A tool for extracting and indexing spatio-temporal information from biographical articles in Wikipedia

Emily G. Morton-Owens

April 26, 2012

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science
Department of Computer Science
New York University

Prof. Ernest Davis

Prof. Ralph Grishman

Acknowledgements

I would like to thank my research advisor, Ernest Davis, for his guidance as I worked on this project and its predecessor, which was created in his Search Engines class.

Thanks also to Ralph Grishman, my second reader, and the NYU Computer Science department for offering a path to studying computer science for students with different undergraduate majors. I am grateful to my department head at the NYU Health Sciences Libraries, Stuart Spore, for his patience during my years of part-time study. My friend Siri Buurma deserves credit for the project's nickname. Joseph Davis also provided useful comments.

As always, I'm most grateful to my parents for their encouragement and support.

Abstract

The Kivrin program, consisting of a crawler, a data collection, and a front-end interface, attempts to extract biographical information from Wikipedia, specifically, spatio-temporal information—who was where when—and make it easily searchable. Some of the considerations standard to moving object databases do not apply in this context, because the texts by their nature discuss a discontinuous series of notable moments. The paper discusses several different methods of arranging the crawler queue priority to find more important figures and of disambiguating locations when the same place name (toponym) is shared among several places. When lifespan information is not available, it is estimated to exclude sightings outside the person's plausible lifetime.

The results are grouped by the number of sightings in the user's search range to minimize the visibility of false drops when they occur. Erroneous results are more visible in times and places where fewer legitimate sightings are recorded; the data is skewed, like Wikipedia itself, towards the U.S. and Western Europe and relatively recent history. The system could be most improved by using statistical methods to predict which terms are more likely personal names than place names and to identify verbs that precede location information rather than personal names. It could also be improved by incorporating the times as a third dimension in the geospatial index, which would allow "near" queries to include that dimension rather than a strict range.

The program can be used at `http://linserv1.cims.nyu.edu:48866/cgi-bin/index.cgi`.

Contents

1 Introduction

1.1 Goal

The goal of this project is to create an index of articles about people in Wikipedia (biographical pages) that has a geospatial facet and a temporal facet. The immediate product is a tool that allows the user to explore different times and places in history through the people associated with them. This unusual form of discovery allows the user to find out about a time and place without prior knowledge and can lead to unexpected connections.

A potential future use of this work would be to create a similar index of, say, a digitized document archive. In that case, instead of indexing which people are relevant to a time and place, the tool would index which documents may be relevant.

1.2 Other answer sources

It is not clear how the question of "who was where when" could be easily and generally answered without this tool. Many specialized reference works exist that cover only parts of this question. "Women writers in the United States: a timeline of literary, cultural, and social history" is an example of a topical reference book organized as a timeline[1]. Many history references are organized around a single geographic place or single time period.

An informal survey of librarians revealed them to be somewhat stumped about a generally applicable method for approaching "who was where when" questions. Such questions are easier to answer when they relate to a time and place that is in the spotlight of history, such as France in 1789. The searcher could read a book on the topic and take note of who is discussed, or just look at the index. It is much harder for less-scrutinized locales, such as Idaho in the 1940s. In that case, librarians suggested consulting local newspapers of the time and possibly the Who's Who books. They also suggested contacting state historical societies. These ideas would probably work, but they require far more effort than this tool does and would not be generalizable outside recent U.S. history.

1.3 Wikipedia as a generalist's resource

The generality and huge scope of Wikipedia (nearly 3.9 million pages in English[2]) makes possible a high-level overview of history and allows the user to explore different combinations of times and places easily. The tool is probably not useful to higher-level users who are already experts in their field; they do not require this type of discovery. It is also not useful for extremely specific queries (like May 1981 in Brooklyn), for reasons that will be discussed.

Any information retrieval tool has to strike a balance between precision and recall. In the design of this tool, I generally leaned towards increasing recall. The tool is meant to reveal people who *may* be of interest; the user still has to look at the articles and determine if they are really interesting. Thus offering more candidates is better than offering fewer. At the same time, an effort was obviously made to minimize false drops that are caused by totally wrong interpretations of the Wikipedia article text, since some of these are so far off that they would undermine the user's trust in the system.

1.4 Components

The project consists of three main components. (For the sake of brevity, I have dubbed it "Kivrin," after the time-traveling historian of Connie Willis's novel *Doomsday Book*[3].)

The first component is a crawler, written in Java, that processes a queue of Wikipedia articles, determines if they are biographical and, if they are, extracts links to follow, biographical metadata, and a series of place-time pairs like "Rome" and "44 BC."

The second component is a database of this information, stored in the document-based (NoSQL) database MongoDB. Instead of tables, MongoDB stores collections of JSON documents.

The third component is a front-end, written in Python with some JavaScript, that allows a user to query the database in several different ways: by place name, by person name, by a pair of geographical coordinates, and by a circle or polygon selector tool (adapted from code written by Marcelo Montagna[4]).

2 Prior work

2.1 Spatio-temporal search in general

Although this project proposes to create a database for times and places, many of the concepts from spatio-temporal search and moving objects databases do not actually apply here[5].

Many moving objects databases deal with objects that have continuously known positions, such as an airplane or cell phone. Others also take into account an object's extent as well as its position (for example, a forest fire). Biographies, however, are written in terms of discontinuous notable moments. The changes to a person's extent are not geographically significant.

Similarly, spatio-temporal databases also sometimes have to do with tracking an object's velocity or trajectory, or predicting future positions. These concepts seem meaningless in the context of a human life—especially the life of someone who is already deceased.

On the other hand, it is useful to consider the concept of an event versus a state. An event is a fact that has no duration, but a state does. If a person in history is observed in a place at a time, is it better to consider that a discrete event or a single observation of an ongoing state? This question is discussed in section 5.2.

2.2 Named entity recognition

In this project, different levels of named entity understanding are required. We assume broadly that the article's sentences are about the subject of the article, so there is no need to identify whether "George W. Bush" and "Bush" refer to the same person through coreference resolution. On the other hand, we do need to identify places and, if the place name is ambiguous, what specific place is being referred to. As is discussed later, place names are identified based on signals like capitalization, context words, and the name's presence in a gazetteer.

The need to differentiate different categories of named entities is similar to the shared tasks for CoNLL 2002 and 2003[6][7]. An earlier paper by Wacholder et al. discusses the

problem in rules-based way, but the signals they use for their tool Nominator (like personal titles) tend not to be present in the style of Wikipedia text[8]. In my initial approach to this project, I avoided the more complex methods used by the CoNLL projects, thinking that the cost would be superfluous queries to gazetteer sources. As discussed in section 12.2, these heuristics leave many personal name identified erroneously as place names.

2.3 Toponym disambiguation

Buscaldi and Rosso provide a valuable comparison of map-based versus knowledge-based toponym disambiguation[9]. In each method, unambiguous toponyms in the surrounding text are used as evidence. In the map-based method, the distance between candidate places and the centroid of the established places is measured. In the knowledge-based method, the established places and the candidate places are located in a tree in "the georeferenced version of WordNet (GeoWordNet)," and a partition of the holonymy hierarchy is calculated. In their experiments, the map-based method performs better when there is sufficient context. In their work, they use the GeoSemCor corpus of general text.

Buscaldi and Rosso discard any context coordinates more than two standard deviations away from the centroid. They also put all of the ambiguous toponyms in the pot together: in their example, using the two Birminghams, there is also "Oxford," which could be in the UK or in Mississippi. They include the coordinates of *both* Oxfords in the context set for Birmingham.

Rauch et al. discuss a confidence-based method for disambiguating potential toponyms [10]. Various heuristics are implemented that go into calculating a final confidence score; both positive and negative hints are considered. Nearby words like "mayor" or "Mr." are taken as evidence for or against the term referring to a place. Populations are taken into account; places with higher populations are more likely to be the correct one (Kivrin uses this idea by setting a minimum population for a place to be considered—see section 4.4). "Correlation of geographic and textual distance" is another of their heuristics. By this, they mean that geographically nearby and containing locations mentioned nearby in the text are strong indicators for disambiguation.

Overell and Rüger discuss the creation of a co-occurrence model to resolve place name ambiguity[11]. As a rationale, they mention (among others) the danger that a user looking for a secondary sense will be "flooded with irrelevant results." They also present three different types of ambiguity: structural (is "North" part of the toponym in "North Carolina"), semantic (does "Lincoln" refer to a president or a town), and referent (is the Lincoln in Nebraska meant or the one in England). They take advantage of the fact that Wikipedia has discrete pages for each sense: "the onus is...on a page author...to correctly link to the intended pages they reference," an assumption also made in the Kivrin project. Overell and Rüger use simple heuristics and basic information accurate enough to create an initial co-occurrence model, which is then applied to a full data dump from Wikipedia. That in turn "produce[s] a co-occurrence model of significant enough size to disambiguate place names in freetext."

In an influential paper, Sehgal et al. attack the opposite problem of entity resolution, or telling when two differing descriptions refer to the same place[12]. Kivrin avoids this question by allowing the user to search a geographic space that may include synonymous

place names.

2.4 Spatio-temporal search in text documents

Manguinhas et al. discuss the development of a multilingual gazetteer service that is used to process RSS feed items in English, Portuguese, and other languages, to extract time and place information[13]. Their paper contains a valuable discussion of possible heuristics for finding the correct interpretation of a geographic name. In particular, they discuss the notion of one referent per discourse ("a geographic reference often means only one of its senses when used multiple times within one discourse context [e.g. the same document]") and the idea of having "a default sense [that] can be assigned to ambiguous references." Both of these heuristics are used in the final version of Kivrin (see section 6.5). The paper further argues that "most document collections can be organized" by time and place and that "relatively simple extraction techniques can still provide results with a sufficiently high quality."

2.5 Information extraction from Wikipedia

A number of prior projects have sought to extract historical time and place information from Wikipedia in various ways.

Sipoš et al. created HistoryViz, which offers a visual timeline about a person[14]. "Using heuristics mostly based on PageRank and cosine similarity of the link structure," they extract relations to other people. The results are a timeline and a browsable network of related people. A picture and capsule biography are also offered, derived from Freebase. This project focuses on the temporal facet as well as on visualization of the data.

In an unpublished paper, Terrell discusses a method of automatically generating timeline pages for Wikipedia[15]. In her work she draws information from Wikipedia, restructures it, and contributes it back to Wikipedia, freeing up human editors for other tasks. She makes several observations about the relatively consistent chronology of articles. She identifies dates down to the granularity of a day, and matches them with compressed sentences indicating why the day is notable. Interestingly, the data is stored in the form of Prolog facts. The facts are used to generate a timeline.

Probably the most similar project to Kivrin is TimeTrails, created by Strötgen et al., which seeks to extract spatio-temporal information from free-text documents[16]. They use Wikipedia featured articles as a test corpus, although they intend the tool for general use on any body of documents. Here the intent is again more along the lines of creating a timeline for an individual person. They discuss the creation of "document trajectories" or "descriptions of event sequences very similar to object trajectories studied in the context of moving object databases." Offsets in time and place are computed. Times are processed using HeidelTime, their own system for extracting and normalizing time expressions[17]. Places are processed using MetaCarta and the data is stored in an SQL database with an R-tree index. The results are visualized on a map. There is also a brief discussion of querying their database by a region, but this is not the primary aim. This work is relatively closely tied to the idea of trajectories in general spatio-temporal databases. It is unclear to me how to interpret this in the context of historical figures; Magellan is unlikely to go anywhere new!

2.6 Disambiguation with Wikipedia

Wikipedia has been a popular source of background knowledge to help disambiguate named entities. Bunescu and Pasca developed have developed a program that uses a support vector machine and Wikipedia redirect and disambiguation pages, categories, and links between articles to determine the most likely assignment of an ambiguous entity to single sense[18].

Cucerzan used similar data signals from Wikipedia to choose an entity assignment that that "maximizes the similarity between the document vector and the entity vector[19]." This system is implemented as a web browser that displays text (from any source) with the canonical form of a surface name displayed as a tooltip.

Nguyen and Cao developed a hybrid system that combines patterns and a vector space model in two steps to match a proper name with a specific entry in Wikipedia[20]. They particularly target Wikipedia disambiguation pages as the source of a "disambiguation dictionary" that is used like a gazetteer. The disambiguation proceeds iteratively using concretely identified named entities to help disambiguate nearby entities.

Dakka and Cucerzan trained a classifier–using a small corpus of manually tagged Wikipedia articles–to classify other Wikipedia pages according to the categories used in the CoNLL 2002-03 shared tasks[21]. In their system, they pay special attention to the first paragraph and abstract as containing the key information. This was more successful than using infoboxes or feature vectors of a text window of three words on either side of the reference.

Han and Zhao explain their procedure as an attempt to move away from naive bag-of-words models and use Wikipedia as a social network for disambiguation[22]. They propose "a novel similarity measure which allows us to take into account the full semantic relations indicated by hyperlinks within Wikipedia, rather than just term overlap or social relatedness between named entities." Context information that is similar in meaning but not identical in terms can be used to group observations into single-entity clusters.

2.7 Place and space

Buckland et al., in a survey of tools to support geographic search, discuss a distinction "between place, a cultural concept, and space, a physical concept[23]." Space can be specified by measurements, but places have human-invented names that may change over time and are more likely to be ambiguous. The choices of searching Kivrin by a geographical shape or a place name offers access to the data by either of these notions.

2.8 Flora & Fauna Finder

This project grew directly from my previous Flora & Fauna Finder project. FFF uses a similar (but much simpler) crawler written in Java, with storage in MongoDB and a front-end written in Python. FFF finds pages that contain the Taxobox template, and processes those pertaining to plants and animals. Looking for phrases like "native to *Nigeria*," it creates a database of where these plants and animals can be found. The results are organized by the taxonomy (genus, family, etc.).

Sample Flora & Fauna Finder results: `http://linserv1.cims.nyu.edu:48866/cgi-bin/classResults.cgi?place=japan`

Kivrin improves on FFF in many ways while taking on the more difficult task of processing human biographies. Where FFF could often extract habitat information from a single sentence, Kivrin finds sightings through the entire article text. FFF does not take account of time at all; dinosaurs could appear alongside plants living today. FFF naively checks whether a candidate phrase is a toponym at all, and does not care where it is geographically or if there is more than one place with that name. It does not store coordinates and can only be searched by place name. FFF accesses the HTML of the Wikipedia articles, which is not only not the preferred method of access, but relatively slow and difficult to parse.

3 Basic design

Kivrin consists of a database of time-place pairs built by a crawler and accessed by a web front end.

The crawler processes pages from Wikipedia, retrieves their metadata to see if they are biographical and get birth and death data, and then examines the free text of the article to look for times and places. The queue of pages to be processed is weighted to promote more important pages being processed earlier.

Times are matched by one of two regular expressions. If the person was born after the year 1000, the regex looks for four-digit numbers with no commas (which are very rarely anything but years). If the person was born earlier, a more complicated regex that accepts lower-digit numbers and AD/BC/BCE is used.

Places are matched by looking for trigger words like "to" or "visited" followed by a capitalized phrase. The candidate place then goes through a series of checks:

1. It is checked against a list of known rejected phrases, like "President."

2. If it is a Wikipedia link, the target page is retrieved. If coordinates can be found there, they are compared against the list of known places so far. If there is already a place with the same name and close coordinates, it is assumed to be the same place. If not, a new place is created based on Wikipedia's information.

3. Not all pages in Wikipedia that are about places have links. If no information was found in Wikipedia, the place is looked up in the prior places found for that person, on the assumption that there is one sense of a place per discourse (article) unless a link clarifies otherwise.

4. Failing that, the place is looked up in the locally created collection (i.e. MongoDB table) of places. If a place with the same name is found there and marked "default," that one is chosen.

5. Finally, the GeoNames gazetteer is tried and the first (most important) result is accepted and marked as the default. If a place with the same name and close coordinates was already present from Wikipedia, that one is marked default.

6. If no information was found in any of these places, the phrase is probably not a valid place; it is ignored and added to the rejects collection to be ignored in the future as well.

If this process results in at least one valid sighting of the person, they are added to the people collection along with their birth and death dates and description. Birth and death years are estimated if not available.

The user can query the front end by drawing a search shape on a map, using a place name, or using a person's name. If there are multiple possible place or person name matches, the user is asked to confirm. The search results are ordered by the number of sightings, with the person who has the most sightings in the desired span appearing first. If there are more than one hundred places, only the top one hundred most frequently seen places are actually processed.

The final version of the data, found at `http://linserv1.cims.nyu.edu:48866/cgi-bin/index.cgi` is created by the final version of the queue design discussed in section 6.1 and the cascadecrawler version of the disambiguator discussed in section 6.5.

4 Data Sources

4.1 What is to be extracted

In this project, the goal is to create pairings of times and places associated with a third entity: the person. This could be viewed as a very simple kind of template filling task in which each entity needs to be represented to create a valid sighting.

There are two kinds of named entities that are of interest: people and places. The identification of people is somewhat trivial, as Kivrin only processes Wikipedia pages that are about a single person and marked as such by a human editor.

Identifying places is a more significant challenge. Some of the signals that are used include shape features (capitalization), predictive words (prepositions and manually selected motion verbs), and gazetteer sources. The gazetteers do not entirely prevent cross-type confusion of personal names being mistaken for toponyms (see sections 10.2 and 12.2).

4.2 Finding people in Wikipedia

A 2008 paper, written when English Wikipedia contained about 2.5 million articles, revealed that about 15% of articles were about "People and self," a category that includes biographies, but is not exclusively composed of them[24]. Using an early version of Kivrin, which did not try to prioritize the queue, it appeared that about 12% of processed pages were biographical.

In this project, I chose to identify biographical pages by only accepting those articles that use the Persondata template[25], similar to my approach of identifying plants and animals in my prior project using the Taxobox template[26]. The Persondata template is actually simpler, including only the fields name, alternative names, short description, and the places and dates of birth and death.

The Persondata template is not usually visible on the page. On the page for Bill Clinton, the displayed infobox is the "Officeholder" infobox. The Persondata *is* present in the article source.

Not all articles about people have the Persondata template; however, its use has been strongly encouraged and *good* biographical articles have it. If an article lacks even this very

basic metadata, it is unlikely to be a high-quality page and can be safely ignored. Over the course of crawling tends of thousands of pages, I discovered only a handful of pages that weren't individual biographies that did have the Persondata template; they were closely related and possibly the work of a single misguided editor.

The short description from the template is used in the search results to give the user a sense of who the person is, even though the quality is uneven. (Thomas Paine is a "pamphleteer" but James Cook merely says "Kingdom of Great Britain.") The birth and death dates are used as the initial pair of sightings and also to limit what other dates are permissible; obviously if someone was born in 1553, any sighting in 1540 should be thrown out.

4.3 Finding places in Wikipedia

Wikipedia also contains many articles with geographic coordinates, whether about places or events, such as Nepal or the Battle of Gettysburg. Kivrin also uses this metadata when possible (see section 6.4). Users can add coordinates to any kind of article they like, even individual streets like the Strand in London.

4.4 References, image captions, and quotes

Wikipedia article text often contains lengthy references in the text.

> [[Luther Bible|His translation of the Bible]] into the [[vernacular]] (instead of [[Latin]]) made it more accessible, causing a tremendous impact on the church and on German culture. It fostered the development of a standard version of the [[German language#Modern German|German language]], added several principles to the art of translation,⟨ref⟩Fahlbusch, Erwin and Bromiley, Geoffrey William. "The Encyclopedia of Christianity". Grand Rapids, MI: Leiden, Netherlands: Wm. B. Eerdmans; Brill, 1999–2003, 1:244.⟨/ref⟩and influenced the translation into English of the [[Authorized King James Version|King James Bible]].⟨ref name="Tyndale" ⟩"Tyndale's New Testament", trans. from the Greek by William Tyndale in 1534 in a modern-spelling edition and with an introduction by David Daniell. New Haven, CT: [[Yale University]] Press, 1989, ix–x.⟨/ref⟩

These must be removed because they contain all sorts of undesirable places and dates, often with un-sentence-like punctuation that is hard to process. Image captions and blockquotes can create similar problems. Images were allowed to remain because more often than not, any times and places mentioned in the caption are relevant; but they may still represent a discontinuity in the article narrative that may cause dates to be applied incorrectly.

4.5 What is in GeoNames and how it can be used

GeoNames is a gazetteer or geographical database that can be used for free (unlike similar references like the Getty Thesaurus of Geographical Names). Its data comes from the US Board on Geographic Names, the National Geospatial-Intelligence Agency, Wikipedia, and user contributions[27].

The information on geographic places it offers includes place names (toponyms), alternate names in various languages, coordinates, what type of place it is, population, and more. Because it contains over ten million records, there is a potential for queries to match even when they do not actually refer to places. For example, "Obama" and "Chair" (often used instead of "chairman") are both words that Kivrin may investigate as potential toponyms. GeoNames will return an exact-match toponym in both cases, even though Obama is far more likely to mean the president than a town of less than 35,000 people in Japan.

To discourage this kind of mistake, Kivrin accepts only a small subset of GeoNames's feature codes[28] that are most likely to appear in historical information, such as countries, regions, oceans, mountains, towns, etc. Examples of excluded codes are "irrigation ditch" and "bus stop."

4.6 Centroids versus bounding boxes

When storing information about geographic places, one can use a bounding box (a minimum rectangle that includes the maximum extent of the entity) or a center of some kind. Wikipedia's templates are intended for a single pair of coordinates, although some articles also offer a bounding box elsewhere on the page. GeoNames offers bounding boxes for countries but not necessarily other entities like cities. Between these two sources, there is a possibly of a mismatch in whether centroids or boxes are used, but there is more centroid data, so centroids were chosen for the project. That said, it seems that in some cases GeoNames contains not a geometrically calculated centroid but rather some general central point. This may be the result of user contributions to GeoNames; centroids sometimes appear to be in the wrong place to human eyes.

Another reason for this choice is that bounding boxes do not work well with strange shapes, which unfortunately many geographic places are. For example, the bounding box for China contains several other countries in their entirety. Using bounding boxes also increases the amount of data that must be stored for each item—two points rather than one.

Figure 1: Bounding box for China[29]

A major downside of centroids is that the centroid of a larger place is near smaller ones, so that all the results for Germany could unintentionally appear in a search around the area of Frankfurt. On the other hand, using bounding boxes as the basis for searching could also be confusing for a user. The user could draw a box around most of Florida but miss part of the panhandle and *not* get results for the state since it was not entirely included.

4.7 Ancient places

The farther back ones goes in history, the more likely one is to encounter political entities that no longer exist or places that have been renamed. GeoNames does not contain coordinates for such places, but Wikipedia does on an inconsistent basis (e.g. Numantia). Kivrin handles these places when coordinates can be found. First, the historic places with the highest number of missed encounters (as counted by the rejects collection in earlier crawls) were preloaded into the system. Examples include Constantinople and the Soviet Union. Beyond these, whether a place gets picked up depends on Wikipedia. If the place links to a Wikipedia page with coordinates, they will be accepted; if not, nothing will be found in GeoNames.

Accepting ancient places is useful for coordinate-based searches; if Kivrin ignored everything about Constantinople, there would be a mysterious dearth of results in that part of the world prior to 1500 because those sightings would have been dropped. Because of the spatial search feature, it is worth creating sightings even for place names like Numantia that are unknown to modern people.

4.8 DBpedia is not used

DBpedia is a project that attempts to extract structured information from Wikipedia (e.g. from templates), store it in RDF (Resource Description Framework), and allow it to be queried using SPARQL (a structured query language for RDF). Initially it seemed that DBpedia would be a good resource for determining whether a Wikipedia article is about a person and for retrieving metadata like birth and death dates.

As it turns out, the Persondata template provides a trivial solution to identifying people. Also, a spot check of birth and death dates and places from DBpedia revealed that whether fields like `dbpedia-owl:birthDate` or `dbprop:dateOfBirth` were favored was inconsistent. The information was not incorrect (with some exceptions), but it was not clear which field should be queried to retrieve the information consistently. The data was drawn from the Persondata template, anyway, which was already available and reasonably easy to process.

DBpedia did not seem to offer any improvement in accuracy or ease of use, and querying another service would have added complication and possibly slowness, so in the end, I did not use DBpedia.

5 Patterns and assumptions about dates and places

In creating this project, I observed patterns about the nature of history, biography, and Wikipedia that informed how Kivrin processes the text. For example, most biographical entries do not jump around in time from 1943 to 1977, then 1959. Further, it is usually

spelled out when a person makes a major move from one place to another. If this were not true, it would be very disorienting to read text about someone in Sydney who then pops up in Bombay without any further explanation.

5.1 Dates are carried forward

As the crawler processes sentences, it picks up dates and holds onto them, trying to match them with places in the current and subsequent sentences. The reverse is not true; Kivrin does not pick up a place and then try to match it against times that appear later. Given this hypothetical text:

> In 1943, he left school. He joined his family in Bombay for several years, leaving in 1945.

Kivrin would retrieve Bombay and 1945, as they appear in the same sentence. If it were written slightly differently, as:

> In 1943, he left school. He joined his family in Bombay for several years, working at the family firm. In 1945, he left for England.

Kivrin would pick up 1943 and Bombay, and then 1945 and England. If the text said:

> He went to Bombay for business. He arrived there in 1943.

No match would be found unless there were a usable date mentioned before the first sentence. This form of discourse feels unnatural, though—almost backwards—and was not encountered often in actual Wikipedia content.

Initially, the crawler was designed to drop a carried-forward date anytime it encountered a section break. A section break might indicate a new discourse that was disjoint from what came before. In practice, though, this caused more useful dates to be dropped than it prevented false hits. It turns out that biographical entries proceed in a mostly chronological way, with the exception of a legacy discussion sometimes appearing at the end of the article.

5.2 Places are not carried forward

Given that the crawler assumes a date is carried forward until it is contradicted, does it equally make sense to assume a place is carried forward until it is contradicted? That is to say, if Kivrin logs Ulysses S. Grant as being in Oregon in 1852, and he isn't mentioned being in a specific different place until (hypothetically) 1856, should it create entries for Oregon in 1853, 1854, and 1855? This is the question of whether to consider an observation of a person in a place as a discrete event or an observation of an ongoing state.

In practice, this seemed like a bad idea for two reasons. First, if there aren't very many places mentioned in an article, this method could generate dozens of "invented" facts implying a continuous state alongside just a few concretely observed events. The volume of data in the system would tip towards dubiously extracted, unconfirmed facts. Second, the front-end is meant to process a time span. If the user wants 1997, 1995 to 1999 would be a good search strategy. The further back in history you go, the longer time span is desirable. If, in the above hypothetical example, the user searched for 1850 to 1860, they would pick

up Ulysses S. Grant even if there is only the 1852 sighting alone. In short, historical text may have a sparse amount of definite sightings and it is not possible for the system to know whether the person stayed in the same place or went to other places that were simply not worth mentioning.

5.3 Notability and specific places

Places need to meet a certain standard of notability to be included, which is partially to reduce false matches but also because it is unlikely that a searcher is really interested in a ditch. The place also needs to be specific; none of the references used is likely to provide coordinates for informal places like "The Great Plains." Finally, continents are excluded because they are too big to be useful. Accidentally picking up the centroid of Europe in a search would result in far too many results, and not the ones the user desired.

5.4 Subject of the discourse

When processing sentences, the crawler assumes they are all about the subject of the article. Sometimes, of course, this is not true. There are cases where it still makes historical sense to consider it a match. Hitler may not have personally gone to Ukraine in 1941, but if you are interested in Ukraine in the early 1940s, Hitler is a major figure who would seem to belong in the search results.

There are harder cases. In Vivien Leigh, some sentences seems to belong in the article about Laurence Olivier:

> Offered the role of Heathcliff in Samuel Goldwyn's production of Wuthering Heights (1939), he travelled to Hollywood, leaving Leigh in London.

Since the crawler is processing information about Leigh, it should only pick up London, but it will also pick up Hollywood. Kivrin does not try to untangle the antecedents of sentences like this.

6 Processing and interpretation

6.1 Processing the queue

The crawler starts with about twenty historical figures, selected to offer diverse starting points across time and space. The figures are meant to be relatively important in history but were also chosen to have links to other Wikipedia articles about important historical figures. For example, Benjamin Franklin gives the crawler entrée to a large number of articles about leaders of Colonial America.

As the crawler processes a page, it also gets the titles of articles that page links to; these are returned in alphabetical order. The API allows requests of up to 500 links, but in the case of long articles this may not be complete. For example, in Bill Clinton, the first 500 alphabetized links only includes those up through "J."

I tried several different ways to prioritize the queue to achieve the best coverage of "important" historical figures without resorting to providing a lengthy, hard-coded list. By important, I mean political and cultural figures whose significance has become the object

of consensus over time as opposed to, say, pop-culture figures of the present moment. At the same time, I sought to ensure that figures from Africa, East Asia, and South America had a chance of being picked up.

The first version of the crawler simply queued and processed articles first-in-first-out. Article titles were upserted into the queue; if the article title was already present, nothing changed. This was a transparent approach but a "dumb" one—in effect, it worked out from the seed articles in slow concentric circles, queuing people who personally knew or were closely associated with people who had already been processed.

The next version attempted to prioritize the queue. Each time a link was encountered, instead of simply upserting it, the number of times it had been seen was incremented. The next article to be processed would be the one with the highest score. This method attempted to allow more important articles (measured by number of encountered inlinks) to be processed earlier, but it retained a bias towards the initial seed set and its successors. Unfortunately, this type of queue showed a tendency to get sucked into local whirlpools of unimportant articles that all link to one another. For example, the page on football player Frank Bausch links to every other page about a player on the 1940 Chicago Bears, giving each of them one inlink. When the next one of those pages is processed, each of the remaining players gets another inlink. Soon, the entire team bubbles up to the top of the list and gets processed, further incrementing the inlink scores of additional football players. Because the queue is, at this point, only processing articles about professional sports players, an article about an American president does not have a chance to get incremented and rise to the top to be processed—at least, not until every football player in history has been.

The obvious solution to this problem is to process articles in order of how many *absolute* inlinks exist to it across all of Wikipedia. A tool exists that gives this information, but it has its own flaws[30]. The first issue is that querying the tool significantly slows the crawl; it can take roughly a second to get the results in the case of a page that many others link to, since the tool is performing an SQL query with multiple joins across huge tables in a copy of the Wikipedia database. Where an article contains many links, more time is spent queuing them than is spent processing the article itself. The second issue is that the tool is sometimes unresponsive or has database errors, and it is not clear what to do with queueable articles when this happens. The third issue is that many of the most linked-to articles within Wikipedia *are* in fact about figures like current professional wrestlers.

The final version uses a queue that only uses information that has been gathered during the current crawl but is weighted to avoid the "whirlpool problem." It is based on the insight that more complete Wikipedia articles are higher quality and more significant, and the pages they link to are also likely to be more significant. The quality of an article is approximated by counting how many sightings it generated. All of the pages it links to get incremented by its number of sightings. This way, an article like Frank Bausch, which is only two sentences long, cannot add very much weight to the pages it links to, but Bill Clinton, which yields dozens of sightings, can lend a lot of weight to an article like Barack Obama. Barack Obama's score is further increased when another high-quality article linking to it, like Al Gore, gets processed.

To introduce an element of chance, every third processed page is chosen simply by popping the oldest one from the queue rather than the one with the highest score. (It may be better to choose at random, but MongoDB does not offer a simple way to do that.)

Both the method of using the absolute inlink count and the method of weighting pages by the quality of their referrers are similar in flavor to PageRank[31]. The final approach, while not perfect, seem to balance the pros and cons the best. Interestingly, it serves to focus the crawl on biographical pages better. The first version, taking pages purely in the order encountered, found about 12% of retrieved articles to be biographical. In the final version nearly half of the dequeues are biographical (see section 10.1).

6.2 Time granularity and assumptions

Because Wikipedia is chiefly meant to be read by humans (and was composed by humans), it does not express dates in a machine-readable way.

The majority of dates in Wikipedia text are expressed in absolute terms, especially if we limit our search to years, but the granularity of time expressions is inconsistent. Times closer to the present day may be expressed more precisely, due to better record-keeping as well as familiarity. (The average person remembers the exact date of 9/11, but the date of the Battle of Thermopylae in 480 BC is known only approximately, which does not bother the average person.)

It turns out not to be practical to index even recent dates very granularly because human-written text is not consistent about the specificity of date expressions. If two people were in New York City on 9/11/01, one person's article may mention the exact date and the other person's may only say 2001. The first person should be returned for a search on 2001 as a whole, but should the second person be returned for a search on 9/11/01? If we restrict results to times the crawler actually observed (accordingly to the logic of not creating "invented" sightings of Ulysses S. Grant in Oregon, above), then the results for 9/11/01 would be sparse, and for unimportant dates, *extremely* sparse—even if a number of indexed people were actually there.

BC dates are transformed into negative numbers for processing and storage, but are converted back to display as BC for usability.

Some dates are affected by the conversion from the Julian to the Gregorian calendar, which involved a drift of about two weeks in the solar calendar as well as a change in whether January 1 or March 25 was considered the start of the year. Because the changeover happened in different countries at different times, it would be difficult to correctly normalize the dates from freely written text. Kivrin does not compensate for Old Style versus New Style dates, since both versions would usually fall in the same multiyear search range anyway[32].

Overall, it is more in keeping with the use case of the tool to only let users search at the granularity of a year. They can confirm their results by reading the article or through further research beyond Wikipedia.

6.3 Estimating birth and death dates

Birth and death dates are important for limiting at what times sightings are accepted; without them, extreme outliers can crop up, including numbers in the single digits (mismatches) and very recent dates pertaining to the subject's legacy—even if the subject lived hundreds of years ago.

If there is a birth date but no death date, the death date is assumed to be 100 years later—a generous span. The estimated death date is used to limit the sightings but is not

entered as metadata about the person. Conversely, if there is a death date but no birth date, the birth date is assumed to 100 years earlier.

If both are missing, it is necessary to provide an estimate. I experimented with an average and a standard deviation limit. Both of these failed because the outlying false dates—exactly the ones that were most likely to be wrong—had too much influence. In the end, Kivrin simply finds the *mode* of all the sightings' times, and uses a range of 50 years on either side of it. This way, if the most sightings are from 1887, 5 and 2008 are both easily rejected. Again, a hundred-year span is meant to be generous and err on the side of accepting information.

6.4 Order of querying

GeoNames limits users to 2,000 queries per hour and 30,000 per day. It is desirable to prevent unneeded queries to avoid hitting that limit. First, strings that appear in place-like contexts but are not places can be maintained in a rejects collection, so GeoNames is never queried for the same string more than once. The rejects collection is pre-loaded with terms that often cause problems ("in *World War II*") and others are added as they are encountered. There is also a count of how many times a rejected term has been encountered, for analysis.

The most frequently encountered confounding terms are months ("June"), geographic (or perhaps linguistic) adjectives ("English"), and places that fail to be matched. Some are too big ("Europe"). Others are historical ("Sparta"), which is not handled by GeoNames (although some of these may have coordinates in Wikipedia, see section 4.6). Finally, some of the most frequently encountered rejects are nongeographical entities like "Parliament," "Christianity," and "Manchester United." As the crawl begins, the rejects list grows quickly, then levels off when the bulk of confounding terms have already been identified. Similarly, the number of GeoNames queries required drops as the crawl progresses, because more and more places are available in Kivrin's own places collection or the rejects collection.

Another way to avoid excessive querying of GeoNames is to get geographic coordinates directly from Wikipedia first. This approach has advantages and disadvantages. On the positive side, if a place name is given as a link to a Wikipedia article, we know that the human editor is telling us unambiguously what place is meant. If it is possible to extract coordinates from that page, they are likely to be highly accurate. On the negative side, coordinates are expressed inconsistently in Wikipedia (see section 11.3) which makes them convoluted to match. This is the slowest component of the crawler, which in the middle of a run processes about two hundred biographies per hour.

GeoNames offers a way to search data drawn from Wikipedia associating coordinates with articles, which may be faster than retrieving the entire article. Unfortunately, it does not allow you to retrieve a single article based on an exact title match (which makes it no more accurate or efficient than a regular GeoNames search, in addition to "costing" a query). It also throws errors when the place name being searched contains a comma, which is a problem because many Wikipedia articles that are linked to have titles like "Vincennes, Indiana." Not being able to distinguish "Vincennes, Indiana" from "Vincennes, France" (if GeoNames only deals with the part before the comma), makes this feature not useful for this project.

6.5 Multimatch - different places sharing a toponym

A number of places share names with other places, such as London, England and London, Ontario. The default behavior of GeoNames is to return matches in order of relevancy, giving weight to capitals and higher populations. Accepting the first match means that London, Ontario may never be associated with any data—not even a mayor of London, Ontario—unless coordinates are picked up through a Wikipedia link.

The "multimatch" feature in Kivrin was tested by introducing pages that were known to discuss different places with a shared toponym. Fred Shuttlesworth and Martin Luther King, Jr. represented Birmingham, AL and James Watt and Joseph Priestley represented Birmingham, England. Most of the toponym conflicts I encountered involved England and her former colonies, so this is a realistic test.

The project went through four different versions of the order for matching place names. The very first version stripped out all Wikipedia markup and only referred to GeoNames. Like the Flora & Fauna Finder project, it simply accepted the first result from GeoNames. It was similar to always hitting the "I'm Feeling Lucky" button in Google. This was only for testing GeoNames and was never intended as a permanent structure.

The second version (tidycrawler) used coordinates from linked Wikipedia articles if available. If not, it used the first GeoNames match. There was a unique index on the place name field. In theory, this meant that if London, Ontario was encountered from Wikipedia before GeoNames was queried for its first result, London, England, all future sightings of London would be placed in Canada. In practice, the most prominent match almost always made it into the database first. The results of this version were mixed: on one hand, the results felt natural and reasonably accurate since the user most often *is* searching for the most prominent match. On the other hand, it was unsatisfying to know that the crawler was definitely going to make foreseeable mistakes.

The third version (multicrawler) attempted to permit an unlimited number of places to share the same name, and then figure out which match was most likely. As in tidycrawler, multicrawler used coordinates from linked Wikipedia articles. If there was no link, it tried to use a version of the centroid calculations discussed by Buscaldi and Rosso[9]. However, whereas they threw all the possibly conflicting toponyms in the same bucket and disambiguated them against all the confirmed place coordinates en masse (see section 2.2), multicrawler disambiguated as it went, using a moving centroid to choose the nearest name match based on the places it has for the person up to that point in processing. For example, Benjamin Franklin was born in Boston, so "Cambridge" encountered early in his article would choose the one in Massachusetts. Later in life, he has many sightings in and near London, which would pull his centroid across the Atlantic. If "Cambridge" were encountered at that point in processing, the university town in the UK would more likely be chosen.

This process introduces an inconsistency: If there are confirmed coordinates from Wikipedia, any amount of jumping around the globe is tolerated, whereas if GeoNames is used, the search will stick closer to "home." This causes new errors. For example, Martin Luther King Jr.'s page discusses his trip to India. Because India is not linked, and King was mostly active in the southern U.S., the sighting is created with a Mississippi town called India, rather than the country. In a naive GeoNames search, the country would have come first and been accepted.

Another new problem was created by this solution: What happens if no centroid can be calculated, because this is the first sighting? Having no centroid at all would create not-a-number exceptions during processing, but using a dummy like 0.0, 0.0 causes bias. Someone born in London could be assigned to London, Nigeria at roughly 5.7, 5.8, since that is closest to 0.0, 0.0 of all the candidates. Now an Englishman's centroid is established in Western Africa, an error that can cause future incorrect choices. Multicrawler combatted this issue by assigning the default beginning point to 43, -37, the midpoint between London and New York City. This was a major assumption, but it capitalized on the known bias of English Wikipedia towards North America and Western Europe. Theoretically this choice could cause problems for the places furthest from 43, -37 (e.g. East Asia), but in practice, this did not happen because there is much less crossover between toponyms in Asia and America or Asia and Europe than between Europe and America. There is not an additional place called "Tokyo" in, say, Wisconsin, so the right one in Japan is chosen regardless of the dummy centroid. In short, the dummy starting centroid was easily overridden when processing most articles about non-Western figures, while often preventing articles about Western figures from starting off on the wrong foot.

The code for multicrawler was much more complicated than tidycrawler, and unfortunately its results appeared to be, if anything, worse. A single error could tug the person's centroid far away from where it was supposed to be, causing further errors. One possible cause is that Buscaldi and Rosso's work was based on a corpus of nonspecific everyday text, such as news articles. News articles tend to have to do with one topic. This assumption does not necessarily hold true for a person's biography. In fact, it seems as though prominent people travel a great deal, especially in recent times. Wikipedia includes more Bill Clintons than Emily Dickinsons.

The fourth version (cascadecrawler) attempts to balance the best parts of tidycrawler and multicrawler. It does away with all the calculations of centroids and distances and tries three options in turn. If there is a linked Wikipedia article with coordinates, those are used. If there is a place with the same name already in that same person's sightings, it is reused. That way, if Birmingham appears in Martin Luther King's article with a link to the one in Alabama, that sense is retained for the entire discourse (article, in this case). If there is no link, cascadecrawler checks whether there is already a place stored that's marked as "default"—from GeoNames. If so, that is used, and if it appears to be the same as a prior Wikipedia match (within one degree of latitude/longitude) the prior Wikipedia match is marked as default. If not, the top GeoNames result is used and added as the default. In short: the top GeoNames result is the default choice for any toponym unless there is different information directly from Wikipedia or earlier in the same article, in which case multiple places can share a name.

This final version appears to have the best results. It allows both Birminghams to appear correctly for Priestley, Watt, King, and Shuttlesworth, but it prevents the Philadephias in Greece and Germany from sneaking in. To be sure, it still causes errors, but the errors are much more traceable than those caused by the moving-centroid approach. For example, it is surprising to find the U.S. President Zachary Taylor in deep South America, but when you realize that GeoNames's top result for Florida is in Uruguay, it is easy to understand how the error arose.

6.6 Child entities

The tidycrawler version of Kivrin added parent entities for each new place: a country and a first-level administrative unit (a state or county, depending on the country). When Philadelphia was created in the database, Pennsylvania and the United States would be created if needed, and Philadelphia's id would be appended to an array of children for each.

Then, on the front-end, if a user searched by name for Pennsylvania, a sighting in Philadelphia would be returned, because the search would include all of Pennsylvania's children.

This feature was removed from the final version for several reasons. First, it conflicted with the multimatch functionality. To avoid accidentally appending Birmingham, AL as a child of the West Midlands county or the United Kingdom would require additional GeoNames queries, burning through the permitted number of queries too fast and requiring another round of iterating through results to find the nearest one. Second, the size of comparable-level entities varies by country. It might be reasonable to search Monaco and find all the sightings associated with it and its child entities. After a crawl of any significant size, searching United States, however, is likely to search thousands of child places and take too long. Third, conceptually, the user can use the polygon selector if she is interested in all the sightings in a region; it makes sense for the place name search to only return sightings involving literally that place name.

7 Data storage

MongoDB is a schema-free, document-based database system, designed to be flexible and allow easy horizontal scaling.

MongoDB has very convenient features for handling geospatial information; in fact, my discovery of these features during my last project inspired this one. It is easy to create a field with latitude and longitude and then have a two-dimensional geospatial index on that field. There are also expressions to search within a polygon or a circle with a given center and radius that match up with the desired front-end design for Kivrin.

MongoDB operates as a series of JSON documents and can be interacted with on the command line using a JavaScript shell. The items in the collection are JSON documents, so a search might return:

```
{ "_id" : ObjectId("4f765ae1bbe04eee951129e1"), "coords" :
[ 28.63576, 77.22445 ], "numSights" : 7, "place name" : "New Delhi" }
```

Insertions and updates are JSON documents as well:

```
db.people.insert({ "title" : "Ferdinand II, Holy Roman Emperor",
"birth" : 1578, "death" : 1637, "description" : "Holy Roman Emperor" })
```

Queries are also JSON documents.

```
db.people.find({"title":"George McGovern"})
```

Using operators like `$gt` for greater than, it becomes easy to compose complicated queries. For example, to find the percentage of people in the database who were born in the 20th century:

```
db.people.find({"birth":{$gt:1900,$lt:2001}}).count() / db.people.count()
```

MongoDB is very easy to use in terms of "upserting" data (insert if does not exist, otherwise update). The user can create any database or collection simply by trying to connect to it. Thus the code at the opening of the crawler is the same regardless of whether it is starting a new run or picking up an existing one. Similarly, the command "ensureIndex" either creates or simply confirms the existence of an index. Upsertions are used throughout the crawler code. When outbound links are added to the queue, they are upserted, with the new score incremented onto the old one if the article already existed. The small amount of data that is preloaded to the places and rejects collections is done as an upsertion; otherwise these items would get duplicated if the crawl is started and stopped.

MongoDB has drivers for Java and Python that are used in this project. The Python code looks similar to the command-line code, but the Java implementation is rather different; each JSON document item must be created as a `DBObject` before it can used as a query, insertion, etc.

```
BasicDBObject newPlace = new BasicDBObject();
newPlace.put("placename", candidatePlace);
Double[] coordsWinner = new Double[2];
coordsWinner[0] = winner.lat;
coordsWinner[1] = winner.lon;
newPlace.put("coords", coordsWinner);
places.insert(newPlace); //this could be done on one line in the shell
ObjectId place_id = (ObjectId) newPlace.get("_id");
```

8 Retrieval

8.1 Ordering results

The front end displays people ordered by the number of sightings in the time/place requested. This promotes higher quality results. Detailed articles that result in a larger number of sightings are likely to have more sightings in the requested span, so they appear higher in the list. More importantly, false matches are likely to be one-offs, and fall to the bottom of the results list.

It would be possible to improve the perceived quality of search results by only displaying people with more than one sighting—thus excluding the one-offs. I did not choose to do this, however, in the spirit of allowing users to examine their own results.

8.2 Weighting places

Places are weighted by the number of sightings associated with a place. If a user draws too large a search area on the map, they may select hundreds of places. Then sightings need to be found for each of those places in turn, which can be very time-consuming. We can limit the number of places searched, but this may result in insignificant places being searched and significant ones excluded. By ranking the number of sightings for each places, we can promote important places and make sure they are included if the number of places picked

up by a query exceeds one hundred. If the user wants their search to include more small places, they can draw a smaller shape.

The number of sightings logged in the database for this weighting often does not exactly match the actual number of matching records in the sightings collection because the count is incremented before a person's timeline is de-duplicated. (If the exact same place/year combination appears more than once, it would be redundant to repeat it in the results.)

8.3 Context

Each sighting is stored with the sentence that gave rise to it, to supplement the results with context. The sentence often says why the person was in that place or what they were doing there. It is also useful for troubleshooting—the source of strange results can be seen immediately. Wikipedia mark-up and HTML should be stripped out, although section headers sometimes remain.

9 Implementation details

9.1 Most important functions

CascadeCrawler (main)

This part of the program controls the crawl. At the beginning is a static Boolean variable that controls whether the crawler will operate in a verbose troubleshooting mode or not; the default is `false`.

The crawler starts by connecting to MongoDB, opening the database, and then connecting to the different collections (e.g. queue, people, etc.). The queue is then seeded with a selection of historical figures spread across time and geography.

The queue is processed until it is empty. Two-thirds of the time, the crawler pops the article with the highest score, but when the counter mod 3 is zero, it pops the first (oldest) item. The article is added to the alreadyIndexed collection.

Next, there is code to check whether the page has already been processed. This is important because of redirect links: Franklin D. Roosevelt and FDR are different titles that access the same page. Also, this step checks a time-stamped collection of already processed articles; in the future, this could be used to reprocess articles that are old.

The program makes several calls to the Wikipedia API. First, it retrieves the article's id, which it needs for the template and link queries. Then it checks whether the page uses the Persondata template. If it is, is passes the article title to a `PageProcessor`.

When the article's count of sightings is returned to this method, the crawler accesses a JSON-formatted list of outbound links from the Wikipedia API and queues all of them, incrementing their score by the number of sightings found in the present article.

This class also contains a counter to intermittently display stats on the progress of the crawl, and various error handlers.

PageProcessor

This part of the program controls the processing of a single biographical page. It fetches the article text, does some preliminary work on it, and then passes it off to have spatio-temporal information identified.

After retrieving the article text, the page processor's first task is to extract the birth and death dates and places. This appears as a part of the Persondata template, which is metadata, not formatted for humans to read. It also gets the person's description.

Regular expressions are then used to trim the article text by removing the references, bibliography, etc., which is the source of many false matches.

The cleaned-up text is passed to the `PlaceTimeFinder` to extract a sorted set of "sightings" of the person. The `PageProcessor` receives this back, matches it with the metadata about the person, and then uses this to insert the person record and all the sightings into the database. If the birth and death dates need to be estimated, this is done by the `PageProcessor`.

PlaceTimeFinder

This class starts by creating a local stoplist of words in the person's name, unless they appear after a word like "of." For example, "Franklin" should not be allowed to be a place in the article about Benjamin Franklin, but in the article about John of Ghent, "Ghent" should be allowed.

Then it examines the article text one sentence at a time. It looks for a time in the sentence, and if it can't find one, it tries to use a valid time from a previous sentence. (If it can't find a time, it drops sentences until it finds one.) If the person was born before 1000AD, it uses a complicated regular expression that handles AD/BC/BCE dates. If not, it more naively (and quickly) looks for four-digit numbers without commas. These almost never appear except as years.

If it has a time in hand, it looks for potential places in the sentence—capitalized phrases that appear after a preposition or a word like "visited." Each candidateplace gets sent to the `PlaceEvaluator`.

If it is able to pair a time and a place, it puts them in the Sorted Set that gets returned to the `PageProcessor`.

PlaceEvaluator

The `placeAnalyzer` takes strings that may be the names of places and decides how to handle them. First it checks a list of known rejects (like "January"). If it hasn't already been rejected, it checks whether it has the form of a Wikipedia link ([[like this]] or [[another link format|this]]).

If the string has the form of a Wikipedia link, it passes it to `WikiCoordsFetcher`. If not, or if the Wikipedia check fails, it tries to reuse a place with the same name that has already been found for this person. If that is not possible, it passes the string to `GeoNamesFetcher`. It either receives a `IdAndCoords` back, which it passes up to `PlaceTimeFinder`, or it adds the candidate place name to the list of rejects.

9.2 Helper classes

PlaceTime

This class gives a structure to pairs of times and places. The times are integers, the places are ids from the places collection in MongoDB, and it also stores a string of context (the sentence that gave rise to the sighting pair, as a sort of keyword-in-context).

The class also implements a comparator that ensures only unique combinations of times and places can be added. A person can be multiple places in one year, or in the same place in different years, but there is no point in having duplicates of identical sightings.

LatLon and IdAndCoords

These classes simply give an easy way to hold geographic data together while processing.

LoadFeatureClasses and LoadPlaceTypes

This class creates a reusable HashSet of the Feature Classes from GeoNames that are acceptable in the context of this project. `LoadPlaceTypes` does the same thing, except for populated places, which are handled in more detail.

NoneSuitableException

This exception handles cases where there is no place id to return to the calling function (either because nothing could be found at all or because nothing acceptable could be found). Sometimes it is caught without further action being taking but if it bubbles up to the `placeAnalyzer` with no further places to check, it creates or increments a reject.

SetOfPlacesToCheck

This makes it easy to pass the set of relevant MongoDB collections between functions. Also, when first instantiated, it preloads small sets of useful information: places that should be handled but don't appear in GeoNames with exactly that name (e.g. "New York City") and terms that appear in place-like contexts but should be rejected (e.g. "January").

TimeStamp

This creates a nicely formatted timestamp to appear at various times during the crawl, for monitoring.

URLConnectionReader

This is used to open web pages (mostly queries to the Wikipedia API for metadata, since the pages themselves are accessed through JWBF).

GetMode

Calculates the mode of the observed dates to estimate a lifespan when none is provided.

9.3 Included packages

MongoDB

The driver for MongoDB offers functions to connect to and manipulate database objects.

GeoNames

The GeoNames API allows the user to query GeoNames for a toponym string and retrieve a list of matching geographical entities. It also returns metadata about the places, which allows filtering and disambiguation in the client application.

Jackson

Jackson is a JSON parser for Java that is needed to process information from the Wikipedia API, e.g. the list of outbound intrawiki links in each article[33].

JWBF

The Java Wikipedia Bot Framework provides easy access to MediaWiki API functions, like retrieving the page content (as distinct from manually accessing the HTML page content, which Wikipedia asks users not to do)[34].

Others

Other included packages are dependencies of those described above.

9.4 MongoDB collections

MongoDB collections are like the tables of an SQL database, except that the documents stored in them can be of any "shape"—if a person lacked a description, that can be left out without any problem. Every record is automatically assigned a unique id.

Because MongoDB is a NoSQL database, though, there is no entity relationship diagram showing how the collections fit together; instead, examples are given.

people

```
{ "_id" : ObjectId("4f765ef7bbe04eee95113019"), "title" : "Indira Gandhi",
"birth" : 1917, "death" : 1984, "description" : "Prime Minister of India" }
```

The people collection contains the basic metadata about each person for whom there are sightings. This collection is indexed on the title (person's name) to allow for name searches on the web.

places

```
{ "_id" : ObjectId("4f7b1145bbe0a9784d904b29"), "coords" :
[ 55.75222, 37.61556 ], "numSights" : 1450, "placename" : "Moscow" }
```

Place records contain a toponym, coordinates, and the number of sightings associated with it. If it is the primary result from GeoNames, it includes "default":true; otherwise that field is missing. This collection is indexed by the place names and coordinates, for searching on the web, as well as by the number of sightings, to allow ordering of places to search when the user's selection is too large.

sightings

```
{ "_id" : ObjectId("4f649ac0bbe034d1ff139381"),
"place" : ObjectId("4f649a75bbe034d1ff139306"),
"person" : ObjectId("4f649ac0bbe034d1ff139376"), "time" : -78,
"kwic" : "Hearing of Sulla's death in 78 BC, Caesar felt safe enough to
return to Rome. " }
```

A sighting combines a time (expressed as an integer between -3000 and 2012), with the ids of the person and place in question. The sentence that gave rise to the sighting is also saved as keyword in context (KWIC). For the sake of brevity, the context sentence is cut off after 200 characters, so sometimes the context is lost in the case of extreme run-on sentences. This collection is indexed by the time to accommodate the user's search range. It is also indexed by the place to allow the look-up associating places and sightings to happen faster.

queue

```
{ "_id" : ObjectId("4f6529f8c9a1bb766b506a46"), "score" : 531, "title" :
"Jeanne Moreau" }
```

The queue combines the title of the article with its current score, which is incremented when more pages link to it, as described in section 6.1. The queue is indexed on the title (to speed upsertions) and the score (to make it faster to identify the currently high-scoring item).

alreadyIndexed

```
{ "_id" : ObjectId("4f649df2bbe034d1ff139978"), "title" : "Aurangzeb",
"date" : ISODate("2012-03-17T14:21:38.121Z") } //valid person
{ "_id" : ObjectId("4f64a098bbe034d1ff139de8"), "title" : "Capitalism",
"date" : ISODate("2012-03-17T14:32:56.200Z") } //not a person
```

This collection is checked to make sure that pages are not being processed repeatedly even if they are linked to a lot. It contains both valid biographies and non-biographies. This collection is indexed on the title, to make it easy to check pages.

rejects

```
{ "_id" : ObjectId("4f64f8b8bbe021436bd46ef1"), "encounters" : 132,
"term" : "Henry" }
```

This collection keeps track of what terms have already been rejected as non-toponyms, to avoid requerying GeoNames. The rejects are indexed on the term.

9.5 Front-end

The web front-end of Kivrin is written in Python with some JavaScript. There are four pages:

Homepage

The search screen briefly explains the project and allows the user to search three different ways. The geographic search (default) uses JavaScript written by Marcelo Montagna to select a map area by a rectangle or circle. I modified this code to restyle it, resize and reorient the base map, and remove additional features that are unneeded.

results.cgi

This script translates the search area into a MongoDB query of the places collection, except that the radius units for MongoDB are degrees rather than miles, so the user's radius in miles is divided by 69 to estimate the degrees.

The script then looks up 100 places in the user's range. This limit was put in place to prevent the application from seeming too slow in cases where a user's selection is very large. The top hundred places are chosen based on the number of sightings they have through the entire project. At this stage, the system doesn't know if there are any sightings in the time range, but this sorting means that the most important places are always included. Otherwise, it might be possible to search Southern California and have Los Angeles happen to not make the cut. Also, sorting the places means that the display of "places searched" always names the most important ones first, which should confirm for the user that the system is working reasonably. If the user needs a greater number of less significant places to be included, he could zoom in and search a smaller map area; it should feel natural to see results for smaller places represented in a search of a smaller area.

For each of the places found, the sightings collection is searched for sightings including that place, and in the user's time range. Note that in cases where there are synonymous or near-synonymous places with similar coordinates, both would be included in the search, avoiding the need for entity resolution to respond to the user's request.

Next, the sightings are grouped by the person and sorted by date for each person. They are displayed with the people with the most sightings first.

timeline.cgi

This script searches people's names with a regular expression and asks the user to confirm which exact person is meant if there is more than one candidate. Then it displays all the

sightings for that person in chronological order. It is mainly intended as a troubleshooting tool.

toponym.cgi

This script allows the user to search by a place name rather than coordinates. If there is more than one candidate, the user is shown a list of possibilities with coordinates and a small spotting map for context. When the user confirms the coordinates, the query is bounced back to `results.cgi` as a search for a single place.

10 Evaluation

The evaluation was performed on a static copy of the final database that contained slightly more than 16,000 people and just short of 1 GB of data. The crawler was then restarted to continue building the data that is currently available on the website.

Where random selections were required, I used a random number generator and then skipped that many items, like this:

```
db.people.find().skip(8677).limit(1)
```

10.1 Identification of biographies

The Persondata template is an almost flawless way of identifying biographies; more interesting is the question of whether the crawler promotes biographies in its queue.

A sample of 100 dequeues was taken at three points in the crawl. There was an average of 1.3% examined articles that were biographical but were ignored because they lacked the Persondata template. None of these articles was high quality (for example, they were marked as lacking citations) so it is just as well to ignore them. An average of 6% of the examined articles were flawed titles (redirects to another article or nonexistent pages in Wikipedia) that were correctly passed over. These statistics were consistent at different points in the crawl.

The balance between correctly identified people and correctly discarded non-people changed through the crawl. Early on, there were 63 people and 30 non-people. Several thousand dequeues later, there were 43 people and 50 non-people. In the last sample, there were 34 people and 58 non-people. On average, 46.7% of dequeued pages were valid biographies.

In a previous version of the crawler that did not prioritize the queue, I observed that an average of 12% of articles are biographical; yet the final version of the crawler finds a much higher proportion. This shows that the crawler, in its attempt to process the queue in a way that prioritizes important articles, is also prioritizing biographies, especially early in the crawl, probably because they tend to link to each other.

10.2 Identification of places

Fifty places were sampled to see if they were, in fact, places (as opposed to personal names or other stray words). Of the fifty, 33 were indeed places and appeared to be the correct

one to match the sightings associated with them. (I did not check that the sightings were 100% correct when there were dozens or more, just that the main sense of the place name was selected or a plausible secondary one for those sightings.)

An additional three places out of the fifty were places, but the wrong one was selected. For example, a place called Bickley in the U.S. was selected instead of one in the U.K. Two others were parsed incorrectly, due to Unicode in the place name. It is difficult to generalize about these examples, so they are left out of the further analysis in this section.

That left twelve out of the fifty as not places. Most were personal names, such as Alexandra.

Looked at by the count of places, this shows a rate of 66% correctly identified places and 24% terms misidentified as places, which is a large percentage of the data. However, the distribution of sightings across places is not consistent. Very many places are associated with a single sighting; others have just a handful; and a small number have hundreds. In this sample, more than half of the places had a single sighting, whereas Toronto had over 300.

If we instead look at the percentages by how many *sightings* associated with the fifty sampled places were assigned to a real place (87%) versus how many were assigned to a non-place (6.6%), the proportion of dead wood in the database looks better.

10.3 Plausible results for a place and time

As with many search engines and similar projects, it would be very difficult to calculate recall for Kivrin, because we have no definitive source of information on what people *should* be retrieved for a given time and place. Still, I attempted to calculate a rough kind of precision for the results: Of the people retrieved for a search, how many of them were plausibly at that place and time? As stated in the introduction, the goal is to retrieve candidate biographies for a user to consider, so I did not insist on a perfectly clear-cut fact to consider a result correct. I counted a person as correct if they were at the location or had a significant influence on it (such as starting a war there).

I did four test searches: Philadelphia region, 1760-80; Berlin, 1942-43; Paris, 1200-1300; and Vietnam region, 1970-74. Of 236 people found, 195 were correct, or 82.6%.

The first three searches were more accurate at 92%. The search for Vietnam had more errors because many people opposed the Vietnam War without going there or influencing the region. Also, there is a place called "No" in Vietnam which got picked up for 13 entertainers with a song that went "to No. 1." These were one-off sightings that appeared, as desired, at the bottom of the results list.

On the whole, if a user searches for a place for which there is data available, they will get useful results.

10.4 Complete results for a person

Obviously, this form of evaluation would be much better performed by unbiased volunteers without access to the system results. Still, I attempted to read through twenty randomly selected articles processed by the system, and mark places/times I felt were clearly associated with the person. I then compared these results to Kivrin's.

Of 165 sightings shown by Kivrin across these 20 biographies, 115 were right and 50 were wrong. Also logged were 68 places that I thought should have been included but were not.

A 30% error rate is disappointing, however, a number of the errors could have been avoided with a single change: making sure to exclude any filmography section, as I did for bibliographies. Because these sections are filled with dates and capitalized words, they are a minefield of false hits. Many of the remaining mistakes were personal names.

Similarly, Kivrin found 115 correct sightings but should have found 183, missing 37%. Many of these errors fit into one of two categories. I made inferences while reading, such as that a person who is the director of the London Symphony Orchestra is probably in London. Kivrin looks for motion-related verbs and prepositions so will not catch such information. Also, the preposition "of" is not included, because it led to many false drops during testing. Unfortunately, it also led to every battle (e.g. Battle *of Gettysburg*) being ignored, and the test set included seven people with military experience.

10.5 Some additional statistics

Sightings per place and person

There are 12.3 sightings per person, on average. There are 7.4 sightings per place on average. The place with the most sightings (over 4,000) is London, England. About 13,000 places, or 47%, have only one sighting.

People per century

I performed a count of the number of people born in each century to document the recency bias of Wikipedia. People without birth dates (of which there are 877 or 5%) are not included. Allowing the date to go back to the beginning of recorded history has not resulted in any sightings in the early millennia. Meanwhile, there is an order of magnitude difference between the millennium from 1 to 1000 (152 people) and the millennium from 1001 to 2000 (15,373).

People per initial

I checked how many people had been indexed by the title of the article. Because Wikipedia only allows users to retrieve the first 500 outbound links from an article, alphabetically, we should expect some "alphabet bias" in what people are available to be queued. It is impossible to say what proportions we should expect, given uneven use of letters across different languages. In the sample, however, we see 1,941 for A, 1,144 for B and 1,155 for C, yet only 594 for S and 510 for T—an extreme enough difference to suggest that late-alphabet articles are having a harder time making it into the queue. A user is unlikely to notice this in their results, but until the crawl is completed, it will have a significant effect on their results.

Table 1: Distribution of births

Timespan	Number of births
-3000 - -2000	0
-2000 - -1000	0
-1000 - 0	22
Century starting 1	17
101	3
201	2
301	12
401	9
501	11
601	27
701	7
801	25
901	39
1001	40
1101	62
1201	95
1301	63
1401	106
1501	142
1601	174
1701	1316
1801	3965
1901	9410
2001	4

11 Limitations

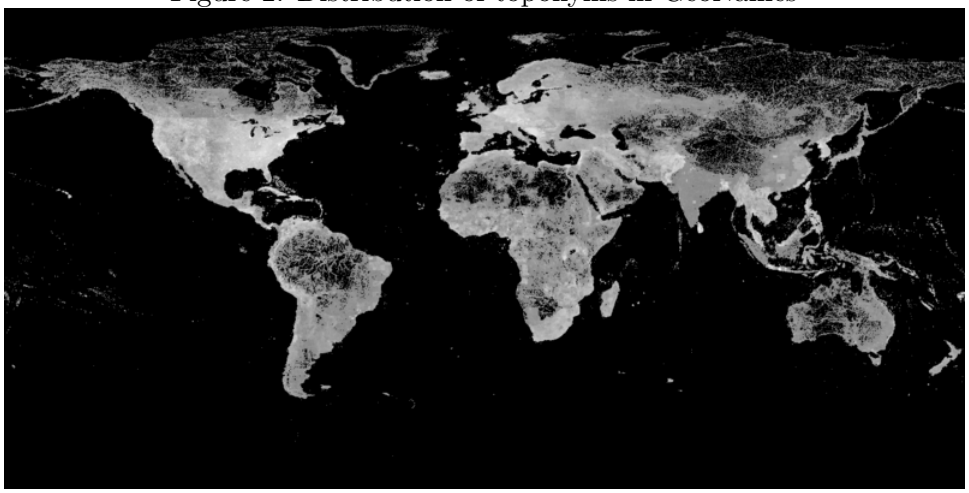
Some aspects of the project were constrained by qualities of the source material in Wikipedia.

11.1 Geographic bias

Wikipedia, specifically its English version, has a well-documented geographic bias in favor of articles about people, places, and events in North America and Europe. In 2009, Mark Graham analyzed almost half a million geotagged articles in Wikipedia to create a map of the intensity of Wikipedia coverage by country, which he then normalized by land area and population[35]. Discussing his results, he observed, "[A]mazingly, there are more Wikipedia articles written about the fictional places of Middle Earth and Discworld than about many countries in Africa, the Americas and Asia."

A visual comparison of the distribution of toponyms in GeoNames[36] versus geotagged articles in Wikipedia[37] visually reveals the extent to which certain parts of the world are neglected.

Figure 2: Distribution of toponyms in GeoNames

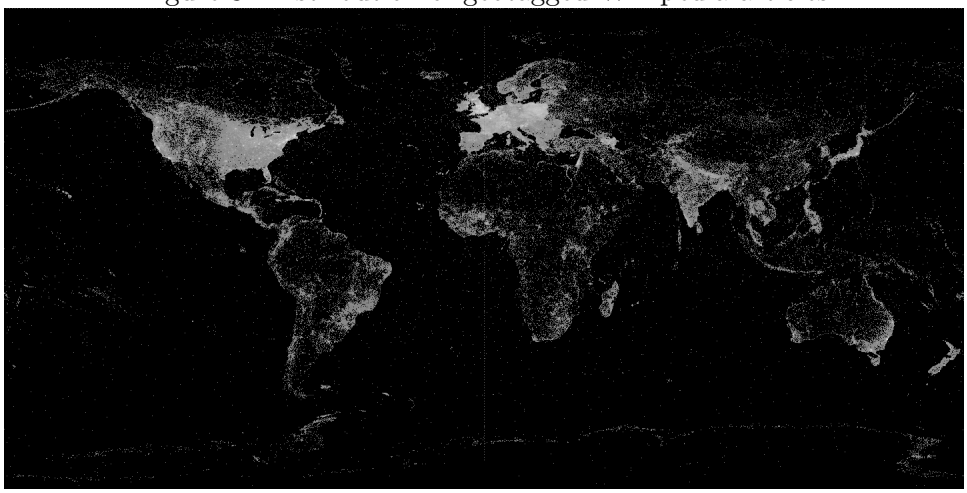


Biographical pages are not geotagged, so they are not included in the map, but it is likely that they are similarly affected. My attempts to include non-Europeans/non-Americans like Jomo Kenyatta among the seed articles may be able to promote inclusion of the non-European/non-American people that *do* exist in Wikipedia, but it is not realistic to aspire to unbiased geographic coverage when the underlying content is biased. Accordingly, searches outside the western world, particularly further back in history, have sparse results, usually with a more noticeable proportion of mistakes.

11.2 Recency bias

In his recent book *The Better Angels of Our Nature*, Steven Pinker discusses the idea of "historical myopia: the closer an era is to our vantage point in the present, the more details

Figure 3: Distribution of geotagged Wikipedia articles



we can make out." In one experiment, "Internet users [were asked] to write down as many wars as they could remember in five minutes." He reports that "the responses were heavily weighted toward the world wars, wars fought by the United States, and wars close to the present[38]."

Wikipedia shares this bias, as is evidenced by the distribution of people indexed by Kivrin by century of birth. This has an effect on the granularity of data available at different times. Searching a single year for ancient Rome will have much thinner results than searching a single year for contemporary Los Angeles.

11.3 Evolving toponyms

Wikipedia articles are reasonably consistent about using names like Constantinople and Istanbul appropriately when the change happened long ago and different names are clearly associated with distinct historical periods, but there are more problems where toponyms changed recently. For example, many sightings in Burma get erroneously assigned to a town in Kazakhstan because Wikipedia prefers the name Burma to Myanmar and GeoNames chooses the opposite. In some cases, there remains a dispute over the name for a place, as in the Falkland Islands/Malvinas. Results for places without a settled toponym are incomplete.

11.4 Inconsistent templating

Wikipedia makes use of templates to elicit certain metadata elements from authors. It also uses a non-HTML mark-up format that is meant to be easier for contributors to use while also adding intelligent features, like automatically linking citations to a bibliography. However, given the number of Wikipedia contributors, it is not surprising that real-life Wikipedia text is more inconsistent and flawed than the ideal model.

The notation of birth and death dates and of geographic coordinates provide examples of this. Dates are encoded in the Persondata template in a variety of ways. Most of these can

be used, even though a significant number use date templates that have been discouraged.

```
1790-01-12
1706|1|17|mf=y [month-first format]
December 10, 1790
ca. 127 BC [and many other formats for estimating dates]
```

Here are some of the different ways geographical coordinates are encoded, which is especially surprising given that there is a project specifically intended to regularize this[39].

```
coord|52|48.2|18|N|2|08|59|W
coord|1.05|S|36.92|E
coord|41.3086|-72.9276
lat_deg=37 |lat_min=58 |lon_deg=23 |lon_min=43
latitude = 48.8567 |longitude = -2.3508
latd=41 |latm=11|latNS=N |longd=73 |longm=58 |longEW=W
latd=41 |latm=11 |lats = |latNS=N |longd=73 |longm=58 |longs = |longEW=W
latd=41 |latm=11 |lats=47 |latNS=N |longd=73 |longm=58 |longs=1 |longEW=W
```

Difficulty recognizing these particular pieces of data has a noticeable effect on the quality of indexing. When birth and death dates cannot be recognized, the `PageProcessor` must estimate the person's lifespan. When coordinates are present in an article but cannot be recognized, this forces a search of GeoNames.

11.5 "Unencyclopedic" or trivial content

What does and does not belong in Wikipedia is a matter of active debate. Most contributors agree that trivia is not appropriate and that there should be some threshold for significance. Autobiographical vanity entries are deleted, for example. Still, Wikipedia has many more entries than a print encyclopedia which means that Kivrin will discover biographies that may not seem important by print standards. The queue design attempts to steer the crawler towards more important entries.

Also, individual Wikipedia pages sometimes contain information that seems unimportant or irrelevant. For example, an earlier version of the crawler that was less precise in matching dates created a sighting for the pop singer Boy George in Cyprus in the year 26—a laughable error caused by an entire paragraph in his article about a looted religious icon he inadvertently bought.

Kivrin assumes Wikipedia articles are about their subjects and doesn't make any attempt to recognize when they veer off to discuss fringe topics. The presence of "unencyclopedic" content results in false drops.

11.6 Reading level

One study (which may not be directly applicable, as it analyzed pages with health information) calculated the reading level of Wikipedia articles based on sentence length and structure. In their sample, the average was 14.1, or college level[40]. These longer sentences make the text more difficult to interpret accurately. A Simple English version of Wikipedia exists, but its extent is not at all comparable (it only has about 81,000 articles)[41].

11.7 Nested tags

At times Wikipedia's simple-seeming mark-up piles up in a way that makes it difficult to manipulate. One example is the use of {{double curly brackets}} to indicate a template. Templates can be nested inside each other, such as when a photo caption includes a citation. This makes it difficult to thoroughly scrub the text of mark-up using regular expressions without risking catastrophic backtracking. Nesting is the cause of mark-up appearing in search results.

12 Further Work

There are a number of improvements that could be made to Kivrin, either with more coding on the project itself or if updates were made to the tools on which it relies.

12.1 Parsing

Kivrin's `PlaceTimeFinder` relies on regular expressions in several places, for example, to identify potential places and times in the text and to match coordinates from Wikipedia. This reliance on regular expressions to identify interesting text seems inefficient, especially since additional processing is sometimes required to handle the text once it has been identified as a phrase of interest. Better results might be achieved by processing the text through a customized parser. It would be necessary to first find the Persondata template, which appears at the end of each article, to get the birth and death dates that serve as guardrails for other observations. After that, the body of the article could be processed linearly.

If each sentence were parsed, identifying its structure better, it might be possible to ignore (some) sentences that are not about the subject of the article. Sometimes these imply information about the subject's whereabouts or influence, and other times they do not, so it is not clear if universally ignoring such sentences would be an improvement.

12.2 Supervised machine learning

A large proportion of the errors remaining in the dataset are caused by named entities that Kivrin interprets as toponyms, but which are actually personal names or words in the title of a book, film, etc. In other words, while final version of the multimatch process was reasonably successful at resolving *referent* ambiguity, the question of *semantic* ambiguity was not addressed, passing flawed candidates into the multimatch process to begin with. A sufficiently large annotated corpus of similar text tagged for people and places would make it possible to use machine learning methods to avoid mistakes based on personal names.

First, we could analyze which names are for places versus people. About Annie Besant, we read "Besant returned to Frank to make a last unsuccessful effort to repair the marriage." Frank is her husband's name, but it could be replaced with "Chicago" and still be grammatical. It may be possible to learn whether names like "Frank" are place or personal names. This could be approached in two ways: either the system could attempt to identify a surface form like "Frank" with a long-form name earlier in the article, or it could simply learn from the corpus that "John" and "Obama" are almost never places, without caring which "John" or "Obama" is meant.

Second, we could analyze which verbs are followed by personal names and which by toponyms. Kivrin is set up to recognize certain obvious, motion-based verbs (like "visit") but mostly looks for prepositions. This makes it vulnerable to false hits like "married to Theobald." The system could learn that "married to" does not involve a place but "invaded" does.

12.3 Sentence tokenization

The project relies on Java's BreakIterator class, which is not ideally designed for this use. The crawler uses `getSentenceInstance` to chunk the text while looking for facts. This tool takes abbreviations for the default locale (en_US) into account (such as Mr. and Dr.) but is frequently fooled by proper names like "U.S." or "Lyndon B. Johnson" or "St. John's." A customized tokenizer could take into account patterns like state abbreviations and abbreviations like "Mt." and "N."–patterns that appear often when describing geographic places.

In the current version, a legitimate place could be discarded if it appears before a year in a sentence that got split in two incorrectly. The most significant error caused by incorrect sentence breaks is the assignment of Washington to a location in England, even though Washington, D.C. is almost always intended. The full name of the American city gets broken into the nonsense "Washington, D." so that even a legitimate link to the Wikipedia page gets broken. When just "Washington" is looked up, the top hit is in England. The toponym Washington, D.C. never gets included in the places collection at all.

12.4 More complex multimatch heuristic

Setting aside the coordinates from Wikipedia, the various multimatch approaches I tried rely on either importance or proximity. The final version (cascadecrawler) threw out proximity and went back to using importance as the deciding factor when no other context could be derived. It may be more effective to strike a balance between proximity and importance by allowing proximity to be considered when one place is not *clearly* the most important one with that name.

For example, London, England, has a population of nearly eight million and is a national capital. The places in the United States that share the name have less than ten thousand people each and are not capitals. Even London, Ontario, is only a county seat, with less than half a million people.

By contrast, Birmingham, AL and Birmingham, England each have between one hundred thousand and one million people and are of the same place type in GeoNames, "seat of a second-order administrative division."

A more subtle multimatch heuristic could select London, England as the only valid London because it is of a more important place type and has more than an order of magnitude more people than any other place sharing the same name, but could perform the disambiguation process for Birmingham, because neither candidate is clearly more important.

Of course, in this example, using Wikipedia solved it correctly anyway, but this more subtle heuristic could work in other instances. It would require the addition of rules about what counts as a sufficient jump in importance like how much difference in population is

significant. It would probably also require ranking the many different kinds of GeoNames feature codes to assert which are more important than others.

12.5 Pleiades

Late in working on this project, I learned of the Pleiades database of ancient places. This resource includes coordinates for ancient places and connects names from different eras, like Gangra and Germanicopolis[42]. Adding this as a gazetteer option (probably only in cases where the person's birth occurs before a certain date, say 500 AD), would reduce the haphazard matching of names of ancient places when they are only available from Wikipedia. More complete data would improve the strength of the association between places (evolving, human-named notions of locations) and spaces (physical locations) in the system.

Using data from Pleiades would still leave a gap between ancient and modern places. For example, a place like Carniola, a medieval state in the Holy Roman Empire, appears in neither Pleiades nor GeoNames, and does not have coordinates in Wikipedia.

12.6 Time normalization

Kivrin does not extrapolate relative times from phrases like "at age 18..." which are relatively rare and mostly appear describing someone's youth (rarely does an article say, "at the age of 47..."). The same is true of phrases like "ten years later..." The argument against doing this would be that matching any significant proportion of these phrase variations would be time-consuming (most sentences do not contain them) and that good Wikipedia text would introduce a major time or place shift using clearer, more spelled-out writing.

Annotation of dates at a more granular level could be supported by a tool like the HeidelTime system for extracting and normalizing temporal expressions[43]. This would require further thought about how to treat user queries for time spans that differ in granularity from the attested data derived from Wikipedia articles. Kivrin will already return a result if someone is in New York City on 9/1/2001 and the user searches for 2001. The question is what happens if the user searches for a specific date. Implementing a system like HeidelTime would reveal whether there is actually enough data in Wikipedia at this level of granularity to offer worthwhile results.

12.7 Filter by field of endeavor

Although DBpedia was not helpful for retrieving metadata that stems from the Persondata template, its ontology could help in other ways. One example would be organizing people by profession. This would not be practical directly from Wikipedia, since the quality of information in profession infoboxes varies. (O.J. Simpson has only the infobox for NFL players despite now being known for something else; many people just have a "person" infobox; meanwhile other categories vary in specificity like "artist" or "musical artist.")

Luckily, DBpedia contains data that tries to organize professions according to Wikipedia categories and WordNet: the YAGO ontology[44]. In DBpedia, Andrea Gabrieli has a number of `rdf:type` entries such as `yago:ItalianComposers`[45]. Using YAGO it is possible to go higher in the ontology and discover that the highest-level parent is "Musicians."

Pulling in this kind of data would make it possible to allow users to filter Kivrin's results according to major fields like arts, science, government, etc. Users might find this interesting, although part of the intent of the project was to allow serendipitous discovery across fields of endeavor.

12.8 Discovering nonbiographical articles

Attempting to process nonbiographical articles would probably not have very useful results. First, numerous trivial results would be included. A tool like Kivrin is not needed to tell us that the 1952 Philadelphia Phillies were in Philadelphia in the 1950s. Also, nonhistorical articles do not necessarily proceed chronologically; Radium, for example, jumps around in the 19th and 20th centuries and discusses the element's half-life of 1601 years. Results from an article like this would be very poor. Finally, articles about events like the Storming of the Bastille would be appropriate to include, but I am not aware of a template that identifies them like the Persondata template does for people.

12.9 Three-dimensional index and case-insensitive search

A major change could be made easily if MongoDB implemented three-dimensional (or n-dimensional) geospatial indexing. This feature, which has been discussed but not decided on, would make it possible to collapse sightings and places into a single collection with the timeline as a third, quasi-spatial dimension. Rather than requiring the user to specify a time range, which is in the idiom of a traditional database schema, Kivrin could then look for sightings that are near a pinpoint query like 46, -122, 1980 (the Mt. St. Helens eruption). Effectively, the range of dates searched would expand or contract as needed to get a reasonable number of hits. This form of searching may feel more natural to users.

It would also improve the system if MongoDB made another change that is under consideration: adding case-insensitive search and indexing. Kivrin compensates for this by running user-typed searches in regular-expression mode, but this is sluggish.

12.10 Offline processing

The project could be approached differently by downloading the content of Wikipedia and GeoNames and processing the information offline. If indexing speed were not an issue, the backlinks tool could be queried for each candidate page to determine its importance in the network of Wikipedia articles, allowing results to be ranked by that metric instead of the number of sightings in the search range. It may become desirable to limit the database to highlight only the more important people. Assuming again that more important people have more complete pages, the system could be set to create a record only if the person's article resulted in, say, ten or more sightings (instead of just one in the current version). Similarly, after indexing, the places collection could be pruned over the significant number of places with only one sighting, to speed query processing.

12.11 Scale and speed

The crawler would obviously process pages much faster if it were multithreaded. I did not attempt this since it would simply burn through the available GeoNames queries faster and then sit idle. GeoNames does offer commercial plans, but only the most expensive ones offer more queries per hour than the free plan.

The MongoDB database should scale well. On a database of about 40,200 people, the size of all the indexes was about 174 MB. If the indexes were to grow too large to fit on one machine, MongoDB is designed to allow easy, automatic sharding/partitioning across multiple nodes, which accounts for its popularity with fast-growing start-ups.

All of the MongoDB indexes, including geospatial ones, are implemented as B-trees. It may be more efficient to implement the index on the coordinates as an R-tree, which would also lend itself to creating a three-dimensional index. Still, the B-tree indexes work very adequately for the current volume of data.

13 Conclusions

Despite some errors, the results given by Kivrin in test searches offer plausible information, serviceable enough for general use. The design of this tool shows how a huge volume of user-generated information can be digested into a form that allows access in novel ways.

The queue design takes advantage of what can be observed about linking patterns in Wikipedia to favor biographical entries and results of higher perceived importance.

The multimatch discussion shows that assumptions about place name disambiguation that may be valid in other contexts do not make sense when dealing with text that is itself already a kind of summary that may omit intermediate steps that would otherwise give context.

The data quality would be most improved by using statistical methods to resolve semantic ambiguity, differentiating personal names from place names.

The front end compensates for errors by ordering results by the number of sightings; one-off mistakes drop to the bottom of the list. Use of a three-dimensional geospatial index could allow users to search in a more natural-feeling way.

References

[1] C. J. Davis and K. West, *Women writers in the United States: A Timeline of Literary, Cultural, and Social History.* Oxford University Press, 1996.

[2] E. Zachte. (2012) Wikipedia statistics. [Online]. Available: http://stats.wikimedia.org/EN/Sitemap.htm

[3] C. Willis, *Doomsday Book.* Spectra, 1993.

[4] M. Montagna. (2012) Some experiments with Google Maps. [Online]. Available: http://maps.forum.nu/

[5] R. Güting and M. Schneider, *Moving Objects Databases*, ser. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2005. [Online]. Available: http://books.google.com/books?id=F79ykXkDSDsC

[6] E. F. Tjong Kim Sang, "Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition," in *Proceedings of CoNLL-2002.* Taipei, Taiwan, 2002, pp. 155–158.

[7] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of CoNLL-2003*, W. Daelemans and M. Osborne, Eds. Edmonton, Canada, 2003, pp. 142–147.

[8] N. Wacholder, Y. Ravin, and M. Choi, "Disambiguation of proper names in text," in *Proceedings of the fifth conference on Applied natural language processing*, ser. ANLC '97. Stroudsburg, PA, USA: Association for Computational Linguistics, 1997, pp. 202–208. [Online]. Available: http://dx.doi.org/10.3115/974557.974587

[9] D. Buscaldi, "Map-based vs. knowledge-based toponym disambiguation," *Proceedings of the 2nd international workshop on Geographic information retrieval*, 2008. [Online]. Available: http://dx.doi.org/10.1145/1460007.1460011

[10] E. Rauch and M. Bukatin, "A confidence-based framework for disambiguating geographic terms," *HLT-NAACL-GEOREF '03 Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references - Volume 1*, 2003.

[11] S. Overell and S. Rüger, "Using cooccurrence models for placename disambiguation," *International Journal of Geographical Information Science*, vol. 22, no. 3, pp. 265–287, Mar. 2008.

[12] V. Sehgal, "Entity resolution in geospatial data integration," in *In: Proc. of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems.* ACM Press, 2006, pp. 83–90.

[13] H. Manguinhas and J. Borbinha, "Extracting and exploring the geo-temporal semantics of textual resources," *IEEE International Conference on Semantic Computing*, 2008.

[14] R. Sipoš, A. Bhole, B. Fortuna, and M. Grobelnik, "Demo: HistoryViz – Visualizing Events and Relations Extracted from Wikipedia," *The Semantic Web: Research and Applications*, 2009.

[15] A. Terrell, "Building Timelines Using Wikipedia," 2011. [Online]. Available: http://pages.cs.wisc.edu/~aterrell/wp-content/uploads/2011/02/21/data_modeling_paper.pdf

[16] J. Strötgen and M. Gertz, "HeidelTime: High quality rule-based extraction and normalization of temporal expressions," in *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Jul. 2010.

[17] ——, "TimeTrails: a system for exploring spatio-temporal information in documents," in *Proceedings of the VLDB Endowment*. VLDB Endowment, Sep. 2010.

[18] R. Bunescu, "Using encyclopedic knowledge for named entity disambiguation," in *Proceedings of EACL*, 2006.

[19] S. Cucerzan, "Large-Scale Named Entity Disambiguation Based on Wikipedia Data," in *Proceedings of EMNLP-CoNLL*, 2007.

[20] H. Nguyen and T. Cao, "Named entity disambiguation: A hybrid statistical and rule-based incremental approach," in *The Semantic Web*, ser. Lecture Notes in Computer Science, J. Domingue and C. Anutariya, Eds. Springer Berlin / Heidelberg, 2008, vol. 5367, pp. 420–433.

[21] W. Dakka, "Augmenting Wikipedia with named entity tags," in *Proceedings of the 3rd International Joint ...*, 2008.

[22] X. Han and J. Zhao, "Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09," in *Proceeding of the 18th ACM conference*. New York, New York, USA: ACM Press, 2009, p. 215.

[23] M. Buckland, A. Chen, F. Gey, R. Larson, and e. al, "Geographic search: catalogs, gazetteers, and maps," *College & Research Libraries*, 2007.

[24] A. Kittur, E. H. Chi, and B. Suh, "What's in Wikipedia?: mapping topics and conflict using socially annotated category structure," in *Proceedings of the 27th international conference on Human factors in computing systems*, ser. CHI '09, 2009, pp. 1509–1512. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518930

[25] Unknown. (2011) Template:Persondata. [Online]. Available: http://en.wikipedia.org/wiki/Template:Persondata

[26] ——. (2012) Template:Taxobox. [Online]. Available: http://en.wikipedia.org/wiki/Template:Taxobox

[27] M. Wick. (n.d.) About GeoNames. [Online]. Available: http://www.geonames.org/about.html

[28] ——. (n.d.) GeoNames feature codes. [Online]. Available: http://www.geonames.org/export/codes.html

[29] G. Aisch. (2011) Rendering SVG country maps in Python. [Online]. Available: http://vis4.net/blog/posts/rendering_country_maps/

[30] Dispenser. (2009) Backlinkscount.py. [Online]. Available: http://toolserver.org/~dispenser/sources/backlinkscount.py

[31] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[32] Unknown. (2012) Old style and new style dates. [Online]. Available: http://en.wikipedia.org/wiki/Old_Style_and_New_Style_dates

[33] Codehaus. (2008) Jackson: High-performance JSON processor. [Online]. Available: http://jackson.codehaus.org/

[34] Loki. (2012) Java wiki bot framework. [Online]. Available: http://jwbf.sourceforge.net/

[35] M. Graham. (2009) Mapping the geographies of Wikipedia content. [Online]. Available: http://www.zerogeography.net/2009/11/mapping-geographies-of-wikipedia.html

[36] Kolossos. (2006) File:geonames4.png. [Online]. Available: http://en.wikipedia.org/wiki/File:Geonames4.png

[37] ——. (2012) [Map of geotagged Wikipedia articles]. [Online]. Available: http://toolserver.org/~kolossos/wp-world/imageworld-art-new.php?la=en&fa=4

[38] S. Pinker, *The Better Angels of Our Nature: Why Violence Has Declined.* Viking, 2011.

[39] Unknown. (2012) Wikipedia:WikiProject Geographical Coordinates. [Online]. Available: http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Geographical_coordinates

[40] M. S. Rajagopalan, "Patient-oriented cancer information on the internet: A comparison of Wikipedia and a professionally maintained database," *Journal of Oncology Practice*, 2011.

[41] Unknown. (2012) Simple English Wikipedia. [Online]. Available: http://simple.wikipedia.org/wiki/Main_Page

[42] C. Foss. (2012) Pleiades: Gangra/Germanicopolis. [Online]. Available: http://pleiades.stoa.org/places/844926

[43] M. Gertz. (2012) Temporal tagging. [Online]. Available: http://dbs.ifi.uni-heidelberg.de/index.php?id=129

[44] G. W. Fabian M. Suchanek, Gjergji Kasneci, "YAGO: A core of semantic knowledge unifying WordNet and Wikipedia," in *WWW 2007 / Track: Semantic Web*.

[45] Unknown. (2012) About: Andrea Gabrieli. [Online]. Available: http://dbpedia.org/page/Andrea_Gabrieli