Feed Corpus : An Ever Growing Up-To-Date Corpus

Akshay Minocha
IIIT Hyderabad
India
akshay.minocha@students.iiit.ac.in

Siva Reddy
Lexical Computing Ltd
United Kingdom
siva@sketchengine.co.uk

Adam Kilgarriff
Lexical Computing Ltd
United Kingdom
adam@lexmasterclass.com

Abstract

Corpus-based lexicography has to keep its pace with the language evolution by using up-to-date corpus. In this paper we propose a method for collecting corpora which is ever growing and up-to-date with the language. We make use of social media to discover sources of dynamic content like blogs and news websites. Most such websites provide a short summary of their content change in a separate page known as feed, which we use to keep track of new content. We collect millions of such feeds using social media. We design a scheduler which rank these feeds based on their frequency of update and the amount of text extracted per retrieval. Based on the rank, we periodically crawl these feeds and add any new content generated to our corpus collection along with the temporal information. Thus the corpus is dynamic and nearly up-to-date with the language evolution. In a month's duration, we collected a corpus of size 1.36 billion words for English, which after deduplication resulted in 300 million words, demonstrating the potentiality of this approach.

1 Introduction

The aim of corpus-based lexicography is to study the behaviour of words based on their usage in language. Since the success of COBUILD project, many static collections of electronic corpora such as BNC, the WaCky Corpora (Sharoff, 2006; Baroni et al., 2009; Kilgarriff et al., 2010), and recently TenTens' (Jakubíček et al., 2013) came into existence. The list has been increasing each year and soon overwhelming with too many choices for the publishing agencies. With the change in corpus choice, the lexicographic cycle has to be rerun, although the major part of the corpus contains the language usage already captured in the past. Instead what if the corpus is dynamic and provides a snapshot of language usage between any two desired periodic points?

In this paper, we propose a solution for creating dynamic corpora which is up-to-date with the language and increasing every day. There were past proposals on dynamic corpora like monitor corpus (Clear, 1987) which is a continuous stream of corpus rather than a static collection. With the current advances in web technology, building such corpora is not far from achievable.

Traditional methods of static corpora collection involve crawling through billions of web pages periodically. Keeping track of changes in such a huge network is pain-staking, and visiting each website again in the the next crawl anticipating for new content is cost-inefficient. Technology such as feeds partly address this problem by providing a mechanism to detect new content. A feed is a collection of temporal updates in a website. The presence of a feed in a website is a plausible indication that the website posts new content once-in-a-while. We aim to fish millions of feeds from the Internet and keep track of the changes and add new content to the corpus collection whenever a website is updated. But how to discover millions of feeds from several millions of websites?

With the advent of social media like Twitter, latest content is one click away. Currently 340 million tweets are published per day[1]. This is expected to increase tremendously given that 87% of all the tweets are posted in the past 24 months (Leetaru et al., 2013). Most newswires, blogs and other frequently updated websites post tweets whenever new content is generated. Additionally, millions of people share hyperlinks of posts containing their newly found information. These tweets are potential sources to the websites which

[1]Twitter Statistics - http://blog.twitter.com/2012/03/twitter-turns-six.html

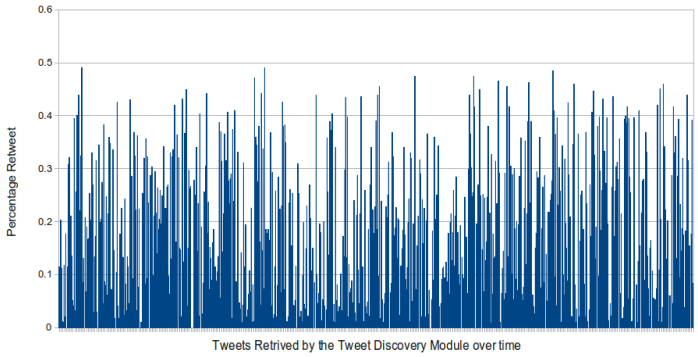

Figure 1: Number of retweets on the scale of 0-1 in the month of March 2013

contain feeds, and once a feed is found, we track it for new content updates. We piggyback Twitter by retrieving tweets containing hyperlinks using keyword search queries, thus in-turn piggybacking on the world's population to find websites that frequently update, saving ourselves from the tedious task of crawling the whole web.

To keep track of millions of feeds, a simple sequential feed aggregator is not sufficient since different feeds have different update frequencies and a long looping delay will lose novel content from frequently updated websites. We design a scheduler which sets a priority for each feed based on its frequency of update between posts and the amount of content generated per retrieval. Based on the scheduler priorities, we fetch feeds and check if any new posts are posted on the website. The new content from posts is added to the corpus collection, resulting in an corpus ever-growing and relatively up-to-date corpus compared to the static corpora.

In our pilot experiment on English run for the month of March 2013, we collected around 1.36 billion words, which after duplicate removal resulted in 300 million words. Currently, we have up to 150 thousand feeds and still growing.

2 Method for Feed Corpus Collection

We briefly outline the steps involved in our approach for collecting an ever-growing up-to-date corpus, the Feed Corpus.

1. **Feed Discovery:** For every 15 minutes, we run keyword search queries on Twitter to retrieve tweets containing hyperlinks. The keywords are chosen such that they result in tweets with hyperlinks to potential websites containing feeds.

2. **Feed Validation:** The domains of the hyperlinks in the tweets from Step 1 are validated for the presence of a feed. If a feed is found, the feed is added to the list of valid feeds or else the domain is blacklisted.

3. **Feed Scheduler:** A feed contains up to top 10 last new posts along with the timestamps at which these posts are created. An initial priority is set to the feed based on its frequency of update and placed in the priority queue.

4. **Feed Crawler:** The top most feed in the priority queue is fetched and verified for any new posts. If a new post is found and is not already in the corpus (content deduplication), the post is added to the corpora collection along with its time stamp. The priority of the feed is updated based on its previous priority and the new average time lapse between posts.

In the coming subsections, we describe each step of feed corpus collection and the corresponding challenges involved.

2.1 Feed Discovery

We design Twitter search queries containing one of the keywords like news, business, arts, games, regional, science, shopping, society etc. We restrict the search results to tweets containing hyperlinks in them. All these queries are run in periodic intervals of about 15 minutes. We ignore retweets since these likely result in duplicates. Figure 1 displays the number of retweets for our search query results on the scale of 0-1 for a duration of a month. Almost 17 percent of the query results constitute retweets.

The hyperlinks we find in the tweets are not the feed links themselves but to the posts that could belong to a domain containing feeds. We validate the feeds in the next step.

2.2 Feed Validation

Given a hyperlink of a web post, this module determines the feed links associated with the hyperlink. Hyperlinks shared on Twitter are unlikely to be feed links themselves. We use simple heuristics to determine the feed link of a hyperlink. In the first step, the hyperlink is verified if it is a feed link by itself. If not, we analyse the main domain of the hyperlink and find out references to any valid feed hyperlinks mentioned in the source. We use meta-data commonly associated with the feeds to identify feeds e.g. `type=application/rss+xml`. If the main domain has no feed links, we analyse the child hyperlink one step above the main domain along the initial hyperlink. For example, if the initial hyperlink is `http://ab.cd.com/ef/ij/kl.rss`, the *main domain* is `http://ab.cd.com/` and *one step above main* is `http://ab.cd.com/ef`.

In case if no feed links are found, we blacklist the domain to avoid repeating these steps in future if any of the hyperlinks associated with this domain are seen again.

2.3 Feed Scheduler

Different feeds have different update times. For example, newswire websites post new content every hour, while corporate websites post content once in a day or two, and personal blogs are updated once in a week or a month or even a year. After Step 2, we end up with millions of feeds over time. A simple sequential aggregator is inefficient in tracking updates: if the time gap between visits to a feed is higher than its frequency of update we lose its updates; if the time gap is lower than its frequency of update, we waste bandwidth. A scheduler is crucial to build an efficient crawler. We determine the initial update frequency of a feed by taking an average time gap between the top 10 recent posts. We implement a time-based priority queue where all the feeds are placed in the queue according to the priority. As the time passes the queue moves forward with the first feed in the queue processed by the feed crawler.

Additionally, we aim to avoid crawling websites which frequently change but results in low yield rate e.g. if the website has too much boilerplate or the language is irrelevant.

Threshold θ ($Y * 10^{-2}$)	Crawler Output size (GB)	Final data size (GB)
0.01	221.16	1.59
0.04	162.52	1.54
0.32	72.31	1.48
2.56	14.36	0.90
5.12	6.71	0.72

Table 1: Amount of data collected with different yield rates in the month of March 2013

We use the yield rate as described by Suchomel and Pomikálek (2012)

$$\texttt{yield rate Y} = \frac{\textit{cleaned data size}}{\textit{downloaded data size}}$$

The yield rate signifies the efficiency of fetching data from a website. We discard the domains whose yield rate (Y) is below a threshold (θ). We use jusText[2] (Pomikálek, 2011) for removing boilerplate text from web pages and langid.py (Lui and Baldwin, 2012) to detect language of the page.

Table 1 displays statistics of corpora downloaded with different thresholds θ of yield rates. We choose a yield rate of 0.003 to allow data even from micro-blogging sites.

2.4 Feed Crawler

In this step, we take the highest priority feed in the priority queue and crawl it to detect any new content. If found, the content is added to the corpus collection after verifying if it is a duplicate of any post already present in the data. We use the deduplication tool Onion[3] (Pomikálek, 2011) to remove duplicates. Since it is expensive to run the deduplication tool for every new post, we run the tool in a batch mode after collecting significant amount of new content.

We also update the yield rate and priority rank of the feed in the scheduler. Feeds do not have a constant update time. We observed bursts of activity occasionally and a period of long silence. In order to take past into consideration, we update priorities in a cumulative fashion taking a weighted mean of the past update time and the current update time as new update time. Based on the new update time, we place the feed in the priority queue.

[2] jusText https://code.google.com/p/justext/

[3] http://code.google.com/p/onion/

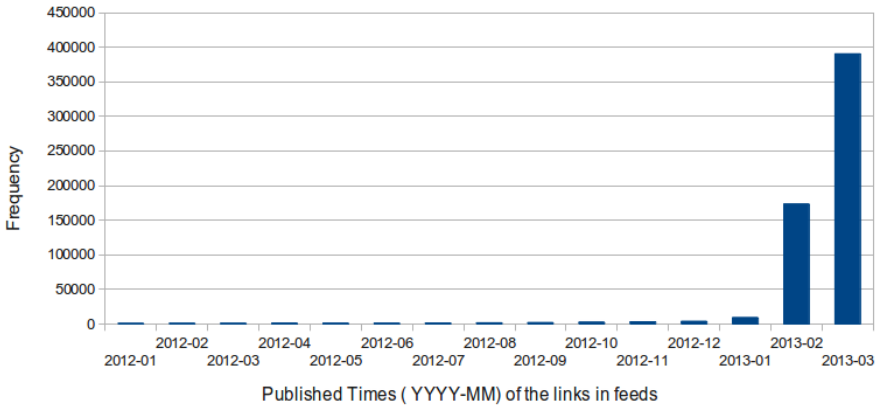

Figure 2: Number of posts from each month in our corpus

3 Results

As a pilot experiment, we ran the above steps for the month of March 2013. The results are preliminary and we hope to extend this work in future. At each iteration of feed discovery, we found around 120 new hyperlinks of which 60% helped in finding valid feeds. We collected around 150,000 feeds in a month which shows Twitter is promising to discover new feed links. On an average we collected around 40 million words per day from the feeds. We collected the corpus from all the posts listed in the feeds, including the posts from previous months of March. Figure 2 displays the number of posts collected for each month. Since we collected feeds from tweets in March, most posts are found to be from March. At the end of the month, we ran deduplication on total corpus of size 1.3 billion words and extracted cleaned corpus of size 300 million words.

4 Conclusion

We presented a simple method for building an ever-growing up-to-date corpora using feeds discovered from Twitter. In a month's duration, we collected around 150,000 feeds and a corpus of 300 million words along with their timestamps of creation. Our preliminary results are promising encouraging us to extend this work for many other languages and over a prolonged period of time. Currently, we only use the temporal information present in the feeds, but in future we also aim to use category tags mentioned in the feeds, thus providing a valuable resource for genre-specific temporal corpora.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Jeremy Clear. 1987. Trawling the language: monitor corpora. *ZURILEX Proceedings. Tübingen: Francke*.

Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlỳ, and Vít Suchomel. 2013. The tenten corpus family. In *International Conference on Corpus Linguistics, Lancaster*.

Adam Kilgarriff, Siva Reddy, Jan Pomikálek, and Avinesh Pvs. 2010. A corpus factory for many languages. *Proc. LREC, Malta*.

Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Eric Shook. 2013. Mapping the global twitter heartbeat: The geography of twitter. *First Monday*, 18(5).

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30. Association for Computational Linguistics.

Jan Pomikálek. 2011. *Removing Boilerplate and Duplicate Content from Web Corpora*. Ph.D. thesis, Ph. D. thesis, Masaryk University.

Serge Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98.

Vit Suchomel and Jan Pomikálek. 2012. Efficient web crawling for large text corpora. In *Proceedings of the seventh Web as Corpus Workshop (WAC7)*, pages 39–43.