Not Just Bigger: Towards Better-Quality Web Corpora

Yannick Versley
SFB 833
Universität Tübingen
versley@sfs.uni-tuebingen.de

Yana Panchenko
Seminar für Sprachwissenschaft
Universität Tübingen
yana.panchenko@uni-tuebingen.de

ABSTRACT

For the acquisition of common-sense knowledge as well as as a way to answer linguistic questions regarding actual language usage, the breadth and depth of the World Wide Web has been welcomed to supplement large text corpora (usually from newspapers) as a useful resource.

While purists' criticism on unbalanced composition or text quality is easily shrugged off as unconstructive, empirical results in some real-world tasks have found Web corpora to be less useful than (smaller) newspaper corpora. More than the early criticism, evidence that Web corpora are doing poorly at their original purpose should raise concerns about the quality of Web corpora. Especially for non-English Web corpora, principled quality assessment and targeted improvements are instrumental in ensuring their relevance.

In this paper, we present our own pipeline for Web corpora, which includes improvements regarding content-sensitive boilerplate detection as well as language filtering for mixed-language documents. We also provide a principled evaluation of the combination of corpora and (non-linguistic and linguistic) preprocessing between more standard types of large corpora (newspaper and Wikipedia) and different Web corpora.

While our current results are focused on German-language Web corpora, both the content-sensitive boilerplate detection and our method of evaluation by constructing an artificial thesaurus from a wordnet are applicable to many other languages.

1. INTRODUCTION

Unsupervised and semi-supervised learning methods use vast quantities of text to improve the coverage and accuracy of language processing over that possible with only hand-annotated data. Large collections of suitable text are important for such learning methods: They extract features from the context of target items, and a large number of contexts (i.e., a large number of texts) is instrumental in avoiding sparse data problems in the learning task(s).

Many kinds of text are used for distributional semantics – large fixed text collection such as newspaper text, Wikipedia or the Gutenberg project, n-gram frequency databases extracted from larger text collections, or online queries to search engines. Among these types, however, Web corpora are the only type that combines replicability of results (online queries may yield different results), scalability in size (unlike single-source text collections) as well as the possibility of using rich linguistic annotations (unlike n-gram databases).

In some cases, however, Web corpora seem to be less useful despite their larger size: [BL08] found that the much smaller British National Corpus (BNC; 100 million tokens) was better-suited for the creation of a window-based distributional similarity resource than the much larger ukWaC Web corpus ([BBFZ09]; 2.25 billion tokens). For their more targeted approach using patterns, the larger size of the Web corpus together with the filtering ability of their learning algorithm were able to make better use of the greater size of ukWaC.

[FP10] used large corpora in the context of building a Named Entity Recognizer (NER) system for German; in their case, they induce a word clustering on the unannotated corpus – either 175 million tokens of German newspaper text (HGC), or the same amount of text from the deWaC Web corpus. They found that the HGC-derived word clusters were substantially more useful in the case of in-domain testing, and somewhat more useful in the case of out-of-domain testing data containing parliamentary debates.

It would be hardly surprising in general that newspaper text is a better source of in-domain unannotated textual data than general-domain text from the World Wide Web. However, the advantage of the (cleaner) newspaper text persists to a certain extent when testing on out-of-domain texts from European Parliament debates. Similarly, the balanced composition of the British National Corpus does not seem any less heterogeneous or a more straightforward match to Baroni and Lenci's psycholinguistic data than the ukWaC corpus counterpart.

These results should be seen in contrast to the evaluation by [LC06], who also compare newspaper text with a Web

corpus. On a distributional semantics task, they show a relatively small difference in scores for size-matched two billion word samples of the English Gigaword Corpus[1] and their own Web corpus.

In sum, these results make it seem worthwile to investigate the factors behind the sensitivity to differences between Web and newspaper corpora. Several attributes – text selection and domain distribution, text quality, the preprocessing used – may contribute to this sensitivity.

In terms of **domain distribution**, the World Wide Web contains texts from a large range of domains, with a few frequent domains that dominate the statistics extracted from such text collections: [Ver08] finds that the largest singular vectors in an SVD model induced from bigrams in Google's English ngram dataset (containing a verb as the first word) reflect predominant domains, including product sales, legal texts, pornography, Unix manuals, and news stories.

The distribution of **text genres** of Web text – how a given topic is talked about, as opposed to the topic itself – is also markedly different from the genres found in newspaper or newswire text. In principle, this could be seen as beneficial - after all, the use of edited newspaper text for knowledge acquisition was motivated by availability rather than other reasons, and informal writing on the Web may be closer in genre to the language use of ordinary people. More importantly, however, informal or unedited text is more difficult to process due to variation in spelling and punctuation, or the presence of more grammar deviations.

The greater variability of Web content also leads to other quality issues resulting from greater variability in how the text is presented: While removing formatting information from single-source text is usually feasible to do in near-perfect quality, Web corpora have to rely on generic techniques for **boilerplate removal** that use effective heuristics (e.g., detecting navigation elements based on the length of the text between two HTML tags, cf. subsection 2.3) but are not perfect.

Last but not least, **preprocessing tools** such as the tokenizers or part-of-speech taggers used on Web corpora may yield lower quality - either because the best-quality processing tools cannot be used for speed or licensing reasons, or because the domain and genre distribution of Web corpora makes the processing of such text more difficult.

In the remainder of this paper, we adopt a similar evaluation framework to that of [LC06] in comparing several large German corpora from a distributional semantics perspective that show different domain and genre distributions. The comparison includes two large Web corpora that we created using a custom pipeline featuring improvements in language detection for mixed-language documents as well as boilerplate removal. Of these corpora, one is targeted at general web content (**web-dmoz**) and the other targeted more at news-style content (**web-news**). As comparison to our own Web corpora, we include a portion of deWaC [BK06] as well as text from the newspaper `die tageszeitung` (TüPP-D/Z; [Mül04]) and a recent Wikipedia dump.

Of the remainder of this paper, section 2 presents the crawling and non-linguistic preprocessing steps involved in creating the Web corpora, while section 3 presents the linguistic pipeline we used on all corpora. Our pipeline is targeted at a useful balance between processing speed, enabling its use on Web-scale corpora, and usable quality of the linguistic annotations. Section 4 describes the evaluation task chosen and presents the results of our evaluation.

2. A WEB CORPUS PIPELINE

Gathering a Web corpus consists in multiple steps: In the first phase, crawling creates an archive of HTML pages that are to be processed further; the second phase consists of boilerplate removal and deduplication, which yields the raw text of these HTML pages without any navigation elements or non-informative text; in the subsequent phases, the raw text is tokenized and sentence splitting and subsequent linguistic processing is applied.

2.1 Crawling

We use the Internet Archive's Heritrix crawler, which starts from a set of *seeds* and subsequently visits linked pages that are within the *scope* of the crawl. In deciding which page to fetch next, Heritrix maintains per-host queues that prevent any single web host from being overloaded by the crawler.

The original Web-as-Corpus approach [Sha06] uses search engine queries for sets of mid-frequent terms to gather the corresponding results as seeds. This approach crucially depends on search engine results offering a suitable quality and diversity of sources. In order to assess the role of seed selection, we created two corpora using the following seed generation strategies:

- For the **web-dmoz** corpus, we use all links from the German section of the OpenDirectory project (dmoz.org) – altogether 233 884 URLs – as seeds, and limit the crawl scope to all .de/.at domains.
- For the **web-news** corpus, we used keyword queries (for medium-frequency German words) to the Google news RSS API in order to discover appropriate seeds from relevant news sources.

  The scope of the crawl is limited to the domains of the seeds, which are usually newspapers or content-focused blog sites. Because of the seed selection strategy, the scope does include sites outside exclusively German-speaking countries (such as the German version of RIA Novosty, a Russian news agency, but also the Swiss *Bieler Tageblatt* or the Namibian *Allgemeine Zeitung*).

  In our case, we use 851 seed words which are medium-frequency nouns extracted from a newspaper corpus, and filter the results returned by RSS queries so that we have at maximum ten URLs per site in our seed list, yielding 7129 seed URLs altogether.

After crawling, all HTML pages with a file size between 4 kB and 200 kB are collected into one Zip archive per site (eliminating all identical duplicates of a given page within a site).

[1]David Graff, Christopher Cieri (2003): English Gigaword; LDC Catalogue number LDC2003T05

This process yields 48GB of compressed data (corresponding to 228 GB uncompressed) in case of the *web-dmoz* crawl, and 124 GB (540 GB uncompressed) in case of the *web-news* crawl, see also table 2.

2.2 Language and Charset Detection

For the material going into classical single-source corpora, it is very simple to make sure that the textual content is from exactly one language, encoded in a uniform fashion. In contrast, textual material from the World Wide Web exists in multiple languages, and – at least for non-English material – is represented in a variety of encodings.

The top-level domain of a web site gives a first approximation of what the language of its document may be, but is usually not very reliable: The top-level domain often correlates with its country of origin, and hence the most likely preferred language of its creator, but needs not be predictive for the language of the documents on that site. Foreign-language and mixed-language sites (even mixed-language documents, such as multilingual forums, or pages where text in one language is discussed in another) are relatively frequent. In addition, the information provided in headers by an HTTP server or declared in a page meta tags is often wrong and hence unreliable as a source of information about page encoding and language.

For accurate information, character encoding detection and language detection based on page content are required. Our approach for character encoding detection relies on the heuristics of IBM's International Components for Unicode (ICU) library,[2] while we use a customized approach for language detection that yields more adequate results for mixed-language documents.

The ICU encoding detector combines heuristics such as checking for illegal sequences for multi-byte encodings, and statistics including byte n-gram frequencies for one-byte encodings for different languages. Pages with character encoding identified with low probability (below 55%) are discarded.

Detecting the language(s) of the content is rendered more difficult by mixed-language documents as well as the presence of other-language boilerplate or nontextual data. As a result, both the exact type of heuristic used (character-level or word-level, as detailed below) and the relative ordering of language and boilerplate detection plays a role.

Two kinds of heuristics are used for language detection in general: One is using character-level properties by building a frequency profile of character n-grams and comparing this frequency profile to a reference distribution [CT94]. The other uses word-level properties of a text by using a list of common function words (which should occur in any linguistically interesting text) and makes assumptions regarding their distribution. In the latter category, [BK06] require the extracted text to contain at least 25% of function words, as well as having at least 30 tokens, and 10 different types of function words from a pre-determined list with 140 entries.

While the function word approach gives high precision in general, we found that the recall of language filtering does suffer and the resulting collection might shrink too drastically. In particular, pages that contain small amounts of useful text tend to be thrown out completely even if they constitute valid linguistic material. Because it uses global word statistics, word-based language detection is also sensitive to different-language boilerplate and must be applied after boilerplate parts of the page have been removed.

The character n-gram approach is more robust in terms of language genres, styles and domain variations, presence of the boilerplate, and size of the content. It is also likely to recognize at least one of the languages in mixed language pages, which makes it more suitable in our eyes than the word-based approach.

In our processing pipeline, we first apply a relatively loose filter in order to eliminate all the documents that are clearly non-German, keeping only German-language and mixed-language documents. The loose filter is based on character trigrams, as described by [CT94]. This first language filter is applied right after the character encoding detection. It filters out 48% of the pages in case *web-dmoz*, and 6% of the pages for the *web-news* crawl. Pages that were filtered out either contain no textual content at all, have a character encoding that could not be recognized, or were clearly identified as being from a different language.

The second step of language filtering happens after boilerplate removal and can use more precise information. In this filtering step, we apply character-level language detection to each individual block of text (corresponding roughly to one HTML paragraph). For small blocks (of less than 60 characters) and for the blocks with low language similarity, we additionally use functional word counts and the language detected for neighboring blocks in our classification.

In the case of our Web corpora, our language detection finds a significant number of mixed-language documents (numbers from *web-dmoz*):

- 80% of pages contain only German blocks.
- 11% of pages contain almost exclusively German (more than 90 % of all blocks are identified as German) .
- 8% of pages contain mostly German (more than 50% but less than 90% German content).
- 1% of pages contain mostly non-German text (less than 50% is recognized as German despite having been classified as German by the first filter language detector). These documents contain about 36% of German, 47% of English and 17% of other-language content.

For the pages containing more than 50% of German content, only the in-language blocks are kept and the remaining text is discarded. As a side-effect, much non-linguistic content, such as URLs, addresses, long list of names, math expressions, etc., which occurs in its own block, also gets discarded. The pages containing less than 50% of German content are removed completely.

[2] http://userguide.icu-project.org/conversion/detection

2.3 Duplicate and Boilerplate Detection

In order to use Web pages as a source of linguistic content, it is necessary to detect, and filter out, boilerplate text – navigation or decorative elements, advertisements, copyright notes, etc. Boilerplate text snippets would hinder both subsequent analysis steps (such as language detection) as well as distort the statistics (including distributional similarity information) of the final corpus. For similar reason, duplicated text – be it teasers for other articles, extractive summaries, or self-plagiarized content on a web page – should be removed from a Web corpus.

The most popular technique for doing boilerplate elimination, as used in the deWaC/ukWaC corpora [BBFZ09] is to find the tag node with the best ratio of (word) token and HTML tag density, and select all other segments as boilerplate. The authors point out that the approach does not handle appropriately a boilerplate in the middle of the informative content, and also can have problems at the span margins.

More refined methods include supervised learning on structural HTML features of shallow features describing a segment [KFN10], but also include computationally expensive methods such as modeling vision/position-based information [CYWM03] or wrapper induction to reconstruct formatting templates based on frequently used patterns [VdSP$^+$06].

In our case, we wanted to be able to process a wide range of genres (including, for example, forums, or user comments on other pages). We also wanted to make as few assumptions as possible on the kind of templating system used for particular pages or sites (or, similarly, absence of a templating system). To reach this goal, we implemented a content-sensitive approach to boilerplate removal that (unlike [VdSP$^+$06]) only makes very basic assumptions on the formatting and document structure.

Our approach relies on the idea that boilerplate (which may or may not be recognizable as part of the navigation based on its HTML markup) is very likely to consist in textual templates (**site patterns**). In the actual pages we would find, the site patterns are either repeated verbatim, or with several gaps filled by content that varies from pattern mention to the next (e.g., dates and user names).

Since the most reliable way to detect such site patterns is to look for repetitions in candidate patterns, there is some amount of interaction between boilerplate removal and identification of partial duplicates: Boilerplate removal may be confused by duplicate content, whereas in turn, duplicate content identification may be confused by remaining boilerplate. To mitigate these interaction problems, we interleave boilerplate filtering and near-duplicate detection: The first step uses a content-insensitive boilerplate filter, which is based on link density. In the second step, we perform the detection of near-duplicate pages within the site. The last step relies on the induction of site-specific boilerplate patterns for the content-sensitive detection of boilerplate text.

In the first step (link density filter), the number of tokens within an `<a>` tag is divided by the total number of tokens in the block. Our link density filter removes blocks with link density values above 0.33, which eliminates most navigational elements while leaving uncertain cases for later (content-sensitive) examination.

	link & text density			*l.d. + site patterns*			err.red.
	P	R	F1	P	R	F1	
Doc. 1	0.60	0.50	0.55	0.97	0.81	0.88	73%
Doc. 2	0.87	1.00	0.93	0.98	0.98	0.98	71%
Doc. 3	0.89	0.79	0.84	0.99	1.00	0.99	93%
Doc. 4	0.84	0.97	0.90	0.97	0.89	0.93	29%
Doc. 5	0.44	0.91	0.59	0.87	0.98	0.92	80%

Table 1: Content-sensitive boilerplate removal: Evaluation on forum HTML content

For the final step of **finding site patterns** and removing the corresponding boilerplate, we found that templates can be both as simple as blocks that repeat with an exact sequence (continuous repeats), or as complicated as blocks repeating with a sequence containing one or several gaps (discontinuous repeats). The gaps can be filled with variable length sequences that vary from pattern mention to mention (e.g. dates and nicknames, or strings such as "*On xxx yyy, zzz posted:*"). Some patterns can look like natural language text, can be lengthy, and may not be formatted specifically, thus presenting difficulty to shallow and purely structure-based approaches.

From each individual web site[3] we extract both continuous and discontinuous patterns using the GAAL library [Kis11], which performs pattern extraction using linearised suffix trees. We limit the extracted patterns to a size of not more than 500 characters, with a maximum number of three gaps, with the gaps not exceeding nine tokens in length. We consider all patterns with frequency above one to be boilerplate if the number of tokens in the repeat part(s) of a pattern exceeds the number of tokens in the gap part(s) of the pattern.

We validated our approach to content-sensitive boilerplate detection using five samples of forum HTML from different sites, which contains a mixture of quoting, signatures, and template strings intermingled with the content. Comparing boilerpipe's content-insensitive approch with our own approach (cf. Table 1), we found that using the site patterns provided substantial quality improvements.

Duplicated content (whether boilerplate or reused informative content), if present, will confuse the duplicated pattern detection. To avoid this, a **detection step for near-duplicates** (within each site) is run after the generic (link density-based) boilerplate reduction, but before the induction of site-specific boilerplate. This within-site deduplication step uses an approach identical to the near-duplicate elimination that is run on the whole corpus to find content duplicated across sites.

Deduplication uses the general approach of shingling described by [BCFM00], in that it computes a min-hash sum-

[3] A web site corresponds to one particular second-level domain, or a third-level domain inside a generic second-level one such as `.ac.at`.

	web-dmoz	web-news
raw crawl (compressed)	298GB	124GB
zip files (compressed)	48GB	112GB
zip files (uncompressed)	228GB	540GB
document counts		
before language detection	8.2M	11.4M
after language detection	3.9M	10.7M
near-duplicate removal	1.0M	5.3M
boilerplate removal	845k	4.4M
cross-site deduplication	602k	3.6M
Corpus size (tokens)	360M	1700M

Table 2: Corpus sizes after each filtering step

mary of the shingles (commonly: n-grams, for some n of 5 or 6) and uses these summaries to approximate the weighted Jaccard similarity measure between those shingles.

For our corpus pipeline, we do not use plain 5- or 6-grams as shingles, instead using an approach that creates a smaller number of still-distinctive shingles. Shingles are created based on "spots" that are composed of a function word (using articles, auxiliaries, and modal verbs in our list) and a trigram of the following three content words, as in the SpotSigs approach [TSP08]. This shingle-based representation together with shingle counts is likely to be an adequate representation for the informative content of the page.

From the shingle representation of each page the fixed length min-hash signature of the page is computed – in our case we use a min-hash signature of length 192. Min-hashing preserves the expected similarity: It has the useful property that the probability of a match at any signature index between two signatures corresponds to the Jaccard similarity of the signatures.

This property of min-hashing allows the application of locality sensitive hashing (LSH) as a next step to efficiently identify candidate near-duplicate pairs. Min-hash signatures are split into a number of bands (in our case 32) and for each band the corresponding segment (in our case segment length is 192/32=6) of the min-hash signatures is hashed to the large number of buckets. We then consider any pair that hashed to the same bucket for any of the hashings (bands) to be a candidate pair.

The SpotSigs library [TSP08] does not use straight min-hashes for a page, but uses a combination of min-hash (which represents several shingles in one hash value and is suitable for detecting exact repetition) and locality sensitive hashing (LSH), which computes a low-dimensional approximation of the content that is suitable for approximate similarity computation.

In our case, a sequence of $k = 6$ shingles (spots) is summarized into one min-hash signature, and the signatures of one document are reduced to a representation containing $l = 32$ buckets.

Locality sensitive hashing is an efficient way to approximate the similarity of the spot signatures of documents; for the actual deduplication of candidates, the weighted Jaccard similarity is calculated and any pair of documents with a similarity score of at least 44% is treated as a near-duplicate.

In the case of within-site deduplication, we reduce each cluster of mutual near-duplicates to one page that is left in the corpus and discard the other near-duplicates. For the duplicate removal across the whole corpus, we completely remove any content that has near-duplicates coming from more than five distinct sites, effectively removing this kind of frequent content. This is motivated by the observation that pages that repeat too frequently across the web present no or little linguistic interest, and frequently consist of contact information, statements of ownership and authorship of a certain text, etc.

Even using the spot-signatures approach for representing document content, boilerplate in a document representation could distort the hashed representation because of the added material. In such a case, duplicates from different sites are not recognized as such, or the boilerplate results in pages falsely recognized as duplicates. Hence, our approach of interleaving boilerplate detection (with a markup-driven first step and a content-sensitive second step) with deduplication is necessary to reach the best results.

In our corpora harvested from the Web, we find that 81% of pages are discarded with 719212 pages left in case of DMOZ crawl, and 64% of pages are discarded with 3865269 pages left in case of the news focused crawl.

3. LINGUISTIC PREPROCESSING

After extracting the raw text from a Web crawl, we can apply general-purpose NLP tools in order to extract linguistic information; In the case of Web corpora, the larger variation of genre and domain makes it necessary to use tools that are more robust than when processing only newspaper text. It is also necessary for the approaches to be reasonably efficient in order to process very large corpora.

3.1 Tokenization

From our initial evaluation of existing Web corpora, mistokenized words as well as general encoding problems were one of the concerns that we thought would most impede the final corpus quality: A mis-encoded or incorrectly tokenized word usually poses more difficulty to tagging or parsing models, especially where those models use word statistics.

Because tokenizers are not generally considered important within NLP research, it is very hard to find a common evaluation; furthermore, the models distributed with libraries such as OpenNLP exhibit rather poor performance not only because they do nothing to capture language-specific idiosyncracies but also because they are trained on inadequate data (detokenized treebank data instead of normal text).

In the case of German (where we can use data from the TüBa-D/Z treebank together with the raw text from the *tageszeitung* newspaper), tokenization exhibits nonlocal properties in the case of compounds such as '*"Sicherheits"-Truppen*' or '*(Schaden-)Freude*'. In examples such as these, quotes or parentheses have to be kept together when they are part of a compound, but not when they connect normal text. OpenNLP's approach based on local classifiers would always

introduce errors in such case, whereas a rule-based tokenizer can reasonably be expected to solve such a problem.

If one wanted to reach perfect agreement with the treebank tokenization, additional semantic knowledge would be necessary to distinguish coordination-like use of dashes ('*5-10 Gramm*'), which has to be represented as separate tokens, with dashes as token-internal separator in room and telephone numbers ('*Raum 5-10*').

Especially for multi-source data, but also for normal newspaper text, it has to be kept in mind that Unicode contains a multitude of alternatives for various nonalphabetic characters, especially dashes and quotes. Common processing tools such as part-of-speech taggers and parsers are best used with text that is similar to the treebank data they have been trained on. In particular, treebank text normally contains only normal quotes and dashes instead of reproducing all and any typographic variation. In order to provide reasonable input to the rest of the processing chain, we took a very pragmatic approach and normalized all quotes and dashes to their standard ASCII forms. We find that the additional typographic information is not important enough to accept (or deal with) the resultant fallout in the rest of the linguistic processing pipeline.

In order to provide the tokenization for our Web corpora, we used a rule-based tokenizer, which first performs an initial segmentation of a text into tokens using a set of regular expressions. In a subsequent step, the tokenizer revises some of these tokenization decisions and also performs sentence boundary detection.

While Unicode-compatible, the regular expressions library included with the standard Java library is not very performant since it needs to support the backtracking needed for a Perl-compatible treatment of regular expressions; as a result, very complicated regular expressions (including cases typical for tokenization, such as lists of abbreviations, or nesting of alternatives for different sub-parts of a token) show some performance deterioration. The `dk.brics.automaton` library[4] allows to use deterministic finite automaton (DFA) representation for matching of expressions in addition to supporting Unicode and the corresponding character classes.

3.2 Morphology and Parsing

For fixed word order languages such as English, approaches to induce semantic information from raw text have been shown to be feasible, even if they do not always reach the performance of more elaborate approaches. In general, however, deeper linguistic information including lemmatization and phrase structure or dependency parses can be expected to yield much better generalization behaviour. This is especially true for languages such as German, which combine a more flexible word order with rich morphological inflection behavior.

For English and its fixed word order, [CM02] have proposed the use of a chunker based on sequence tagging and subsequent chunk linking in order to gain information on grammatical relations that can be used as describing context for nouns, in particular premodifying adjectives, subjects, and direct objects. In the case of German, syntactic properties of the language make a chunking-based approach problematic, more so as a simple chunk linker would be unable to recover most argument relations (subject, object) in a language with somewhat free word order.

As a fast compromise between (insufficiently powerful) chunking and (slow) full parsing, we use a pipeline that relies on deterministic dependency parsing to provide complete dependency parses at a speed that is suitable for the processing of Web-scale corpora.

The parsing model is based on MALTParser, a transition-based parser, and uses part-of-speech and morphological information as input. Morphological information is annotated using RFTagger [SL08], a state-of-the-art morphological tagger based on decision trees and a large context window (which allows it to model agreement more accurately than a normal trigram-based sequence tagger). While transition-based parsers are quite fast in general, an SVM classifier (which is used in MALTParser by default) becomes slower with increasing training set. In contrast, using the MALTParser interface to LibLinear by [Cas09], we can reach a much larger speed of 55 sentences per second (against 0.4 sentences per second for a more feature-rich SVM-based model that reaches state of the art performance).

For lemmatization, the original version of deWaC uses the lemmatization component of TreeTagger [Sch95], which is easily the most popular lemmatization component for German since it is freely available (at least for noncommercial purposes) and easy to use. Several weaknesses of TreeTagger make it less than ideal for the use in a pipeline for learning lexical semantic information: firstly, TreeTagger uses a fixed lexicon and provides no treatment for unknown words; secondly, it provides no solution for reattaching separable verb prefixes, yielding only partial verb lemmas in many cases; thirdly, it always provides the same lemma for one word/POS combination and does not use morphosyntactical information for disambiguating lemmas.

In our case, we use the syntax-based TüBa-D/Z lemmatizer [VBHT10], which uses a separate morphological analyzer and some fallback heuristics. The compositional and derivational SMOR morphology [SFH04] serves to provide morphological analyses for novel words. For unanalyzed novel words that are not covered by SMOR, the lemmatizer falls back to surface-based guessing heuristics. It uses morphological and syntactic information to provide more accurate lemmas; In addition to dependency structures, the morphological tags from RFTagger as well as global frequency information are used.

4. EVALUATION

For English newspaper text, [CM02] provide an account of the relation between corpus size and processing algorithms used on one hand, and the quality of the learned thesaurus as well as the amount of computation that is necessary for the preprocessing on the other hand.

Because it provides a unified framework for evaluating a combination of the (kind and amount of) text used as input,

[4]http://www.brics.dk/automaton/

as well as the processing tools used, Curran and Moens' experimental framework provides an excellent way to compare different ways of acquiring text (such as a fixed amount of newspaper text versus a much larger amount from a Web corpus). It also allows to explore the most efficient way to deal with resource constraints: In the case of a fixed amount of computing power being available, they find that window-based approaches, despite being computationally very cheap, yield results that are significantly inferior to applying more elaborate techniques on a smaller amount of text. In comparison, their approximate chunk linking technique works well enough for English that it may be preferred to a full parser when computation time is limited.

4.1 Single-source Corpora

Besides Google's n-gram dataset for German, which necessitates a different approach from the corpora we use here (as it is only usable for shallow pattern extraction), we use a 600 million word subset from the deWaC corpus of [BK06] and two single-source corpora that would be used alternatively to a Web corpus: on one hand, about 200 million words of newspaper text extracted from the years 1986-1999 of *die tageszeitung*; on the other hand, a recent Wikipedia dump (amounting to about 400 million words) that has been cleaned using the Tanl Wikipedia extractor[5] before applying our linguistic processing pipeline.

4.2 A German 'Gold Standard' Thesaurus

[CM02] use several existing thesauri for English, namely the Macquarie, Roget's and Moby thesauri, to create a merged thesaurus that provides a gold standard for "matching synonyms" that a system should retrieve. To our best knowledge, such thesauri do not exist for German (and may be nonexistent or hard to get for other languages), which makes it necessary to adapt their approach.

Using GermaNet 6.0 [HH10], we found that looking for exact synonyms (i.e., words that occur in one of the synsets of the target word) yields very narrow synonym lists, which are often focused on non-dominant senses of a word. Thesauri, in contrast, contain broader lists of related terms and also include near-synonyms (i.e., cohyponyms). In order to get lists of related terms that are better suited for evaluating semantic similarity measures than just using synonyms, we adapted the method of *radial glosses* [Gur05] for our purpose.

Gurevych discusses a number of possibilities to extract super- and/or subordinate and coordinate terms from a wordnet; for our purpose, we aimed at words of roughly the same level of generality and sharing an appropriately high number of properties. To implement this, we ordered neighbouring synsets using a path distance measure and added the words of the closest unused synset until 30 words have been retrieved. For the distance measure itself, we postulate higher distances for links between very general terms (especially the top three layers of GermaNet's noun hierarchy), and for links that have a large number of sister links (such as those that link a particular kind of animal to the individual species that make up this kind).

[5] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

To yield a testing set for the evaluation of corpora and processing pipelines, we took samples of target words consisting of the top 30 items (by frequency in the TüPP-D/Z corpus), and additional 30 items from the frequency ranges of 10-20, 5-10, 2-3, 1-2 occurrences per million. (For a 200 million word corpus, these would correspond to around one million occurrences among the most frequent words, and between 3000 and 300 occurrences of the sampled words with lower frequency of occurrence).

4.3 Distributional Thesaurus Construction

In order to compare corpus quality via the intermediary of distributional semantics tasks, we create vectors of co-occurring lemmas based on various methods. The vector entry for each co-occurring lemma is weighted based on the conservative pointwise mutual information estimate of [PR04]. To retrieve semantically similar words based on the vector representation, we then use a similarity measure based on the Jensen-Shannon divergence to compute a ranked list of (frequent) distributionally similar terms. The Jensen-Shannon divergence was found by [PKS04] to yield the best results among several alternative vector similarity functions.

For computing the vectors, we extracted related lemmas using two methods: The first method extracts weighted collocations in the way described by **Padó and Lapata's method** [PL07], which weights (same-clause) collocates by their distance in the dependency graph (yielding 1, $\frac{1}{2}$, or $\frac{1}{3}$ for nodes that are 1, 2, 3 edges away, respectively).

The second method uses selected **grammatical relations** extracted from the dependency parses and uses the frequency profile of terms co-occurring with one specific relation to construct the word vector. We found that premodifying adjectives (the `ATTR` label in the dependency parses) performed best among all relations, with accusative objects (`OBJA`) also doing fairly well.

As a means to make use of Google's German n-gram data, we used **shallow patterns** consisting of a sequence of one known adjective and one known noun, or a coordination of two known nouns, using a simple precomputed lemma mapping.

4.4 Evaluation Results

Using a distributional similarity evaluation gives us a way to compare the effect of varying the text basis as well as the methods for computing the actual distributional similarity values. Looking at the results from table 3, we see that our evaluation results are in fact better for the `tageszeitung` corpus (TüPP-D/Z), despite avoiding any kind of explicit bias. [6]

Table 6 compares different methods for collocate extraction both on dependencies from TüPP-D/Z corpus as well as pat-

[6] Implicit sources of bias may be seen in the frequency lists used for choosing words for addition to GermaNet, which have been constructed from varied sources including *tageszeitung* and the German Wikipedia, but also by the fact that the frequency-based sampling of words was performed based on frequency data numbers from the `tageszeitung` corpus.

corpus	200m	400m	600m
TüPP-D/Z	0.69	—	—
Wikipedia	0.67	0.68	—
web-dmoz	0.66	0.66	—
web-news	0.64	0.66	0.66
deWaC	0.68	0.70	0.69

Table 3: InvR scores for different subcorpus sizes, Padó and Lapata's method

corpus	200m	400m	600m
TüPP-D/Z	0.88	—	—
Wikipedia	0.77	0.84	—
web-dmoz	0.77	0.84	—
web-news	0.80	0.86	0.89
deWaC	0.82	0.87	0.90

Table 4: InvR scores for different subcorpus sizes, premodifying adjectives

corpus	200m	400m	600m
TüPP-D/Z	0.71	—	—
Wikipedia	0.62	0.72	—
web-dmoz	0.58	0.64	—
web-news	0.63	0.72	0.75
deWaC	0.61	0.71	0.74

Table 5: InvR scores for different subcorpus sizes, accusative objects

terns extracted from Google's German n-gram dataset. It is quite surprising that the numbers extracted from the larger n-gram dataset are substantially lower, since the n-gram corpus was constructed using about 500 times more data (100 billion words versus 200 million words). It is also notable that the grammatical relation approach is decidedly superior to the one using simple (dependency-)syntactic neighbourhood collocates.

Looking at the lowest frequency range shows that Padó and Lapata's method of collecting collocates independent of a particular location is much less sensitive to sparse data problems – indeed the results do not seem to improve with corpus size – while the grammatical relation approach as well as the shallow n-gram patterns decidedly show deficiencies in the lowest frequency range. (In the frequency range from 1-2 occurrences per million tokens, Padó and Lapata's approach is the strongest feature extraction method for all corpora except for the newspaper corpus, where it ranks behind premodifying adjectives).

The grammatical relation approach indeed profits from larger corpus size, leading to improvements even over the newspaper corpus in the case of the larger Web corpora. One important question would be if the frequency selection in Curran and Moens' work (which we used as a starting point for our own gold standard) is really typical for real-world applications in that it contains very few low-frequency terms, hence being less sensitive to data sparseness than other tasks would be.

method	corpus	InvR
PL07	TüPP-D/Z	0.69
Adj-N	web-news	0.89
Adj-N	ngrams	0.57
N-and-N	ngrams	0.51

Table 6: Parsed corpora vs. n-gram patterns

5. SUMMARY

In this work, we have extended the evaluation framework for English corpora and processing used by [CM02] for use with German Web corpora and have used it to compare different approaches to building corpora and extracting features for distributional semantic models. We have used this evaluation framework to provide evidence for the quality of our pipeline for building textual corpora from samples of the World Wide Web. The pipeline incorporates several improvements on the state of the art. In particular, we propose a novel approach to boilerplate removal which uses the crawled data for one site to realize content-sensitive filtering of boilerplate-heavy text, and present state-of-the-art techniques for linguistic processing of the resulting text.

The results from our evaluation procedure show that producing and using Web corpora in general yields better results than using shallow pattern search on Google's German n-gram dataset, and further that different methods for the construction of vector representations show considerable difference in their sensitivity to the amount of corpus data available.

For researchers aiming to create Web corpora in other languages than German, a number of insights can be formulated: One is that, even in the presence of Google's n-gram collections for many languages, a Web corpus affords more complex linguistic processing and may be considerably more useful for many purposes.

Secondly, it is apparent that methods such as that of Padó and Lapata are at a disadvantage to the explicit modeling of grammatical relations as soon as there is sufficient data for the latter to work; in our evaluation, as well as that of Curran and Moens [CM02], the sampling of test words is biased towards higher-frequency items and may overemphasize this tendency even at moderate corpus sizes. Lastly, the processing pipeline based on deterministic dependency parsing with a linear classifier (together with a step of morphological analysis) that we use here scales well to Web-scale corpora and may present an appropriate choice for many languages.

Acknowledgements The research reported in this paper was partially funded by the Deutsche Forschungsgemeinschaft as part of collaborative research centre (SFB) 833. The authors would like to thank Emily Jamison for helpful comments on an earlier version of this paper.

6. REFERENCES

[BBFZ09] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The Wacky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*,

43(3):209–226, 2009.

[BCFM00] Andrei Z. Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630 – 659, 2000.

[BK06] Marco Baroni and Adam Kilgariff. Large linguistically-processed web corpora for multiple languages. In *EACL 2006*, 2006.

[BL08] Marco Baroni and Alessandro Lenci. Concepts and word spaces. *Italian Journal of Linguistics*, 20(1):129–156, 2008.

[Cas09] Sofia Cassel. MaltParser and LIBLINEAR - transition-based dependency parsing with linear classification for feature model optimization. Master's thesis, Uppsala University, 2009.

[CM02] James Curran and Marc Moens. Scaling context space. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[CT94] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[CYWM03] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In *Proc. 5th Asia Pacific Web Conference (APWeb)*, volume 2642 of *LNCS*, 2003.

[FP10] Manaal Faruqui and Sebastian Padó. Training and evaluating a German named entity recognizer with semantic generalization. In *Proceedings of KONVENS 2010*, 2010.

[Gur05] Iryna Gurevych. Using the structure of a conceptual network in computing semantic relatedness. In *IJCNLP 2005*, 2005.

[HH10] Verena Henrich and Erhard Hinrichs. GernEdiT - the GermaNet editing tool. In *LREC 2010*, 2010.

[KFN10] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, 2010.

[Kis11] Alexander Kislev. Automated multilingual alignment of discontinuous sequences. Master's thesis, Seminar für Sprachwissenschaft, Universität Tübingen, 2011. See also http://code.google.com/p/gaal/.

[LC06] Vinci Liu and James R. Curran. Web text corpus for natural language processing. In *EACL 2006*, 2006.

[Mül04] Frank Henrik Müller. Stylebook for the Tübingen partially parsed corpus of written German (TüPP-D/Z). Technischer Bericht, Seminar für Sprachwissenschaft, Universität Tübingen, 2004.

[PKS04] Viktor Pekar, Michael Krkoska, and Steffen Staab. Feature weighting for co-occurrence-based classification of words. In *COLING'2004*, 2004.

[PL07] Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[PR04] Patrick Pantel and Deepak Ravichandran. Automatically labeling semantic classes. In *HLT/NAACL 2004*, 2004.

[Sch95] Helmut Schmid. Improvements in part-of-speech tagging with an application to German. In *Proc. ACL-SIGDAT Workshop*, 1995.

[SFH04] Helmut Schmid, Arne Fitschen, and Ulrich Heid. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of LREC 2004*, 2004.

[Sha06] Serge Sharoff. Open-source corpora: using the net to fish for linguistic data. *International Journal of Corpus Linguistics*, 11(4):435–462, 2006.

[SL08] Helmut Schmid and Florian Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *COLING 2008*, 2008.

[TSP08] Martin Theobald, Jonathan Siddharth, and Andreas Paepcke. SpotSigs: robust and efficient near duplicate detection in large Web collections. In *Proc. 31st SIGIR conference on Research and development in information retrieval*, pages 563–570. ACM, 2008.

[VBHT10] Yannick Versley, A. Kathrin Beck, Erhard Hinrichs, and Heike Telljohann. A syntax-first approach to high-quality morphological analysis and lemma disambiguation for the TüBa-D/Z treebank. In *Proceedings of the 9th Conference on Treebanks and Linguistic Theories (TLT9)*, 2010.

[VdSP+06] Karane Vieira, Altigran A. da Silva, Nick Pinto, Edleno S. de Moura, Ao M. Jo, and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM'06)*, 2006.

[Ver08] Yannick Versley. Decorrelation and shallow semantic patterns for distributional clustering of nouns and verbs. In *ESSLLI 2008 Workshop on Distributional Lexical Semantics*, 2008.