspend it again when the computers are needed for other purposes. It remains to be seen how many PCs can work simultaneously without overtaxing the available 10 Mb bandwidth (1 Gb after renovation). When all workers have completed their tasks, the data are merged and *n*-gram and text databases are built. A network file server acts as central depository for seed search term lists and for final data aggregation, and one wPC is a dedicated central MySQL server (CDB) for duplicate prevention.

At the start of data collection, each wPC reads its search terms into the LDB and starts querying LS with the goal of netting 100 "keeper" texts per search term pair in the geographic proportion discussed above, with a minimum of 10% / 5 documents oversampling per country to replace rejected ones. A list of target and corresponding cache URLs is compiled working backwards from the last item in each hitlist.[14] Before addition to the "fetchset" each URL is verified for likely productive document length by a HEAD request to LS' cache.[15] Candidate URLs are hashed and compared first to the LDB, then to the CDB to eliminate duplicate URLs; surviving "keeper" URLs are added to both DBs.

Next a wPC works its way through the URLs one document at a time:

- fetch page from LS' cache, strip HTML, normalize full text and calculate wordcount and mean paragraph length (PL)[16] in words; discard text under 500 words or with PL < 13 or > 500 (the former are likely lists or text fragments, the latter server logs, repetitive forum postings etc)
- hash survivors and compare to LDB and CDB; discard duplicates and highly repetitive documents, reduce boilerplate[17] and create a second normalized full text; discard documents under 500 words and near-duplicates ;[18] for very long

---

[14] *i.e.* starting with the 1000th or last item in each list of hits, to reduce the proportion of commercial sites or pages with unusual salience of the search terms due to repetition, short text, SE spam etc.

[15] As a guideline I have proposed 5kB minimum, 250kB maximum HTML file size (see Fletcher 2004b:198-9 for rationale and consequences); since PDFs in LS' cache are converted to extremely verbose HTML, the guidelines are 10kB / 500kB respectively.

[16] The following tags are assumed to start a new paragraph: division (<DIV>), paragraph (<P>), blockquote (<BLOCKQUOTE>), two or more successive line breaks (
), heading (<H1> etc.), horizontal rule (<HR>), table row <TR>, list item (<LI>, <DT>, <DD>), form (<form>), text area (<TEXTAREA>), select option <OPTION>, and two or more successive line feeds in a preformatted text block (<PRE>). During normalization table cells (<TD>, <TH>) are also separated by spaces.

[17] As a simple approximation to boilerplate stripping my software finds the first and last paragraph of 13 words or more; text preceding the former and following the latter is dropped.

[18] Near-duplicates are detected with shingling and hashing techniques, surveyed and compared by Henzinger (2006); cf also Bar-Yossef, Keidar and Schonfeld (2007). While traditional hashes can yield very different values for texts that differ by a single byte, an intriguing new technique uses *simhash*, a 64-bit hash claimed to produce similar values for similar texts (Manku *et al.* 2007); if the recall proves high enough, only a small subset of the texts would require the high overhead of shingle-wise comparison.

texts, sample a 40,000 word chunk for PoS tagging and inclusion in the WaC database.

- preprocess survivors for input to CLAWS[19] and tag them; reformat raw output and remap tagset onto the one used by the BNC and save in two files, one of actual word-forms, one of lemmas
- repeat for next URL

When a search term pair has been processed, verify that a proportional number and volume of texts remain from each country, then process additional documents as necessary. Select proportional random subset of texts for inclusion in WaC.

When its search term set is exhausted, a wPC has the HTML and six plain-text versions of each webpage: complete text, text less boilerplate, CLAWS input text and output text with complete C7 tagset (140 tags), and two versions of tagged text mapped onto the BNC's reduced tagsets (C5 tagset, 57 word-class tags, plus the new simplified set from the *BNX XML Edition* comprising 11 word-class tags), in addition to the original HTML from every page downloaded. Fortunately this ridiculous level of redundancy is feasible thanks to the falling cost of hard disk storage (currently around $200/TB), and it permits future reanalysis, including other approaches to stripping boilerplate, tagging and genre recognition. From the tagged text the wPC generates *1-8*-gram frequency lists of all the texts selected for WaC, one set for each country in the sample and one merged set. These are uploaded to the file server, and one wPC merges datasets from all the wPCs, a lengthy process that can begin as soon as two wPCs have finished their search term lists.

2.1.4. Details of the procedure actually followed

Due to the unexpected availability of our PC laboratory facilities, the process of querying LS and downloading matching pages was carried out over the period of a week on three available PCs. These PCs were on different networks, so their actions could not be coordinated. Each PC was assigned a set of queries and tracked URLs and MD5 hashes of the webpages downloaded in an SQLite 3 database to eliminate local duplicates; the LS cache id of each document provided some additional protection against duplicates when the results were merged.[20] A PHP script developed for webascorpus.org was used to query LS, retrieve matching webpages, strip HTML tags, and save HTML and plain-text versions of each webpage. This shortcut proved unfortunate: not only did querying and downloading become a unnecessary bottleneck

---

[19] I am grateful to Paul Rayson and UCREL for providing access to WinCLAWS, the Windows release of the CLAWS4 part-of-speech tagger used for tagging the BNC (http://www.comp.lancs.ac.uk/ucrel/claws/).

[20] Each webpage in the LS cache has a 12-digit id number which changes each time pages are crawled and the cache is refreshed. Since crawling took place over the period of a week instead of 1-2 days as originally envisioned, almost 1% of the documents appeared twice among the downloads under different cache ids. These cache ids were used as filenames, so the file system eliminated most duplicates when they were copied to a single directory automatically unless the cache id had changed.

due to PHP's single-threaded model, but subsequent analysis of hash collisions and text yield revealed serious bugs in PHP's `strip_tags()` function as well. HTML to plain-text size ratios of 100:1 and greater for over 0.2% of the roughly 800,000 pages downloaded resulted from PHP's "greedy" stripping of HTML tags, which clearly deleted content as well as markup. This necessitated complete restripping and rehashing of all webpages using Windows routines originally developed for KWiCFinder.

While downloading was still in progress I examined webpages from each LOCATION (country) specified in the query. Many had specific geographic references, and the content appeared to derive primarily from commercial, government or media sources. To ensure broader representation of Web content I submitted a portion of the remaining queries without specifying the country. The preliminary distribution of unique and "first-pass keeper"[21] documents appears in Table I below.

Some anomalies in the distribution of PDF files (*e.g.* very high yield of words per document for all countries but US) may be artifacts of the range of document types encoded as PDF in the various countries, of the PDF-to-HTML-to-text conversion routines, or even of my winnowing algorithms. Since these anomalies require further study to resolve, no PDF documents are included in WaC's first release.

A single PC was used to consolidate and process webpages from the three worker PCs. At this stage processing entailed elimination of duplicates by URL, HTML-to-text conversion, elimination of duplicates by MD5 hash of the entire document, generation and merging of 1-6-gram files by country, and merging of all *n*-gram files into a composite database of all countries. At this writing (August 2007) only this composite *n*-gram database based on roughly half a billion tokens is available online, but search by country should also be implemented in time for WAC3.

PoS tagging has been deferred for several reasons. First, it is very costly in terms of computing resources, which was the original rationale for a parallel processing approach: preprocessing of the documents into acceptable input for the CLAWS4 tagger[22], tagging, and postprocessing to map the tagger output onto something comparable to the BNC encoding requires up to 2 hours per million words on a decent PC (AMD Athlon 64 3500+ / 1.8 GHz processor, 512 MB memory), that is about 1000 hours (= 6 weeks on one PC, vs. a single day on a lab-full of PCs) for the HTML-only documents. Moreover, I intend to take advantage of techniques identified by CLEANEVAL as most successful at eliminating boilerplate to restrict tagging to the

[21] My program kfWinnow discards duplicates and sorts out "keeper documents" with word counts between 500 and 50,000 and average paragraph length of 13 to 500 words. Shorter documents are ignored, while longer documents will be reviewed individually for possible excerption and inclusion.

[22] CLAWS expects only lower-ASCII alphanumeric characters and sentence punctuation. Other symbols (*e.g. $, £*) and upper-ASCII characters (*e.g. é*) must be mapped onto SGML entities. URLs are removed and reinserted during postprocessing so they are not tagged and misinterpreted as erroneous text.

coherent text in the documents. This likely will entail reanalyzing the original HTML documents, a step that should precede tagging. In addition, I would like to understand and resolve the anomalies in the PDF files outlined above. Finally, I want to find another group of PCs to test the feasibility and efficiency of the parallel processing approach of my original research design.

Unique documents downloaded

country	type	docs	%	total / country	words	%	total / country
AU	HTML	59,646	8.6%		104,987,928	12.7%	
	PDF	11,779	12.6%	71,425	58,671,791	19.6%	163,659,719
CA	HTML	75,875	11.0%		103,253,583	12.5%	
	PDF	11,428	12.3%	87,303	41,944,225	14.0%	145,197,808
GB	HTML	162,648	23.6%		172,776,173	20.9%	
	PDF	21,615	23.2%	184,263	74,170,447	24.7%	246,946,620
IE	HTML	18,542	2.7%		58,067,404	7.0%	
	PDF	6,061	6.5%	24,603	53,440,750	17.8%	111,508,154
NZ	HTML	18,850	2.7%		46,396,180	5.6%	
	PDF	6,259	6.7%	25,109	46,276,141	15.4%	92,672,321
US	HTML	230,904	33.5%		229,856,806	27.8%	
	PDF	27,059	29.1%	257,963	6,396,250	2.1%	236,253,056
unspecified	HTML	123,493	17.9%		112,865,921	13.6%	
	PDF	8,932	9.6%	132,425	18,991,926	6.3%	131,857,847
TOTALS		783,091			1,128,095,525		
	HTML	689,958			828,203,995		
	PDF	93,133			299,891,530		

First-pass keeper documents *(not eliminated by automatic criteria)*

AU	HTML	30,351	9.9%		64,330,423	11.8%	
	PDF	4,386	14.7%	34,737	43,365,583	17.0%	107,696,006
CA	HTML	36,936	12.0%		67,734,290	12.4%	
	PDF	3,558	11.9%	40,494	29,021,268	11.4%	96,755,558
GB	HTML	71,468	23.3%		117,300,203	21.5%	
	PDF	6,605	22.2%	78,073	50,113,983	19.6%	167,414,186
IE	HTML	11,443	3.7%		34,717,520	6.4%	
	PDF	3,849	12.9%	15,292	47,285,382	18.5%	82,002,902
NZ	HTML	11,020	3.6%		27,514,457	5.1%	
	PDF	3,336	11.2%	14,356	38,587,688	15.1%	66,102,145
US	HTML	99,181	32.3%		156,308,220	28.7%	
	PDF	6,152	20.7%	105,333	36,309,334	14.2%	192,617,554
unspecified	HTML	46,742	15.2%		76,616,049	14.1%	
	PDF	1,899	6.4%	48,641	10,643,254	4.2%	87,259,303
TOTALS		336,926			799,847,654		
	HTML	307,141			544,521,162		
	PDF	29,785			255,326,492		

Table 1. Geographic and Document-Type Distribution

2.2. Deploying the WaC databases online

2.2.1. Building databases from the data

When all the data had been normalized[23] and merged, MySQL databases were built using a similar architecture to PIE. For PIE I was able to compile the databases on an desktop machine, then copy them directly to MySQL's data directory on the server, which accelerated the process greatly: MySQL's LOAD DATA INFILE syntax is much faster than multiple inserts, and subsequently all the PC's resources could be dedicated to indexing and compressing the databases. This approach is only possible with cooperation from the system administrator, as access to the database directory on the server poses a potential security risk. In contrast, both my private hosting companies only permit me to upload SQL scripts to build and index databases from scratch, an arduous task in a hosting environment in which all resources must be shared and long-running scripts are suspended by the host. For this reason I have limited the tables of *n*-grams to those occurring 3 or more times for all but $n = 1$. Table 2 illustrates the both the savings and the losses this practical decision entails. In later iterations I will build separate tables of the low-frequency *n*-gram to enable their study without hampering overall search performance for higher-frequency items.

	1-grams	2-grams	3-grams	4-grams	5-grams	6-grams
total	3,123,996	57,140,986	210,320,192	359,073,268	440,426,238	471,511,994
1x	57.0%	67.0%	79.5%	87.7%	92.5%	94.8%
2x	14.0%	13.1%	10.2%	7.3%	5.1%	3.9%
≥ 3x	29.1%	19.9%	10.3%	5.0%	2.3%	1.3%

Table 2. Frequency Distribution of 1-6-grams in WaC (518,129,710 tokens)

PIE supports full-text search to return a random set of concordances of a given *n*-gram. Unfortunately MySQL's full-text indexing is both relatively limited and slow. In particular, it ignores stopwords, words shorter than 4 characters and those occurring in more than 50% of the fields indexed,[24] all factors which make it worthless for finding phrases with most function words. For phrases including stopwords PIE simply does a wildcard search in a randomized version of the corpus. Alternatives to MySQL full-text indexing will be evaluated. Of particular interest: Sphinx, which is tightly

[23] For PIE and WaC I normalize to make the data more manageable by eliminating word-external punctuation, converting all letters to lower case and mapping numerals onto #. Users who require further distinctions will find them in the concordances available by clicking on a word or phrase that matches their query.

[24] MySQL permits these lists and parameters to be modified, but a more inclusive index makes full-text searching even slower.

integrated with MySQL, and Lucene, which is readily scalable to gigacorpora.[25] I also am intrigued by the potential of the Bindings Engine (Cafarella and Etzioni 2005 and 2006), which is not yet available for general distribution.

2.2.2. WaC and PIE

While implementing WaC I will also revamp PIE's database design and Web user interface, overdue for revision after four years. The next step will be to update PIE with data from the 2007 BNC-XML release. The revised PIE will be mirrored on a neutral site, http://phrasesinenglish.org, that is independent from my institution, which by policy must block traffic from "potentially hostile" sites (including respected universities in Belgium and Brazil).

WaC will be tightly integrated with PIE: a query to either can be linked to or filtered by the other, either to find comparable data from both or to identify data that exist in one corpus but not the other. Experience suggests that there will be little in the BNC that is not echoed by WaC, and much in WaC that has no counterpart in the BNC (Fletcher 2004b:201). The rich variation inherent in large-scale Web-based corpora rewards the user who understands and tolerates the uncertainties inherent in Web data. The added value of grammatical annotation will allow us to complement fully the now-classic (but static) BNC with a dynamically expanding corpus from the Web.

2.3. The future of Web as Corpus

We Web as Corpus evangelists have been rightly accused of preaching a mixed message: on the one hand we extol the potential of the Web as a corpus and for corpus compilation, yet on the other we caution against the inherent dangers and possible misuse of Web data (*e.g.* Kilgarriff 2007). Now many of us are working independently on parallel efforts to make the Web more accessible and more credible as a linguistic resource (*e.g.* Renouf, Kehoe and Banerjee 2007; Baroni and Kilgarriff 2006; Sharoff 2006). Our shared challenge is to provide conscientiously compiled Web corpora which enhance raw data with meaningful linguistic tools and support responsible (and replicable) research by scholar and novice alike. Our mutual reward will be general acceptance of reliable Web data for linguistic scholarship.

[25] http://www.sphinxsearch.com/ and http://lucene.apache.org/

References

ANAGNOSTOPOULOS A., BRODER A. and CARMEL D. (2005), "Sampling Search-Engine Results", WWW 2005, May 10-14, 2005, Chiba, Japan.
http://www2005.org/cdrom/docs/p245.pdf

ARCHER D., WILSON A. and RAYSON P. (2002), "Introduction to the USAS Category System".
http://www.comp.lancs.ac.uk/ucrel/usas/usas%20guide.pdf

BAR-YOSSEF Z. and GUREVICH M. (2006), "Random Sampling from a Search Engine's Index", WWW 2006, 23-26 May 2006, Edinburgh, Scotland.
http://www2006.org/programme/files/pdf/3047.pdf

BAR-YOSSEF Z., KEIDAR I. and SCHONFELD U. (2007), "Do Not Crawl in the DUST: Different URLs with Similar Text", WWW 2007, 8-12 May 2007, Banff, Alberta.
http://www2007.org/papers/paper194.pdf

BARONI M. and KILGARRIFF A. (2006), "Large Linguistically-Processed Web Corpora for Multiple Languages", in Keller F. and Proszeky G. (eds), *Conference Companion EACL 2006* (11th Conference of the European Chapter of the Association for Computational Linguistics), East Stroudsburg PA, ACL : 87-90.

CAFARELLA M. J. and ETZIONI E. (2005), "A Search Engine for Natural Language Applications", WWW 2005, May 10-14, 2005, Chiba, Japan.
http://www2005.org/cdrom/docs/p442.pdf

CAFARELLA M. J. and ETZIONI E. (2006), "BE : A Search Engine for NLP Research", Second Web as Corpus Workshop, 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006), Trento, Italy, 3 April 2006.

DAVIES M. (2005), "The Advantage of Using Relational Databases for Large Corpora: Speed, Advanced Queries, and Unlimited Annotation", in *International Journal of Corpus Linguistics,* 10(3) : 307-334.

FLETCHER W. H. (2004a), "Facilitating the Compilation and Dissemination of Ad-Hoc Web Corpora", in Aston, G., Bernardini S. and Stewart D. (eds.), *Corpora and Language Learners,* John Benjamins, Amsterdam: 271-300.

FLETCHER W. H. (2004b), "Making the Web More Useful as a Source for Linguistic Corpora", in Connor U. and Upton T. (eds.), *Corpus Linguistics in North America 2002: Selections from the Fourth North American Symposium of the American Association for Applied Corpus Linguistics,* Rodopi, Amsterdam: 191-205.

FLETCHER W. H. (2005), "Towards an Independent Search Engine for Linguists: Issues and Solutions", Web as Corpus Workshop, SSMILT, Forlì, Italy, 14 January 2005.
http://www.kwicfinder.com/WaCForli2005-01.pdf

FLETCHER W. H. (2007), "Concordancing the Web: Promise and Problems, Tools and Techniques", in Hundt M., Nesselhauf N. and Biewer C. (eds), *Corpus Linguistics and the Web,* Rodopi, Amsterdam: 25-45.

HENZINGER M. (2006), "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms", SIGIR '06, 6-11 August 2006, Seattle, Washington.
http://infoscience.epfl.ch/getfile.py?mode=best&recid=99373

KILGARRIFF A. (2007), " Googleology is Bad Science", in *Computational Linguistics* 33(1): 147-151.

LEECH G. (2007), "New Resources, or Just Better Old Ones? The Holy Grail of Representativeness", in Hundt M., Nesselhauf N. and Biewer C. (eds), *Corpus Linguistics and the Web,* Rodopi, Amsterdam: 133-149.

MANKU G. S., JAIN A. and SARMA A. D. (2007), "Detecting Near Duplicates for Web Crawling", WWW 2007, 8-12 May 2007, Banff, Alberta. http://www2007.org/papers/paper215.pdf

MCCOWN F. and NELSON M. L. (2007a), "Characterization of Search Engine Caches", IS&T Archiving 2007, 21-24 May 2007, Arlington, VA. http://arXiv.org/cs.DL/0703083

MCCOWN F. and NELSON M. L. (2007b), "Agreeing to Disagree: Search Engines and their Public Interfaces", ACM IEEE Joint Conference on Digital Libraries (JCDL 2007). June 17-23, 2007. Vancouver, BC. http://www.cs.odu.edu/~fmccown/research/se_apis/pubs/se-apis-jcdl07.pdf

OWENS M. (2006), *The definitive guide to SQLite,* APress, Berkeley, AC.

RAYSON P., WALKERDINE J., FLETCHER W. H. and KILGARRIFF A. (2006), "Annotated Web as Corpus", in Kilgarriff A. and Baroni M. (eds), *Proceedings of the Second Web as Corpus Workshop, 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006),* Trento, Italy, 3 April 2006: 27-33. http://acl.ldc.upenn.edu/W/W06/W06-1705.pdf//

RENOUF A., KEHOE A. and BANERJEE J. (2007), "WebCorp: an Integrated System for Web Text Search", in Hundt M., Nesselhauf N. and Biewer C. (eds), *Corpus Linguistics and the Web,* Rodopi, Amsterdam: 47-67.

SHAROFF, S. (2005), "Open-Source Corpora: Using the Net to Fish for Linguistic Data", *International Journal of Corpus Linguistics* 11(4) : 435-46.

SHAROFF S. (2006), "Creating General-Purpose Corpora Using Automated Search Engine Queries", in Baroni M. and Bernardini S. (eds). *Wacky! Working Papers on the Web as Corpus.* GEDIT, Bologna: 63-98. http://wackybook.sslmit.unibo.it/sharoff.pdf

UEYAMA M. (2006), "Evaluation of Japanese Web-Based Reference Corpora: Effects of Seed Selection and Time Interval", in Baroni M. and Bernardini S. (eds). *Wacky! Working Papers on the Web as Corpus.* GEDIT, Bologna: 99-126. http://wackybook.sslmit.unibo.it/ueyama.pdf