Serverless typeahead over Linked Open Data with the TREE specification

Ruben Dedecker, Pieter Colpaert, Ruben Verborgh

IDLab, Department of Electronics and Information Systems, Ghent University – imec

HTML version at http://pieter.pm/icwe2020-demo-typeahead

Abstract. Typeahead is an essential feature for end-users to indicate, using a web-form, to which entity they are referring. It raises data quality by, early in the data life cycle, reconciling the creator's intention to the right identifiers. However, setting up a new typeahead today starts by setting up a new server on which the necessary data must be replicated and indexed. We introduce the idea of serverless data to indicate that the front-end can customize and code their own typeahead functionality without setting up a server. We propose that data publishers of individual base registries publish their data in HTTP documents. When a dataset is too large for one HTTP document, we propose adding a contextual hypermedia link with the TREE specification. In a demonstrator, we made an implementation of such a fragmented dataset for addresses in Flanders and provide a serverless typeahead on top of it. The demo shows no regressions in user experience, and adds new opportunities towards customization.

1. Introduction

Picking the right identifier for an item in a collection based on a description of that item is a fundamental functionality required when building a well-linked knowledge graph. One such method of *reconciliation* is typeahead [1]. This is a user interface feature to select an item from a large collection of entities based on a prefix. Examples of typeahead tasks on the Web of data include completing addresses, filling out points of interests such as public transport stops, or finding Wikidata entities.
A well-established design for typeahead services is an API that filters a full collection of items on a prefix on the server, as illustrated with this URL template: `https://example.org/{?q}`. However, either this requires a data publisher to expose server functionality to all its reusers, either it requires a data reuser to set up a server of their own. In this demo paper, we study whether a data publisher can still publish their data as if it were static files while providing a typeahead. The TREE specification is used to design a hypermedia search tree of documents. At the same time, the data consumer is able to do all remaining processing on the client. We set up our demo of street names in the region of Flanders with a dataset with 73k entities. With this demo paper, you can tell for yourself whether this provides for an acceptable user-perceived performance.

2. State of the art

Open source software such as Elastic Search and triple stores such as Virtuoso (`bif:contains`) or Blazegraph (`bds:search`), offer full server-side full-text search querying. This can be used for a typeahead (e.g., as is the case in Linked Open Vocabularies [2]). Specialized autocompletion software stacks can also be found, such as Pelias, which extends Elastic Search for address autocompletion and geocoding. Particularly interesting is that Pelias also publishes their own user experience guidelines for autocompletion. These include: (i) throttling requests, (ii) taking into account possible out of order responses and (iii) using a pre-written client on the front-end if possible.
This paper is a demo of applying Linked Data Fragments to prefix search. The idea is to find in-between solutions between data dumps – which require high effort from a reuser – and query servers – which require high effort from the data publishers – by fragmenting a data dump and telling an autonomous agent [3] how to navigate through the fragments using hypermedia controls. The only work done on text-search within this area so far is by Van Herwegen et al [4]. The Triple Pattern Fragments interface [5] was extended with substring filtering on objects. While the knowledge graph itself was fragmented in pages, the server-side filter was added for querying the data in the same way as the set-up with a query server.

3. Static file-based hypermedia structure

In order to make typeahead possible in a serverless environment, clients need to understand how to prune their search space. Therefore, we use the TREE hypermedia specification that is able to describe prefix-based links. Using such hypermedia controls, we designed a B-tree inspired prefix-tree, depicted in Fig. 1. The source code to create such a B-tree structure is available at https://github.com/Dexagod/linked_data_tree

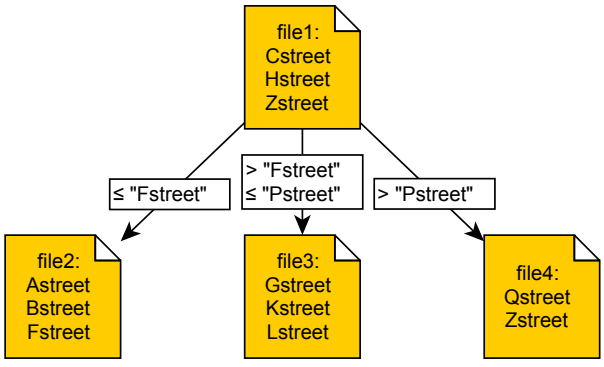

Fig. 1: B-tree structured index over files containing street names. When an autonomous agent is looking for Kstreet, it can understand from the typed links that only the middle fragment (and maybe the first fragment) may contain what it is looking for. The left and right fragments can be pruned from the agent's search space.

4. “Serverless” demonstrator

We coin the term “serverless data” for data that can be reused for querying purposes without the consumer having to set up their own server. Serverless typeahead is already possible over small data: one fragment of data would be downloaded and the filtering could happen entirely in the browser. We extended this idea to big datasets, by fragmenting these datasets in smaller chunks. To demonstrate such a *serverless* typeahead, we produced a tree structured index over the dataset of all addresses in Flanders. The files are statically served using a reverse proxy cache (nginx). Compression and cross origin resource sharing headers are enabled.
The demo can be found at http://193.190.127.164/paperdemo/ (will be moved to a more permanent location once accepted) and is depicted in Fig. 2.

Fig. 2: Demo application providing a typeahead over all street names in Flanders from the address registry of the Flemish government. Try typing “Rue du Culot”.

An interesting feature can be noticed when opening the query console of your browser when trying to type for example “Rue du Culot”, and afterwards correcting this to “Rue du Bois”. While making this correction has a Levenshtein distance of 5 only 1 extra HTTP request needed to be done. This is enabled by the ability of the client to keep application state when traversing the search tree. Furthermore, even when the page is reloaded and doing the same look-up, all HTTP requests are going to be served from cache.

5. Conclusion

The new client-server relation for prefix search has an effect on the user experience guidelines of Pelias (see). (i) Throttling requests became unnecessary. When altering the prefix for which no extra HTTP request would be needed, the throttling can be

very low. In this research we have tried to stay as close as possible as the state of the art, as judging from its adoption, this performance is widely accepted. In a similar way, there is also no danger of out of order responses. The client-side algorithm always interrupts the current prefix evaluation to emit responses, but forwards its on-going query processing to the next prefix. Last but not least, (iii) using a pre-written client was a guideline when working with the query server design, and remains. Now however, this pre-written client is given more responsibility, and with more responsibility also comes more flexibility to implement the typeahead feature just the way you want.

Yet, two other guidelines must be followed. (i) The most important of them all will be setting caching headers and enabling compression. Next, for public datasets, also (ii) Cross Origin Resource Sharing (CORS) headers need to be enabled. This will enable application developers to create *serverless* JavaScript applications, implementing the typeahead just the way they want. For example, when results that are nearby the user's current location need to be prioritized, that does not require extra server-side functionality: the client can implement this based on the same data publishing.

References

1. Maali, F., Cyganiak, R., Peristeras, V.: Re-using cool URIs: Entity reconciliation against LOD hubs. LDOW. (2011).
2. Vandenbussche, P.-Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semantic Web. 8, 437–452 (2017).
3. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a modular SPARQL query engine for the web. In: ISWC. pp. 239–255. Springer (2018).
4. Van Herwegen, J., De Vocht, L., Verborgh, R., Mannens, E., Van de Walle, R.: Substring Filtering for Low-Cost Linked Data Interfaces. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S. (eds.) Proceedings of the 14th ISWC. pp. 128–143. Springer (2015).
5. Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P.: Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. Journal of Web Semantics. 37–38, 184–206 (2016).