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

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 case insensitivity. These additions allow the discovery of patterns that were previously difficult to find in the Google Books Ngram data, 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; it is freely available for download. The interactive Ngram Viewer[1] 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. To cite a few examples, the ngrama 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 personalities throughout history (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).

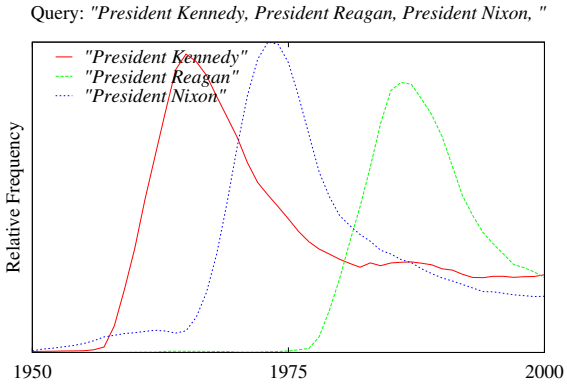

Figure 1: Manual query of president's names

One major limitation of the Ngram Viewer, however, is that all the reasoning needs to be carried out by the user, and that the Viewer cannot automatically discover interesting trends. For example, to compare the popularity of different presidents, one might search for 'President Kennedy, President Nixon, President Reagan' etc. In order to determine the most popular president, one would need to search for all presidents! This is cumbersome and something that we want to do automatically.

In this paper we therefore present an updated version of the Viewer that enhances its search functionality. Traditional searches retrieve and plot the frequency of individual, user-specified ngrams. Instead, the three new features that we introduce expand the user query and automatically retrieve related ngrams in order facilitate the automatic discovery of patterns in the underlying

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

data. First, users can replace one query term with a placeholder symbol '*' (wildcard, henceforth), which will return the ten most frequent replacements of the wildcard in the underlying corpus for the specified year range.

In this paper we therefore present an updated version of the Viewer that enhances its search functionality. Traditional searches retrieve and plot the frequency of individual, user-specified ngrams. Instead, the three new features that we introduce expand the user query and automatically retrieve related nrgams in order facilitate the automatic 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 underlying corpus for the specified year range. Second, by adding a specific marker to any word in a query ('_INF'), ngrams with all morphological inflections of that word will be retrieved. Finally, the new Viewer supports case-insensitive 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 searches can be implemented via brute-force computation, supporting an interactive application with low latency necessitates some non-trivial engineering. In particular the wildcard search 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 on our system architecture in §2 and discuss how the new search features are implemented in §3.

In addition, this demonstration presents an overhaul of the Viewer's user interface (§4), with interactive features that allow for easier management of the increase in data points returned. For example, to normalize for morphological inflections or casing, the frequencies of multiple ngrams can be aggregated via a (right) mouse-click. Additionally, lines can be highlighted or faded (via hovering or mouse-clicks) to focus on particular ngrams and make trends more apparent.

While it is fairly obvious how the above searches can be implemented via brute-force computation, supporting an interactive application with low latency necessitates some non-trivial engineering. In particular the wildcard search poses some challenges because the most frequent replacements 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 on our system architecture in §2 and discuss how the new search features are implemented in §3.

In addition, this demonstration presents an overhaul of the Viewer's user interface (§4), with interactive features that allow for easier management of the increase in data points returned. For example, to normalize for morphological inflections or casing, the frequencies of multiple ngrams can be aggregated via a (right) mouse-click. Additionally, lines can be highlighted or faded (via hovering or mouse-clicks) to focus on particular ngrams and make trends more apparent.

While an analysis and interpretation of trends uncovered with the new search interface is beyond the scope of this paper, we include some interesting use cases in §5. Many of the presented queries were difficult (or even impossible) to execute in the previous versions of the Ngram Viewer. We hope the functionality of the Viewer will help uncover trends and patterns not readily apparent in the data.

2 System Overview

We start by presenting an overview of the system architecture. We briefly review the underlying corpus and architecture of previous versions of the Viewer (Michel et al., 2011; Lin et al., 2012) and then focus on the extensions added in this version. It should be emphasized that this demonstration updates only the Viewer, and provides tools for easier analysis of the underlying data. The ngram data itself is not updated and is identical to that of Lin et al. (2012).

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

[a] These figures show fewer results than returned by the Ngram Viewer due to space considerations.

[2] Downloadable from https://books.google.com/ngrams/datasets.

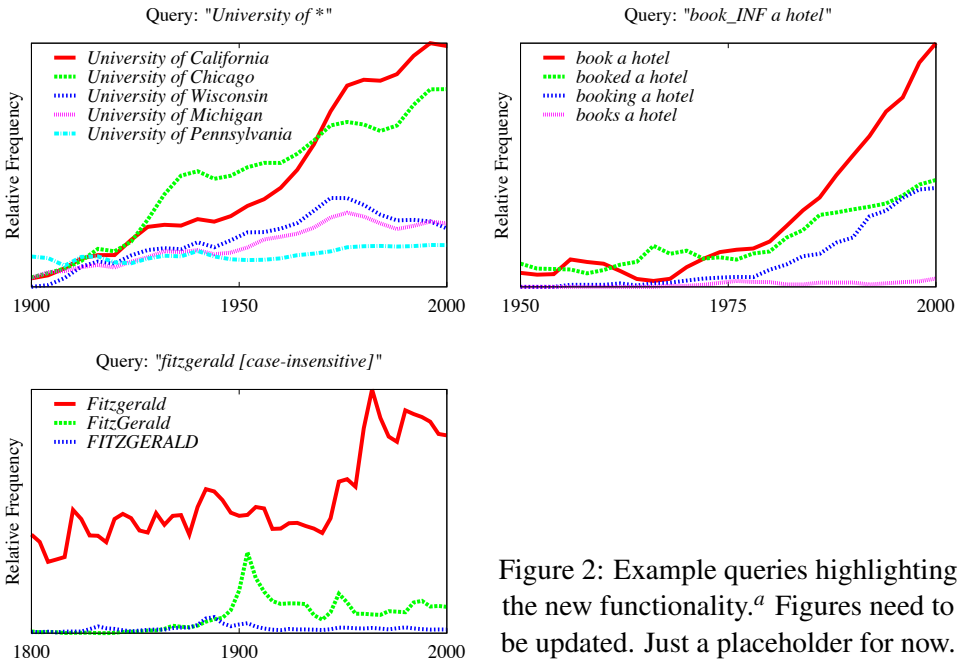

Figure 2: Example queries highlighting the new functionality.[a] Figures need to be updated. Just a placeholder for now.

Google, and represents more than 6% of all publicized texts (Lin et al., 2012). Two editions of 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). Our new features are available for both editions.

Michel et al. (2011) extract ngrams for each page in isolation. More specifically, they use whitespace tokenization and extract all ngrams up to a certain length (5). These ngrams include ngrams 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 based on manually devised rules[3]. 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 (Lin et al., 2012). They also make sure to include sentences that span across page boundaries. 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 additionally 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.

2.2 Architecture

The 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 containing none of the tools offered by the update.

The new types of queries could be in principle

[3]Excluding Chinese, where a statistical segmentation system is used

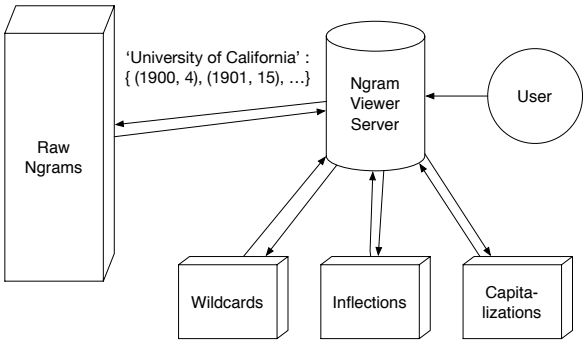

Figure 3: Diagram of new system architecture.

be implemented by scanning the raw ngrams on the fly and returning the relevant subset. Given the large quantity of ngrams, such an approach would be computationally very expensive and too slow for an interactive application. We therefore pre-compute intermediate results that can be used to more efficiently retrieve the results for the new queries. The intermediate results are stored in additional lookup tables (shown at the bottom in Figure 3) that are queried first and then trigger potentially multiple requests to the raw ngram tables. For example, the intermediate results table for the morphological variants search contains inflected forms for all unigrams. These inflected forms are substituted for the selected query term and the resulting ngram is looked up in the raw ngram table. We describe the intermediate results tables and how they are generated in the next section.

It should be noted that we only support one expansion per query ngram. This is needed in order to avoid the combinatorial explosion that would result from mixing multiple expansions in the same ngram.

3 New Features

Here, we describe the novel aspects of the Ngram Viewer introduced in this demonstration paper.

3.1 Wildcards

We support the use of wildcards by utilizing an additional database that stores the most frequent expansions 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 expansions 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 expansions 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 expansion because 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. At runtime, union of expansion 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 wildcard substitutions, there is no need for pre-computation; given a query with the keyword _INF attached to a word (e.g., 'John said_INF'), we get the counts of all the ngrams that start with 'John' and have the morphological inflections of the root form 'say' as its second word in the Ngram Corpus. We have noticed that there can be more than 10 results returned per query, especially for morphologically rich languages like Russian, unlike the wildcard search feature. Therefore we have updated the user interface to better deal with more data lines (§4). We do not allow the combination of morphological inflections with wildcards and/or case insensitive searches because it slows down the retrieval process.

Query	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=>eat_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: Examples of precompiled wildcard, inflection, and case insensitive queries.

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. During experimentation, we noticed that often there are many case variants of a given query for which the ngram counts are negligible; hence, to keep our retrieved results meaningful, we filter out all case variants that have a cumulative count of less than 1% of the most frequent case variant for a given year range.

4 User Interface

Due to the increased number of results returned per query, we have also updated the user interface. 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. 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 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.

5 Use Cases

The previous version of the Ngram Viewer brought the power of syntactic annotations and dependencies to the Ngram Corpus (Lin et al., 2012), allowing for greater specificity when searching the corpus. The features we have added above continue this trend of specialization, but also provide the ability to group similar searches, preventing the need for overly verbose queries. Below we examine example uses of the new features independently and combined with those already present. We show specific queries here to highlight specific usages that we feel users will find the most beneficial, but wish to leave the analysis of the results to the experts.

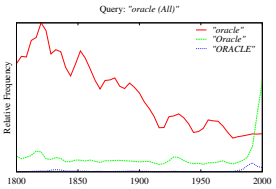

Figure 5: Discovery of Capitalization usage

The wildcard feature used on its own can be a powerful tool for the analysis of top expansions for a certain context. While powerful, we have found greater functionality from wildcards by their specification 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 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 *'/'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 nouns 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.

[J M Most popular words for context across year ranges or languages]

One of the primary benefits of case insensitive searching is the combination of multiple searches in one. As in Figure 5, using case-insensitive searches allows for immediate identification of changing orthographic usage of a word or phrase,

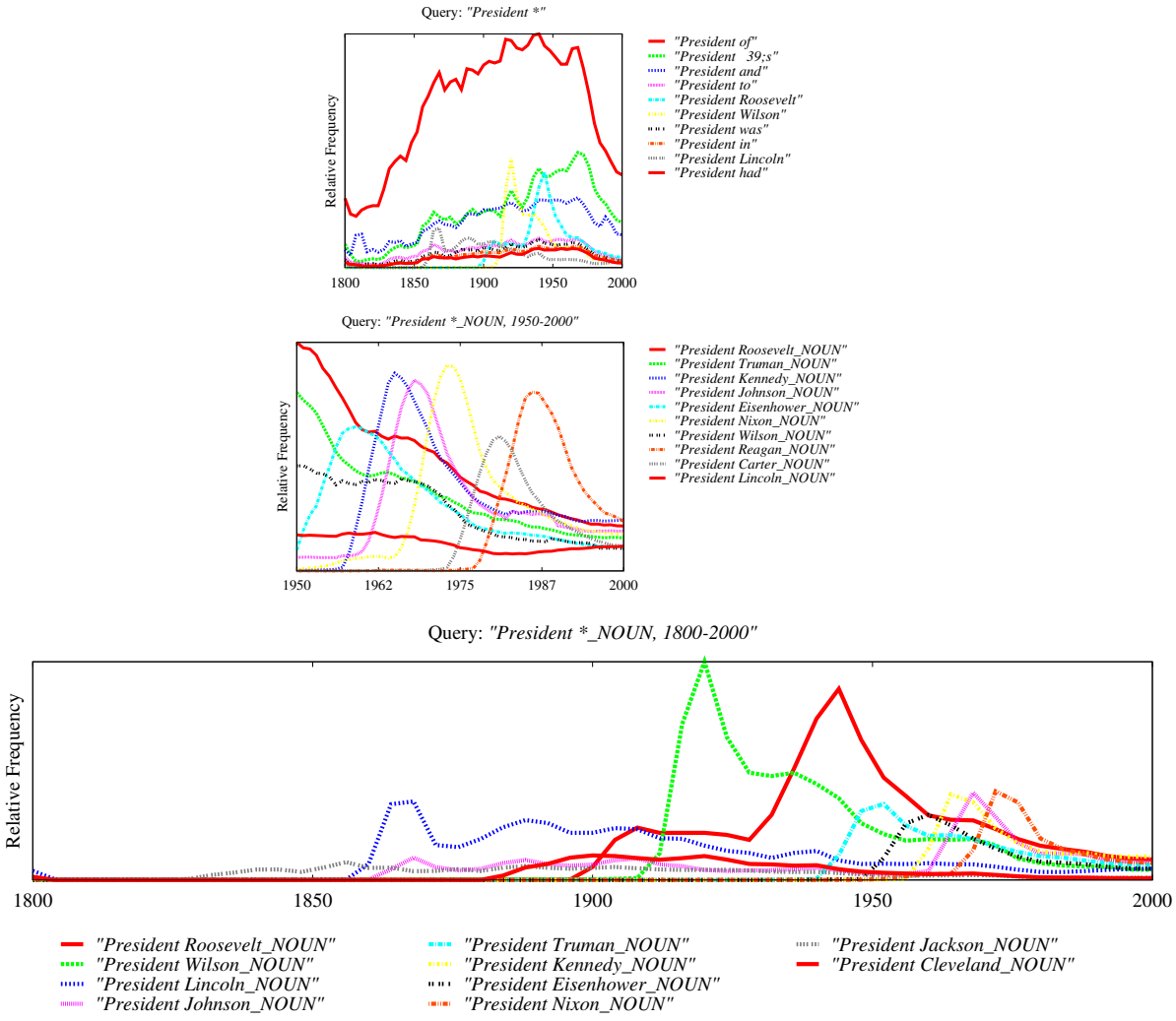

Figure 4: Comparison of specification of POS tag in wildcard search

which in this case shows the arrival

[J M Most popular words for context across year ranges or languages]

[J M Automatic viewing of irregular vs regular verbs]

6 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. The new features have been highlighted in a recent article by the Atlantic, as well on various blogs.

[J M 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

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*, pages 1185–1195.

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 2012 System Demonstrations*, pages 169–174. Association for Computational Linguistics.

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*.

[a]Queries were initiated by translation from 'drink' using Google Translate [J M is there a reference we should cite here for google translate?]

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.

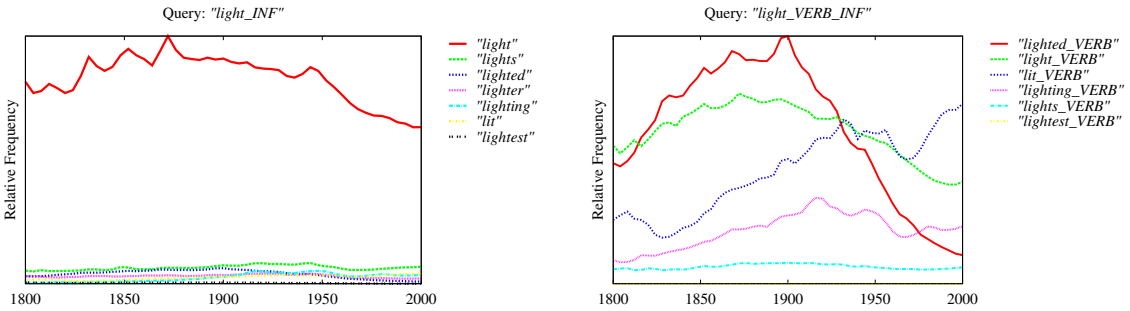

Figure 6: Comparison of specification of POS tag in wildcard search

English (All)	American English	British English	German	French	Russian	Italian	Chinese (Simplified)	Spanish	Hebrew
water	wine	wine	Wein	vin	чай	vino	酒	agua	ויי
wine	water	water	Glas	verre	воду	acqua	茶	vino	מים
blood	cup	health	Wasser	au	вино	icchiere	水	á	סוכ
tea	blood	tea	Kaffee	sang	его	sangue	咖啡	sangre	ה
cup	tea	blood	Bier	lait	водку	caffé	人	aguas	הת

Table 2: Comparison of top dependencies from ‘drink’ in all corpora[a]