Wildcards and Morphological Inflections for the Google Books Ngram Corpus

Jason Mann, David Zhang, Lucille Yang, Slav Petrov and Dipanjan Das
Google Inc.
jcm2207@columbia.edu, dzhang21@gmail.com, ly77@cornell.edu
{slav,dipanjand}@google.com

Abstract

We present a new edition of the Google Books Ngram Viewer, which plots the frequency with which words and phrases were used over the last five centuries; its data encompasses 6% of the world's published books. The new edition adds three features for more powerful search: wildcards, morphological inflections, and case insensitivity. These additions allow for the discovery of previously unknown patterns in the Google Books Ngram data, and further facilitate the study of linguistic trends in printed text.

1 Introduction

The Google Books Ngram Viewer[1] and its corresponding Google Books Ngram Corpus (Lin et al., 2012) are useful tools for the analysis of cultural and linguistics trends through five centuries of data in eight languages. We present an updated version of the Viewer which introduces several new features.

First, users can replace one query term with a placeholder symbol (wildcard, henceforth), which will return the ten most frequent replacements in the underlying corpus for the specified year range. Second, by adding a specific tag to any word in a query, morphological inflections (or variants) will be returned. Finally, the new interface has an additional option to allow for multiple capitalization styles. In addition, this demonstration presents an overhaul of the Viewer's user interface, with interactive features that allow for easier management of the increase in data points returned.

How about giving an example for each of the queries and then having a 3 small figures on the next page that show the results? The examples can come from the about page.

Mention related and prior work here.

While it is obvious how the above searches can be answered via brute-force computation, supporting an interactive application with low latency necessitates some pre-computation. To this end, we provide an overview of our system architecture in section 2 and discuss some of our design choices. We then detail interesting use cases in section 4, which were difficult (or even impossible) to search in the previous versions of the Ngram Viewer that did not handle wildcards in the search queries. Additionally, we detail how the two other aforementioned features introduced in this demonstration paper result in interesting retrieval results. Beyond specific searches, we envision the new functionality of the tool uncovering trends and patterns not readily apparent in the data.

2 System Overview

In this section we present an overview of the Ngram Viewer backend. Before going into the details, we first describe the corpora on which users of this tool can issue queries.

2.1 Ngram Corpus

The Google Books Ngram Corpus[2] provides ngram counts for eight different languages over more than 500 years; additionally, the English corpus is split further into British vs. American English and Fiction to aid domain restriction. This corpus is a subset of all the books digitized at Google, and represents more than 6%(Lin et al., 2012) of all publicized texts in its newest edition. The differences between the first and second ver-

[1] `http://books.google.com/ngrams`

[2] `http://storage.googleapis.com/books/ngrams/books/datasetsv2.html`

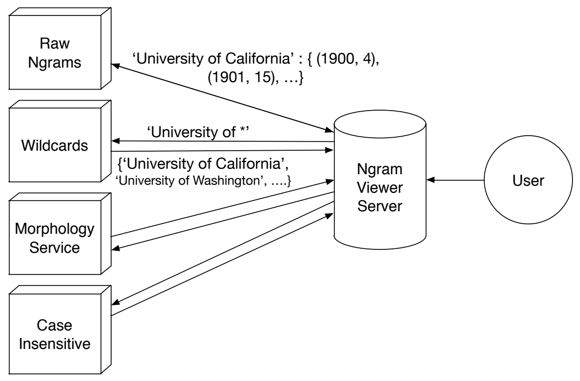
Raw Ngrams
'University of California' : { (1900, 4), (1901, 15), ...}
Wildcards
'University of *'
{'University of California', 'University of Washington', ...}
Ngram Viewer Server
User
Morphology Service
Case Insensitive

Figure 1: Diagram of new system architecture.

sions of the corpus are discussed at length in Lin et al. (2012) and the work in this demonstration does not concern an update with the corpus data, but rather new tools in the Ngram Viewer that can be used to better analyze the latest 2012 corpus.

The current version of the corpus is tagged using the universal part-of-speech tag set outlined by Petrov et al. (2012). Individual words exist in both tagged and untagged forms for all ngrams up to a length of three, including dependency relations. Also, for these ngrams, POS tags can stand in place of a word to represent the sum of all the ngram counts for that specific tag. When combined with the tools we describe below, these tags provide a further layer of abstraction in a query.

2.2 Architecture needs better title?

In previous versions of the Ngram Viewer, user requests were directed through the server to a single lookup table containing the raw ngrams. This data flow is maintained for queries containing none of the tools offered by the update. For the new queries the server first requests the correct lookup table or service for each of the new search types, receives a list of possible raw ngram replacements, and then requests those ngrams from the original table. While the number of possible replacements for case insensitive and morphological inflection searches typically remained at a manageable amount, naïve handling of wildcards could in the worst case have returned V^n replacements where V is vocabulary size and n is the length of the ngram. We go into the detail of this issue in Section 3.1.

3 New Tools

3.1 Wildcards

We support the use of wildcards by utilizing an additional database that stores the most frequent replacements of queries to the ngram corpus. This wildcard database is created as a pre-compilation step when creating the Ngram Corpus from the Google Books data. When a new ngram is created, one word or tag at a time is replaced with the wildcard symbol, '*', creating a wildcard query. The query becomes a key in a string indexed lookup table, and the original ngram is added to a list of ngrams which are its values. As mentioned above, due to space considerations we were not able to store all possible replacements for every wildcard ngram we precomputed. Also, we could not precompute a perfect set of the top 10 ngrams for every possible year range, while this would require intermediate storage space on the order discussed above, as well as the computation time to calculate the top 10 for $\frac{n(n+1)}{2}$ time periods for each wildcard. Instead we estimate the top 10 during the creation of wildcard ngrams, by limiting the number of replacements to the top ten for every year. We gather these top 10 lists into a union and map the raw ngrams to the wildcard ngram string in our table.

During this process, we filter punctuation from consideration as a valid replacement, while in practice punctuation returns uninteresting results. On this note, it may be useful to specify a specific POS tag (i.e. '*_NOUN') to provide more specific results. On runtime, union of replacement ngrams is processed for the specific year range requested and the top ten results are returned. For examples of expansions see Table 1.

3.2 Morphological Inflections

Inflections of words in search queries are handled using a Google Search interface that can provide morphological variations of words for different syntactic categories (Durrett and DeNero, 2013). As this type of request utilizes an internal API, the results from a specific request are subject to change. Unlike the wildcard substitutions, there is no need for pre-computation, while the results returned, even for languages such as Russian with vast morphological diversity, keep to a manageable number. While manageable, there can be more than 10 results returned per query, unlike the wildcard search; therefore we have updated the

Wildcard	Possible Replacements
'a * man'	a young man, a good man, a kind man, a wild man
'booked⇒*_NOUN'	booked⇒ flight_NOUN, booked⇒ passage_NOUN, booked⇒ room_NOUN, booked⇒ seat_NOUN
'John said_INF'	John says, John said, John say, John saying
'book_INF_NOUN'	book, books
'the cook' (case insensitive)	THE COOK, the cook, The Cook, the Cook, The cook

Table 1: A table showing examples of the possible precompiled wildcard, inflection, and case insensitive queries.

user interface to better deal with more data lines (see Section 3.4). Due to the time complexity of resulting queries, we do not allow the combination of morphological inflections with wildcards and/or case insensitive searches.

3.3 Case Insensitive

Case insensitive searches are enabled by selecting a check box on the new interface. These queries, like the wildcards, are referenced to a separate database that contains a mapping of different case combinations to the ngram string in lowercase. The possible combinations of case are: ALL CAPS, first letter Capitalization (all possible variations), and all lower case. We utilize a threshold that is a certain percentage what percentage? we should mention this for sure. I think it was 99% but we can check of the top results returned and ignore the results below.

3.4 User Interface

Due to the increased number of results returned per query, we have also updated the user interface. Interactive functionality was added to the graph that allows you to highlight a line by hovering over it, keep that focus by left clicking, and clear all focused lines by double clicking. For any of the three queries mentioned above, you may also right click on any of the queries returned to combine them into the total line for the wildcard query. Another causal feature added to the interface is static URLs which maintain all of the raw ngrams retrieved from any query. This is to prevent statically linked charts from changing over time, and allowing for backwards compatibility.

4 Use Cases

We present multiple use cases that can be captured using the several features that we have presented in this paper. First, we show some examples of each of these individual features; next, we present some example queries that combine queries that use syntactic annotations and the current additions to exhibit the type of results that the Ngram Viewer can retrieve.

5 Conclusions

We have presented a new version of the Ngram Viewer with some new functionality. With the introduction of these new features, users can perform more powerful searches that show trends which were not possible to extract from earlier versions of the tool.

We can cite examples from the media where this has been mentioned, and also show examples from several blog posts/entries from the internet: http://sciencerefinery.com/2013/10/28/google-ngram-viewer-now-more-powerful-than-ever/ http://www.devingriffiths.com/google-books/google-n-gram-studies/
http://languagelog.ldc.upenn.edu/nll/?p=8472
http://www.textualscholarship.nl/?p=14051

References

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL-HLT*, pages 1185–1195.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, pages 169–174. Association for Computational Linguistics.

S. Petrov, D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.