Scaling to Billion-plus Word Corpora

Jan Pomikálek[1], Pavel Rychlý[1] and Adam Kilgarriff[2]

[1] Masaryk University, Brno, Czech Republic
[2] Lexical Computing Ltd, Brighton, UK

Abstract. Most phenomena in natural languages are distributed in accordance with Zipf's law, so many words, phrases and other items occur rarely and we need very large corpora to provide evidence about them. Previous work shows that it is possible to create very large (multi-billion word) corpora from the web. The usability of such corpora is often limited by duplicate contents and a lack of efficient query tools.
This paper describes BiWeC, a Big Web Corpus of English texts currently comprising 5.5b words fully processed, and with a target size of 20b. We present a method for detecting near-duplicate text documents in multi-billion-word text collections and describe how one corpus query tool, the Sketch Engine, has been re-engineered to efficiently encode, process and query such corpora on low-cost hardware.

1 Introduction

There's no data like more data, and one place to get more data almost without limit (for general English and some other languages and varieties) is the web. One way to use the web is to create a local corpus by downloading web pages: in [1] we argue that it is the optimal way to use the web for linguistic research. A number of corpora have been built in this way: Baroni and colleagues developed web corpora with nearly 2 billion words for German, Italian and English [2, 3] and have made them available for research as tar archives. Liu at al. [4] describe the creation of a 10 billion word corpus. In this paper we introduce BiWeC, a Big Web Corpus currently of 5.5b words, with a target size of 20b.

Very large corpora can be created on low cost hardware in a few person-months. Most of the steps have linear complexity and scale up well. The two outstanding issues we focus on in this paper are:

1. removing duplicate content
2. efficient querying.

The article is organized as follows. In section two our motivation for creating larger corpora is discussed and the advantages of using more data for various tasks is explained. Section three describes the process of creating BiWeC, with a focus on removing duplicate and near-duplicate documents. Section four deals with corpus processing and querying using the Sketch Engine corpus manager. Then we present some figures about BiWeC and outline future plans.

2 Motivation

Bigger corpora provide more information. To illustrate why a BNC[3]-sized corpus is often not enough we take a sample of headwords from one page of Macmillan English Dictionary [5] and compare their frequencies in the 50,000 word Suzanne corpus, the 100m word BNC, and BiWeC (Big Web Corpus), our new web corpus containing 5.5 billion words of general English.

Table 1. Word frequencies comparison for Susanne, BNC and BiWeC

	Susanne	BNC	BiWeC	BiWeC/BNC
Size [m words]	0.15	111	5,500	49×
heavy (adj)	11	9,089	252,305	27×
hector (v)	0	37	956	25×
hedge (n)	0	1,525	19,526	12×
hedonism (n)	1	63	1,757	27×
heebie-jeebies	0	0	151	-

Corpora like Susanne are not usable for lexical knowledge, BNC is good for high frequency words but scarcely provides enough information to make informed generalisations on *hector* (verb) and certainly does not for *heebie-jeebies*. BiWeC provides ample evidence in both cases. The issue is more acute still for phrasal and collocational items.

One common role for corpora is exploration of variation in language, according to, for example, genre, domain or region, or over time. To support such studies, a corpus should be big enough to have subcorpora where the expected frequency for the item being studied is at least thirty or forty. Here, the BNC is often too small. Consider DOMAIN: the written part of the BNC can be divided into ten broad domains, with subcorpus sizes between three and nineteen million words. So a word like *hedonism* has an expected frequency of just two in the smallest of these subcorpora, and there is not enough data in the BNC to support a study of the use of the word across domains.

As we make more and more use of corpora, so the merits of having ample data even for non-core phenomena are more and more evident. In recent work we have developed GDEX, a "good dictionary example extractor" [6]. We aim to find good candidate sentences to be used to exemplify collocations in dictionaries and language teaching. So, we gather all the example sentences in the corpus for a given collocation, reject those that have undesirable features (such as non-words, 'bad' words, many long words, excessive punctuation, or where the sentences are long) and choose the most desirable of the remainder (preferring sentences that are short, contain common words, appear to have standard grammar and contain other words typical of the settings in which the collocation is found). If there

[3] British National Corpus, see `http://natcorp.ox.ac.uk/`, comprising 100m words.

was a set of 100 sentences to start with, we are likely to find a good candidate for a dictionary or teaching example. If we only had five sentences to start with, we are unlikely to.

2.1 Relations with Google

One response to the argument for size above is "why not, then, use the web via Google". This is an entirely pertinent response: Google is a spectacularly fast query engine looking at a spectacularly large corpus. For Google, the 'local corpus' is as much of the web as it has succeeded in indexing. (It would like to index the whole web, but in the course of its crawling it does not find everything. It probably finds a greater proportion of everything than any of its competitors: for a review of these issues and discussion on index sizes see [7].) Google, Yahoo and other web search engines share many of the concerns of linguistic corpus developers and corpus tool developers. They want large corpora, with wide coverage, with duplication intelligently handled and with spam weeded out. They want to index on terms in the content of the page rather than in navigation bars and advertisements. 'Text-heavy' pages and pages that have lots of readers are or highest value to them. They would like to operate (by default) on lemmas; word class information would be useful to them. While the goals are distinct, since search engines help people find out about things whereas corpus research looks at the language denoting the things, the route to those goals is often shared.

The Google-indexed web is far larger than any corpus developed for linguists: in Table 2 we repeat Table 1 with Google counts added.

Table 2. Comparing corpus frequencies with the web via Google. Note that: Google hit counts are for document counts not word counts; they are for word forms not lemmas and are not word-class-specific; and they can vary from one hour or day to the next. The searches were undertaken with default settings except that the language was set to English and 'allintext' was set so that the search term had to be in the text rather than, for example, in a link. The differences in what is being counted probably account for the high ratio for *hector*, which often occurs as a name.

	Susanne	BNC	BiWeC	BiWeC/BNC	Google	Google/BiWeC
Size [tokens]	.15m	111m	5 500m	49×		
heavy (adj)	11	9 089	252 305	27×	242.0m	955×
hector (v)	0	37	956	25×	22.1m	23,000×
hedge (n)	0	1 525	19 526	12×	25.8m	1,321×
hedonism (n)	1	63	1 757	27×	2.4m	1,343×
heebie-jeebies	0	0	151	-	20,900	138×

A cluster of researchers have used Google and other search engines in this way, see e.g. [8]. The great benefit is the size. Where the phenomena under investigation are too rare to have a reasonable number of hits in 5.5b words

of English, this is currently the only course available. But there are costs and problems relating to the strategy [1]:

- the query language is limited
- no searching for lemmas or by word class or other linguistic parameters
- search hits are ordered in a very specific manner which is not relevant to linguistic research and cannot (in Google) be turned off
- searches are limited: over-users of Google may have their access blocked, and also a maximum of 1000 hits are given per query
- methods are not published
- Google may change methods and the query language, without saying they are doing so or offering any explanation how or why, at any point.

For these reasons, wherever the corpus is big enough to provide enough data to support the research question, we advocate the use of a corpus prepared using the methods described here and loaded into a query tool specialised for linguistic research. So – the bigger the corpus, the more research questions can be addressed without recourse to Google or other commercial search engines, with all the associated disadvantages.

3 Building BiWeC

3.1 Crawling

We retrieve textual data from the web using procedures similar to [2, 3]. We used the Heritrix web crawler developed by the Internet Archive[4] and started the crawl from the same list of URLs as Ferraresi et al.[5] The crawl was restricted to domains considered likely to contain English texts, such as `.uk` or `.au`. For practical reasons we only processed HTML pages (`content-type: text/html`). We also applied restrictions to the size of the processed documents and filtered out all web pages smaller than 5 kB or larger than 2 MB. The rationale is that pages smaller than 5 kB contain little if any textual content and pages larger than 2 MB are usually not what we want but are logfiles, lists, catalogues or similar. Unlike Ferraresi et al. we configured the web crawler to perform this basic filtering for us rather than storing all crawled data and filtering as a postprocessing step, so reducing bandwidth requirements and disk space.

3.2 Cleaning

'Cleaning' the retrieved web pages involves stripping out the HTML mark-up and removing content such as navigation links, copyright notices, advertisements, etc. To perform this step we considered the participating systems in the CleanEval

[4] `http://www.archive.org/`

[5] We would like to thank the ukWaC corpus team for providing us with the list of seed URLs.

competition for cleaning webpages [9]. However, our experiments revealed that even the winning system of CleanEval is matched by the BTE algorithm [10].[6]

BTE works from the observation that the material we wish to remove is usually rich in markup. It establishes the ratio of text to markup for different chunks of the page. The ratio is most often high at the beginning and end of the page and lower in the middle. BTE retains only that part of the page where the ratio is low.

3.3 Character encoding conversion

To unify the character encoding of the corpus contents we converted all downloaded pages to UTF-8. As long as we only processed HTML pages, we were able to determine the original character encoding from the HTML headers. We are well aware that the header information is not always reliable and that techniques exist for guessing the character encoding from textual contents of a web page. However, for English, occasional errors in character encoding conversion do not cause problems. Therefore we considered using any advanced encoding detection techniques unnecessary.

3.4 Language detection

Language detection was performed to filter out non-English documents which occasionally occur in the crawled domains. The problem of language detection has been well described in the literature (e.g. [11]) and many freely available language detection systems exist. Having the postprocessing procedure fully implemented in Python, we chose the Trigram class[7] which performs the language identification based on frequencies of triples of characters. As a by-product of the language detection, noisy texts were also filtered out, such as documents full of JavaScripts which the cleaning phase failed to remove.

3.5 Removing duplicate and near-duplicate documents

Duplicate documents in text corpora do damage to corpus derived statistics. Much corpus use is based around identifying patterns which are much more common that one would expect by chance. For example, collocation studies are premised on a word and its collocate appearing together remarkably often. Collocation-finders are repetition-spotters. If a corpus contains many duplicate texts, then supposed collocation lists will often have contents that result, not from a *bona fide* collocation, but from the duplication. Because much corpus research is regularity-spotting, it is easily derailed by the regularities provided by duplication. Duplicate concordance lines are also an irritant, and potentially

[6] BTE achieved a score of 85.41 on the CleanEval test set in the text-only cleaning task, while the CleanEval winner (Victor) scored 84.07.

[7] `http://code.activestate.com/recipes/326576/`

misleading, in the manual exploration of corpora. For a high-quality corpus, removing duplicates is essential.

Identical web pages can be easily detected by comparing their checksums. This, however, does not work for near-duplicate web pages as even a small change to the contents changes the checksum.

Liu and Curran [4] describe how they created a 10 billion words corpus but do not discuss duplication. Ferraresi et al. [3] removed near-duplicate documents from the ukWaC corpus using a technique based on Broder's fingerprinting algorithm [12]. This method works well only for very similar documents but does not detect documents which contain both significant identical parts and different parts. This is illustrated in Fig 1. We have analysed ukWaC to establish the duplication it retains, and while it does not include many 100 % duplicates, it does contain large numbers of partial duplicates. Of a total of 2.58m documents, 28,716 have 100 % duplicate content, but 85,693 have at least 80 % duplicate content.

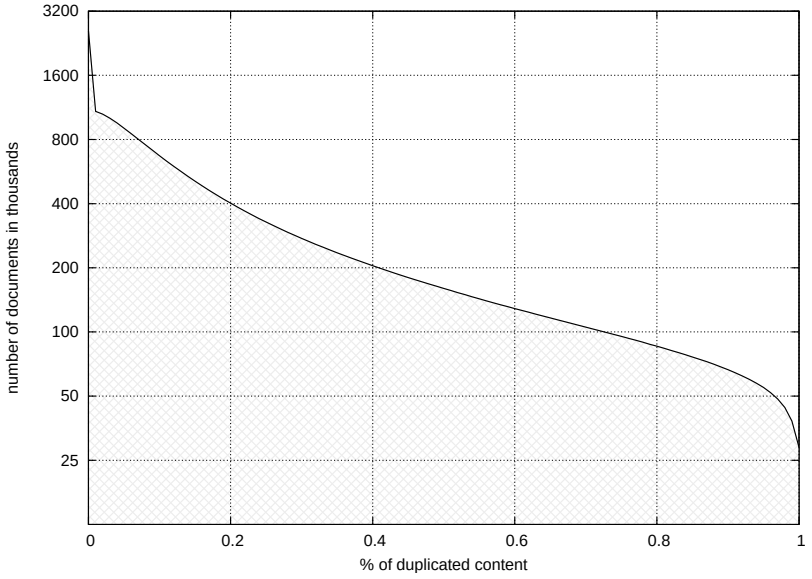

Fig. 1. Duplicate and near-duplicate documents in the ukWaC corpus

Many variations of fingerprinting algorithms for near-duplicate documents exist. These are mostly based on finding shared sequences of words among documents. To make the process feasible for very large data, only small samples of each document are used. This, however, constitutes a loss of data which means that only very close documents are identified, such as a regular web page and a version of it for printing. The technique is commonly used by search engines

for grouping near-duplicates but is not so suitable for us since it lets through a substantial amount of duplicate text.

In our previous work [13] we developed a new algorithm for detecting duplicated text, and this is what we have applied in the building of BiWeC. It works as follows.

Let us assume that undesirable duplication occurs wherever a string of n or more words is duplicated (we have used $n = 10$). An exhaustive method of duplicate detection would work with all 10-word strings in the corpus. This is prohibitive for large corpora, but we can reason that we do not need to work with all of them. We only need to find the duplicate 10-word strings.

The core idea of our algorithm is to use an external sort method to generate all n-grams together with their counts directly, in one step (as opposed to the SPEX algorithm [14] which iterates through 3-grams, 4-grams, ... n-grams, removing non-duplicated items at each stage: our earlier paper shows that SPEX does not scale well). The program splits the input text into chunks which fit into a fixed amount of memory. For each chunk, a sorted list of n-grams is generated and saved to a temporary file on disk. The final phase joins all temporary files and outputs any n-gram with total count higher than one. This is a typical external sort method which would require a huge amount of disk space and a lot of runs (chunks) to process the whole input text.

Having the list of duplicate 10-grams available, we can easily determine the amount of duplicated data in each document. This allows us to explore the tradeoff between partial-duplicate-removal and corpus size. We may choose to remove all documents with over 10 % of data duplicated, or over 50 %, or over 90 %. The optimal figure will depend on the sensitivity of the applications and users to duplication, versus the demand for a large corpus. For BiWeC, after some experimentation, we have used a threshold of 50 %. The size of BiWeC before and after de-duplication is shown in Table 3.

Table 3. BiWeC size before and after removing duplicate content

	before deduplication	after deduplication
number of documents	≈ 9 mil.	4.3 mil.
number of tokens	**9 bil.**	**5.5 bil.**
number of different word forms	36 mil.	28 mil.
number of different lemmas	32 mil.	27 mil.

The algorithm does not give rise to a database which can be used to ask, of a new document, does it duplicate material already in the corpus, so is not suitable where a corpus is frequently to be added to. Such a database is provided by fingerprinting algorithms such as Broder's.

In the paper cited above we demonstrate that the algorithm scales well to billions of words. Also the de-duplication was a one-off exercise: if the corpus is regularly to be added to, we shall revisit suitable methods.

3.6 Corpus Annotation

For tagging the corpus we have used TreeTagger [15], a widely-used state-of-the-art part-of-speech tagger based on decision trees. For English, it lemmatizes as well as POS-tags. We have used it with the English tagset and parameters as distributed, and have not re-trained it. We have good experiences of its performance and results over the past seven years: the accuracy has proved to be adequate for many corpus based tasks such as building word sketches [16, 17] and distributional thesauruses. In addition, the TreeTagger is fast, processing about one million words per minute on a single machine: sufficient even for a corpus with several billion words.

4 Query Engines for multi-billion-word corpora

Many interesting linguistic results can be computed by simple batch processing of the whole corpus. They include word list and bigram (or n-gram) lists with frequencies, collocation lists for a particular word or lemma, lists of words or lemmas which are particularly salient in a particular subcorpus ('keywords') and lists of words particularly associated with a grammatical construction or pattern, such as "nouns with a strong tendency to occur in the plural" [18].

Studying such lists, linguists usually find data which are surprising and need some explanation. This is best done by browsing concordances. Users need to be able to query the corpus interactively as well as in batch mode. This demands fast query evaluation.

Most current computers use 32-bit numbers and even on 64-bit machines the default numeric type in many programming languages is 32-bit only. The maximum number for a signed 32-bit numeric type is $2\hat{3}1$, which is slightly more than 2 billion: the computer cannot (readily) count higher than 2 billion. Many off-the-shelf algorithms use the default numeric type and are not prepared for more than 2 billion items. For example, advanced implementations of data processing systems often use *memory-mapping*, which enables mapping of data in files to main memory addresses without a long sequential read of the whole data file. The operating system is responsible for reading the respective data block from files to the main memory. This feature can greatly improve the speed of a program and also simplify the implementation. However 32-bit machines cannot memory-map more than 2 GB of data.

While the revision of the code to use 64-bit numbers at all points is not intrinsically a hard technical challenge, there are many fixes that need to be made to cover the code of the whole system.

We have recently re-engineered the Manatee corpus query engine so that it can handle corpora of over 2 billion words [19].[8]

[8] Manatee is incorporated in a few derived products; it is the core engine of the Sketch Engine [17].

4.1 Corpus Encoding

In Manatee, the binary files holding the encoded and indexed corpus are roughly the same size as the input plain or annotated text in Manatee's one-word-per-line input format. Each variety of annotation (e.g., wordform, lemma, POS-tag, sentence-markup) increases the size of the input data and the size of the encoded data in similar ways. The same amount of disk space again is needed for temporary storage during the encoding process. Data sizes are shown in Table 4.

Table 4. BiWeC data sizes (after removing duplicates)

downloaded data size	≈ 1 TB
cleaned data in vertical format (POS-tagged, lemmatised)	72 GB
encoded data	69 GB

The total encoding time depends not only on the number of words in the corpus but also on the amount of annotation.

The first stage of encoding on a standard server take about 10 hours per billion tokens. Encoding of 9.5 billion token corpus ran for four days on a 2 GHz AMD Opteron machine with 2 GB of memory dedicated to the encoding process.

After this stage, the corpus can be queried for concordances, concordance-based functions including frequency distributions, collocations and concordance sorting, and word lists. Additional indexes can be computed to speed up some type of queries (for example ignoring upper and lower case of lemmas). Computation time for such indexes depends on the size of the lexicon, that is, the number of different words in the corpus.

An encoded corpus can be used by users to create concordances and word lists. A powerful query language can be used for queries. We are using the Sketch Engine to build word sketches (one-page, automatic, corpus-derived summary of a word's grammatical and collocational behavior [17]). The usefulness of such information depends on how many instances of grammatical relations we find in the corpus for the given word.

For the BNC there are 7,414 words for which more than 1000 grammatical relation instances were found. In the current fully-processed version of BiWeC, comprising 5.5b tokens, there are 76,689 such words. We have ten times as many words for which we can generate a detailed and thorough account of the word's behavior.

Table 5. BiWeC processing times

computing word sketches	16 hours
computing thesaurus	40 minutes

Table 6. The number of lemmas for which rich, data-driven analyses are available in each of three corpora. The analyses are built on the evidence of the grammatical relations (gramrels) that a word occurs in, and the other words it occurs with. At least one hundred, and preferably several hundred instances are required for a good word sketch.

gramrel hits	>100	>1000
BNC (100m)	29,931	7,414
ukWaC (1,500m)	74,293	21,614
BiWeC (5,500m)	335,294	76,689

5 Conclusions and future work

We are able to build multi-billion word corpora which are suitable for linguistic research, and we have tools which can encode and query them efficiently. We have introduced one such corpus, BiWeC, a corpus of general English, currently of 5.5 billion words fully prepared and accessible in our tools (Manatee and the Sketch Engine) and with a target size of 20 billion words.

We have a long agenda for further developments to both the corpus and the tool. Our highest priority for the tool is to address hardware limitations relating to disk access speed. We shall be exploring Amazon's 'cloud computing' which is rumored to offer very fast disk access. In relation to the corpus, in addition to gathering more data, we wish to classify documents using text classification methods along a number of dimensions including at least domain, formality and region. In a companion paper [20] we describe a parallel stream of work in which we are developing a 100m word "New Model Corpus", half of which is taken from BiWeC (with data-driven classification) and the other half taken from web sources which we know to be fiction, or chatshow transcripts, or film transcripts, so we have a corpus which, like the BNC and before it LOB and Brown, supports a variety of studies across language varieties. We also want to explore additional textual markup, including for named entities, time phrases, and semantic categories of nouns: we plan to do this in an open, collaborative model, working with other groups who have tools and expertise in these forms of annotation. The New Model Corpus will be made freely available for research purposes. Our plan is that the two streams should merge, to give a multi-billion word corpus with many interesting and still very large subcorpora and rich textual markup.

Acknowledgments

This work has been partly supported by the Academy of Sciences of the Czech Republic under the projects T200610406, T100300419 and by the Ministry of Education of CR within the Centre of Basic Research LC536 and National Research Programme 2C06009.

References

1. Kilgarriff, A.: Googleology is Bad Science. Computational Linguistics **33**(1) (2007) 147–151
2. Baroni, M., Kilgarriff, A.: Large linguistically-processed web corpora for multiple languages. Proceedings of European ACL (2006)
3. Ferraresi, A., Zanchetta, E., Baroni, M., Bernardini, S.: Introducing and evaluating ukWaC, a very large web-derived corpus of English. In: Proceedings of the 4th Web as Corpus Workshop (LREC 2008). (2008)
4. Liu, V., Curran, J.: Web Text Corpus for Natural Language Processing. EACL. The Association for Computer Linguistics (2006)
5. Rundell, M., Fox, G.: Macmillan English Dictionary: For Advanced Learners. Macmillan (2002)
6. Kilgarriff, A., Husák, M., McAdam, K., Rundell, M., Rychlý, P.: GDEX: Automatically finding good dictionary examples in a corpus. Proceedings of Euralex (2008)
7. Sullivan, D.: End Of Size Wars? Google Says Most Comprehensive But Drops Home Page Count. Search Engine Watch (3551586) (2005)
8. Nakov, P.: Noun compound interpretation using paraphrasing verbs: Feasibility study. In: Proc. 13th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'08). (2008)
9. Baroni, M., Chantree, F., Kilgarriff, A., Sharoff, S.: CleanEval: A competition for cleaning webpages. Proceedings of LREC 2008 (2008)
10. Finn, A., Kushmerick, N., Smyth, B.: Fact or fiction: Content classification for digital libraries. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries. (2001)
11. Beesley, K.: Language identifier: A computer program for automatic natural-language identification of on-line text. Language at Crossroads: Proceedings of the 29th Annual Conference of the American Translators Association (1988) 12–16
12. Broder, A.: Identifying and filtering near-duplicate documents. Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (2000) 1–10
13. Pomikálek, J., Rychlý, P.: Detecting co-derivative documents in large text collections. Proceedings of LREC 2008 (2008)
14. Bernstein, Y., Zobel, J.: A scalable system for identifying co-derivative documents. In: Proc. String Processing and Information Retrieval Symposium, Padua, Italy (2004) 55–67
15. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. Proceedings of International Conference on New Methods in Language Processing **12** (1994)
16. Kilgarriff, A., Tugwell, D.: WORD SKETCH: Extraction and Display of Significant Collocations for Lexicography. Proceedings of the 39th ACL **10** (2001) 32–38
17. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116
18. Kilgarriff, A., Rychlý, P.: Finding the words which are most X. Proceedings of Euralex (2008)
19. Rychlý, P.: Manatee/bonito - a modular corpus manager. In: Recent Advances in Slavonic Natural Language Processing(RASLAN), Masaryk University, Brno (2007) 65–70
20. Kilgarriff, A.: New model corpus. Technical report, Lexical Computing Ltd. (2009 in preparation)