Enhanced Search with Wildcards and Morphological Inflections in the Google Books Ngram Viewer

Jason Mann[♣][*] **David Zhang**[♢][*] **Lu Yang**[♡][*] **Dipanjan Das**[♠] **Slav Petrov**[♠]
[♣]Columbia University [♢]USC [♡]Cornell University [♠]Google Inc.
Contact: `dipanjand@google.com`, `slav@google.com`

Abstract

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

1 Introduction

The Google Books Ngram project facilitates the analysis of cultural, social and linguistic trends through five centuries of written text in eight languages. The Ngram Corpus (Michel et al., 2011; Lin et al., 2012) consists of words and phrases (i.e., ngrams) and their usage frequency over time.[1] The interactive Ngram Viewer[2] allows users to retrieve and plot the frequency of multiple ngrams on a simple webpage. The Viewer is widely popular and can be used to efficiently explore and visualize patterns in the underlying ngram data. For example, the ngram data has been used to detect emotion trends in 20th century books (Acerbi et al., 2013), to analyze text focusing on market capitalism throughout the past century (Schulz and Robinson, 2013), detect social and cultural impact of historical personalities (Skiena and Ward, 2013), or to analyze the correlation of economic crises with a literary 'misery index' reflected in printed text during crises periods (Bentley et al., 2014).

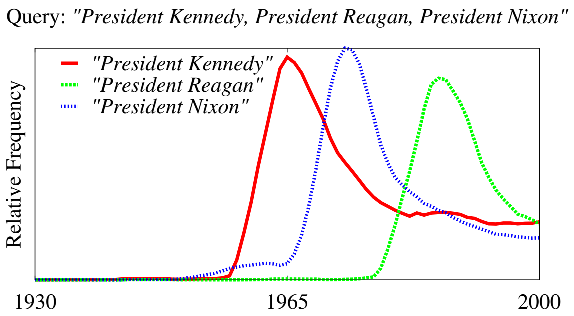

Figure 1: Mention frequencies for three different American presidents queried one-by-one.

A limitation of the Viewer, however, is that all the reasoning has to be done by the user, and only individual, user-specified ngrams can be retrieved and plotted. For example, to compare the popularity of different presidents, one needs to come up with a list of presidents and then search for them one-by-one. The result of the query '`President Kennedy, President Nixon, President Reagan`' is shown in Figure 1. To determine the most popular president, one would need to search for all presidents, which is cumbersome and should ideally be automated.

In this paper, we therefore present an updated version of the Viewer that enhances its search functionality. We introduce three new features that automatically expand a given query and retrieve a collection of ngrams, to facilitate the discovery of patterns in the underlying data. First, users can replace one query term with a placeholder symbol '⋆' (wildcard, henceforth), which will return the ten most frequent expansions of the wildcard in the corpus for the specified year range. Second, by adding a specific marker to any word in a query ('`_INF`'), ngrams with all

[*] The majority of this work was carried out during an internship at Google.

[1] The Ngram Corpus is freely available for download at `http://books.google.com/ngrams/datasets`.

[2] See `http://books.google.com/ngrams`.

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 115–120,
Baltimore, Maryland USA, June 23-24, 2014. ©2014 Association for Computational Linguistics

morphological inflections of that word will be retrieved. Finally, the new Viewer supports capitalization searches, which return all capitalization variants of the query ngram. Figure 2 provides examples for these three new types of queries.

While it is fairly obvious how the above search features can be implemented via brute-force computation, supporting an interactive application with low latency necessitates some precomputation. In particular, the wildcard search feature poses some challenges because the most frequent expansions depend on the selected year range (consider the frequency with which presidents are mentioned during different decades, for example). To this end, we provide details of our system architecture in §2 and discuss how the new search features are implemented in §3. In addition, we present an overhaul of the Ngram Viewer's user interface with interactive features that allow for easier management of the increase in data points returned.

Detailed analysis and interpretation of trends uncovered with the new search interface is beyond the scope of this paper. We highlight some interesting use cases in §4; many of the presented queries were difficult (or impossible) to execute in the previous versions of the system. We emphasize that this demonstration updates only the Viewer, providing tools for easier analysis of the underlying corpora. The ngram corpora themselves are not updated.

2 System Overview

We first briefly review the two editions of the Ngram Corpus (Michel et al., 2011; Lin et al., 2012) and then describe the extensions to the architecture of the Viewer that are needed to support the new search features.

2.1 The Ngram Corpus

The Google Books Ngram Corpus 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-specific analysis. This corpus is a subset of all books digitized at Google and represents more than 6% of all publicized texts (Lin et al., 2012). Two editions of the corpus are available: the first edition dates from 2009 and is described in Michel et al. (2011); the second edition is from 2012 and is described in Lin et al. (2012). The new search features presented here are available for both editions.

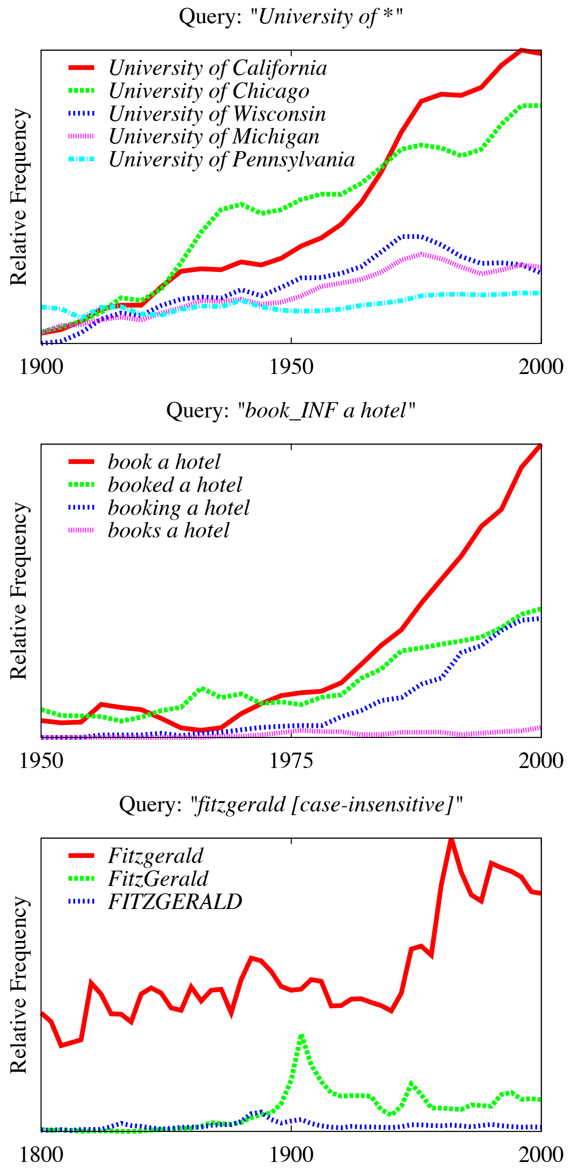

Figure 2: In the new enhanced search features of the Ngram Viewer, a single query is automatically expanded to retrieve multiple related ngrams. From top to bottom, we show examples of the wildcard operator ('*'), the '_INF' marker that results in morphological inflections, and the case insensitive search functionality. Due to space considerations we show only a subset of the results returned by the Ngram Viewer.

Michel et al. (2011) extract ngrams for each page in isolation. More specifically, they use whitespace tokenization and extract all ngrams up to length five. These ngrams include ones that potentially span sentence boundaries, but do not include ngrams that span across page breaks (even when they are part of the same sentence). Lin et al. (2012) on the other hand perform tokenization, text normalization and segmentation into sentences. They then add synthetic _START_ and _END_ tokens to the beginning and end of the sentences to enable the distinction of sentence medial ngrams from those near sentence boundaries. They also ensure that sentences that span across page boundaries are included. Due to these differences, as well as the availability of additional book data, improvements to the optical character recognition algorithms and metadata extraction for dating the books, the ngrams counts from the two editions are not the same.

The edition from Lin et al. (2012) additionally includes syntactic ngrams. The corpus is tagged using the universal part-of-speech (POS) tag set of Petrov et al. (2012): `NOUN` (nouns), `VERB` (verbs), `ADJ` (adjectives), `ADV` (adverbs), `PRON` (pronouns), `DET` (determiners and articles), `ADP` (prepositions and postpositions), `CONJ` (conjunctions). Words can be disambiguated by their POS tag by simply appending the tag to the word with an underscore (e.g. `book_NOUN`) and can also be replaced by POS tags in the ngrams, see Lin et al. (2012) for details. The corpus is parsed with a dependency parser and head-modifier syntactic relations between words in the same sentence are extracted. Dependency relations are represented as '=>' in the corpus. Our new enhanced search features for automatic expansions can also be applied to these syntactic ngrams. In fact, some of the most interesting queries use expansions to automatically uncover related ngrams, while using syntax to focus on particular patterns.

The Viewer supports the composition of ngram frequencies via arithmetic operators. Addition (+), subtraction (-) and division (/) of ngrams are carried out on a per year basis, while multiplication ($\star$) is performed by a scalar that is applied to all counts in the time series. Where ambiguous, the wildcard operator takes precedence over the multiplication operator. Parentheses can be used to disambiguate and to force the interpretation of a mathematical operation.

2.2 Architecture

The Ngram Viewer provides a lightweight interface to the underlying ngram corpora. In its basic form, user requests are directed through the server to a simple lookup table containing the raw ngrams and their frequencies. This data flow is displayed in the top part of Figure 3 and is maintained for queries that do not involve the new expansion features introduced in this work.

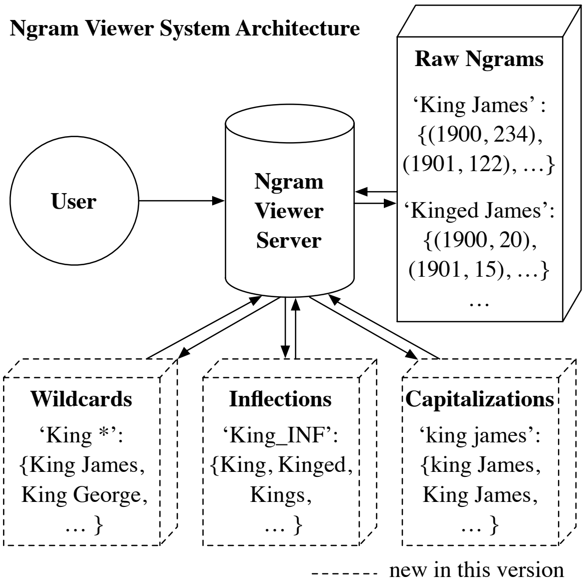
Ngram Viewer System Architecture
User
Ngram Viewer Server
Raw Ngrams
'King James' : {(1900, 234), (1901, 122), ...}
'Kinged James': {(1900, 20), (1901, 15), ...}
...
Wildcards
'King *': {King James, King George, ... }
Inflections
'King_INF': {King, Kinged, Kings, ... }
Capitalizations
'king james': {king James, King James, ... }
------- new in this version

Figure 3: Overview of the Ngram Viewer architecture.

The expansion queries could in principle be implemented by scanning the raw ngrams on the fly and returning the matching subset: to answer the query '`President *`', one would need to obtain all bigrams starting with the word `President` (there are 23,693) and extract the most frequent ten. Given the large number of ngrams (especially for larger n), such an approach turns out to be too slow for an interactive application. We therefore pre-compute intermediate results that can be used to more efficiently retrieve the results for expansion queries. The intermediate results are stored in additional lookup tables (shown at the bottom in Figure 3). When the user executes an expansion search, the query is first routed to the appropriate lookup table which stores all possible expansions (including expansions that might not appear in the corpus). These expanded ngrams are then retrieved from the raw ngram table, sorted by frequency and returned to he user. We describe the intermediate results tables and how they are generated in the next section.

Note that we only support one expansion operation per query ngram. This is needed in order to avoid the combinatorial explosion that would result from mixing multiple expansion operators in the same query.

3 New Features

The three new search features are implemented via the same two-step approach. As shown in Figure 3, we add three new lookup tables that store intermediate results needed for efficiently support-

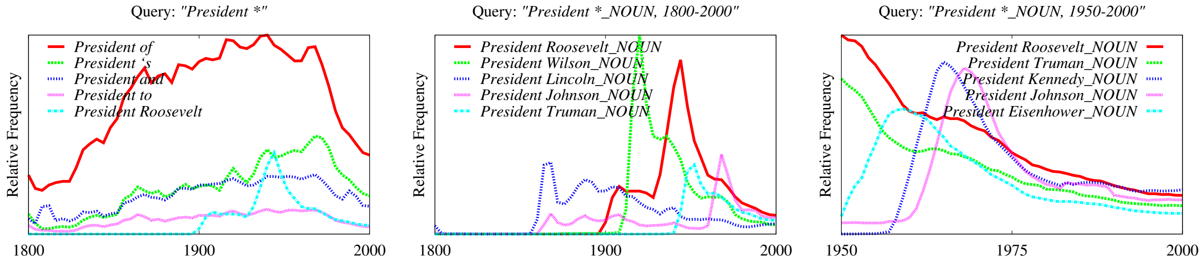

Figure 4: Different wildcard queries for bigrams starting with `President`. Specification of a POS tag along with the wildcard operator results in more specific results, and the results vary depending on the selected yaer range.

ing the new search types. In all cases the lookup tables provide a set of possible expansions that are then retrieved in the original raw ngram table. Below we describe how these intermediate results are generated and how they are used to retrieve the final results.

3.1 Wildcards

Wildcards provide a convenient way to automatically retrieve and explore related ngrams. Because of the large number of possibilities that can fill a wildcard slot, returning anything but the top few expansions is likely to be overwhelming. We therefore return only the ten most frequent expansions. Determining the most frequent expansions is unfortunately computationally very expensive because of the large number of ngrams; the query ‘`the ⋆`’ for example has 2,353,960 expansions.

To avoid expensive on-the-fly computations, we precompute the most frequent expansions for all possible queries. The problem that arises is that the ten most frequent expansions depend on the selected year range. Consider the query ‘`President ⋆`’; we would like to be able get the correct result for any year range. Since our data spans more than 500 years, precomputing the results for all year ranges is not a possibility. Instead, we compute the possible wildcard expansions for each year. The top expansions for the entire range are then taken from the union of top expansions for each year. This set is at most of size $10n$ (where n is the year range) and in practice typically a lot smaller. Theoretically it is possible for this approximation to miss an expansion that is never among the top ten for a particular year, but is cumulatively in the top ten for the entire range. This would happen if there were many spikes in the data, which is not the case.

To make the wildcard expansions more relevant, we filter expansions that consist entirely of punctuation symbols. To further narrow down the expansions and focus on particular patterns, we allow wildcards to be qualified via POS tags. Figure 4 shows some example wildcard queries involving bigrams that start with the word ‘`President`.’ See also Table 1 for some additional examples. Note that it is possible to replace POS tags with wildcards (e.g., `cook_⋆`) which will find all POS tags that the query word can take.

3.2 Morphological Inflections

When comparing ngram frequencies (especially across languages, but also for the same language), it can be useful to examine and potentially aggregate the frequencies of all inflected forms. This can be accomplished by manually deriving all inflected forms and then using arithmetic operations to aggregate their counts. Our new inflected form search accomplishes this automatically. By appending the keyword `_INF` to a word, a set of ngrams with all inflected forms of the word will be retrieved. To generate the inflected forms we make use of Wiktionary[3] and supplement it with automatically generated inflection tables based on the approach of Durrett and DeNero (2013).

Because there are at most a few dozen inflected forms for any given word, we can afford to substitute and retrieve all inflections of the marked word, even the ones that are not grammatical in a given ngram context. This has the advantage that we only need to store inflected forms for individual words rather than entire ngrams. If a generated ngram has no support in the corpus, we simply omit it from the final set of results. We do not perform any additional filtering; as a result, an inflection search can produce many results, especially for morphologically rich languages like Russian. We have therefore updated the user interface to better deal with many data lines (§4).

[3] See `http://www.wiktionary.org/`. Because Wiktionary is an evolving resource, results for a particular query may change over time.

Query	Possible Replacements
* 's Theorem	Lagrange 's Theorem, Gauss 's Theorem, Euler 's Theorem, Pascal 's Theorem

War=>*_NOUN	War=>World_NOUN, War=>Civil_NOUN,
	War=>Second_NOUN, War=>Cold_NOUN

лубовь_INF	лубил, лублу, лубит, лубить, лубила, лубимый, лубишь
book_INF	book, books, booked, booking
book_INF_NOUN	book, books
cook_*	cook_NOUN, cook_VERB
the cook (case insensitive)	THE COOK, the cook, The Cook, the Cook, The cook

Table 1: Examples expansions for wildcard, inflection, and capitalization queries.

3.3 Capitalization

By aggregating different capitalizations of the same word, one can normalize between sentence-initial and sentence-medial occurrences of a given word. A simple way to accomplish this is by searching for a lowercased, capitalized and all caps spelling of the query. This however can miss CamelCase spelling and other capitalization variants (consider FitzGerald for example). It is of course not feasible to try all case variants of every letter in the query. Instead, we perform an offline precomputation step in which we collect all ngrams that map to the same lowercased string. Due to scanning errors and spelling mistakes there can be many extremely rare capitalization variants for a given query. We therefore filter out all variants that have a cumulative count of less than 1% of the most frequent variant for a given year range. Capitalization searches are enabled by selecting a case-insensitive check box on the new interface.

4 Use Cases

The three features introduced in this paper represent a major extension of the capabilities of the Ngram Viewer. While the second edition of the Ngram Corpus (Lin et al., 2012) introduced syntactic ngrams, the functionality of the Viewer had remained largely unchanged since its first launch five years ago. Together, the updated Corpus and Viewer enable a much more detailed analysis of the underlying data. Below we provide some uses cases highlighting the ways in which sophisticated queries can be crafted. While the results produce some intriguing patterns, we leave their analysis to the experts.

Since we have made no modifications to the underlying raw ngrams, all of the plots in this paper could have also been generated with the previous version of the Viewer. They would, however, have required the user to manually generate and issue all query terms. For example, Figure 1 shows manually created queries searching for specific presidents; contrarily, Figure 4 shows single wildcard queries that automatically retrieve the ten most frequently mentioned presidents and uncover additional trends that would have required extra work on behalf of the user.

The wildcard feature used on its own can be a powerful tool for the analysis of top expansions for a certain context. Although already useful on its own, it becomes really powerful when combined with POS tags. The user can attach an underscore and POS tag to either a wildcard-based or inflection-based query to specify that the expansions returned should be of a specific part of speech. Compare the utility of the generic wildcard and a search with a noun part-of-speech specification in a query examining president names, 'President *' vs. 'President *_NOUN' shown in Figure 4. The former gives a mixture of prepositions, particles, and verbs along with names of presidents, and because the latter specifies the noun tag, the top expansions turn out to be names and more in line with the intention of the search. Also, note in Figure 4 the difference in expansions that searching over two different time ranges provides. In Table 2, we compare the combination of the wildcard feature with the existing dependency link feature to highlight a comparison of context across several languages.

It is worth noting that the newly introduced features could result in many lines in the resulting chart. Hence, we have updated the Viewer's user interface to better handle charts involving many ngrams. The new interactive functionality allows the user to highlight a line by hovering over it, keep that focus by left clicking, and clear all focused lines by double clicking. A right click on any of the expansions returned by an issued query combines them into the year-wise sum total of all the expansions. We added another feature to the

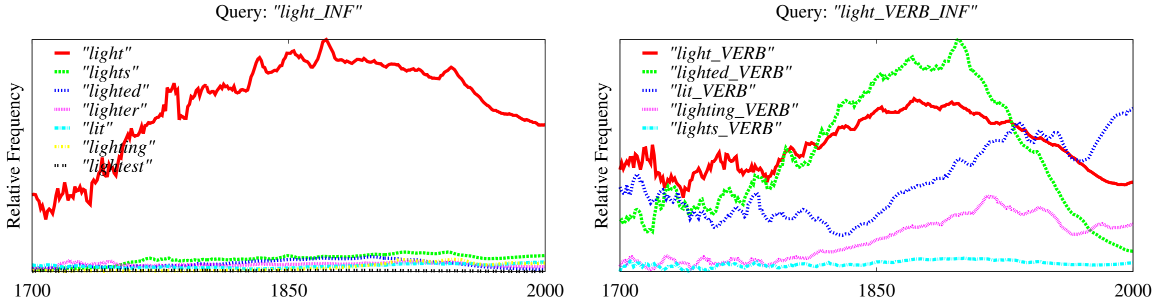

Figure 5: Comparison of specification of POS tag in wildcard search.

English (All)	American English	British English	German	French	Russian	Italian	Chinese (Simplified)	Spanish	Hebrew
drinks	drinks	drinks	trinkt	boit	пьёт	beve	喝	bebe	שתה
water	water	water	Bier (beer)	vin (wine)	он (he)	vino (wine)	酒 (wine)	agua (water)	יין (wine)
wine	wine	wine	Kaffee (coffee)	sang (blood)	чай (tea)	acqua (water)	茶 (tea)	vino (wine)	מים (water)
milk	coffee	tea	Wein (wine)	eau (water)	воду (water)	sangue (blood)	水 (water)	sangre (blood)	ה (the)
coffee	beer	blood	Wasser (water)	cafe (coffee)	Он (He)	birra (beer)	咖啡 (coffee)	vaso (glass)	כוס (cup)
beer	milk	beer	Tee (tea)	verre (glass)	вино (wine)	caffé (coffee)	人 (person)	cerveza (beer)	תה (tea)

Table 2: Comparison of the top modifiers of the verb drinks, or its equivalent in translation, in all corpora, retrieved via the query drinks_VERB=>*_NOUN and equivalents in the other languages. The modifiers can appear both in subject and in object position because we have access only to unlabeled dependencies.

interface that creates static URLs maintaining all the raw ngrams retrieved from any query. This prevents statically linked charts from changing over time, and allowing for backwards compatibility.

One of the primary benefits of the capitalization feature is the combination of multiple searches in one, which allows the user to compare case-insensitive usages of two different phrases. An alternative use is in Figure 2(c), where capitalization search allows the immediate identification of changing orthographic usage of a word or phrase; in this case the figure shows the arrival of F. Scott Fitzgerald in the early to mid 20th century, as well as the rise in popularity of the CamelCase variety of his surname at the turn of the 19th century.

Searches using inflections can be useful for the same reasons as the capitalization feature, and also be used to compare changes in spelling; it is particularly useful for the analysis of irregular verbs, where the query can return both the regular and irregular forms of a verb.

5 Conclusions

We have presented an update to the Ngram Viewer that introduces new search features. Users can now perform more powerful searches that automatically uncover trends which were previously difficult or impossible to extract. We look forward to seeing what users of the Viewer will discover.

6 Acknowledgements

We would like to thank John DeNero, Jon Orwant, Karl Moritz Hermann for many useful discussions.

References

A. Acerbi, V. Lampos, and R. A. Bentley. 2013. Robustness of emotion extraction from 20th century english books. In *Proceedings of the IEEE International Conference on Big Data*.

A. R. Bentley, A. Acerbi, P. Ormerod, and V. Lampos. 2014. Books average previous decade of economic misery. *PLOS One*, 9(1).

G. Durrett and J. DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL-HLT*.

Y. Lin, J.-B. Michel, E. L. Aiden, J. Orwant, W. Brockman, and S. Petrov. 2012. Syntactic annotations for the Google Books Ngram Corpus. In *Proceedings of the ACL*.

J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, The Google Books Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*.

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

J. Schulz and L. Robinson. 2013. Shifting grounds and evolving battlegrounds. *American Journal of Cultural Sociology*, 1(3):373–402.

S. Skiena and C. Ward. 2013. *Who's Bigger?: Where Historical Figures Really Rank*. Cambridge University Press.