Auto-completion for Question Answering Systems at Bloomberg

Konstantine Arkoudas
Bloomberg
New York, USA
karkoudas@bloomberg.net

Mohamed Yahya
Bloomberg
London, United Kingdom
myahya6@bloomberg.net

ABSTRACT

The *Bloomberg Terminal* is the leading source of information and news in the finance industry. Through hundreds of functions that provide access to a vast wealth of structured and semi-structured data, the terminal is able to satisfy a wide range of information needs. Users can find what they need by constructing queries, plotting charts, creating alerts, and so on. Until recently, most queries to the terminal were constructed through dedicated GUIs. For instance, if users wanted to screen for technology companies that met certain criteria, they would specify the criteria by filling out a form via a sequence of interactions with GUI elements such as drop-down lists, checkboxes, radio and toggle buttons, etc. To facilitate information retrieval in the terminal, we are equipping it with the ability to understand and answer queries expressed in natural language. The following are examples of questions that our systems can already understand and answer:

- *What are the top 10 Asian tech companies with eps at least 4?*
- *Find corporate bonds rated A or better and with coupon higher than 7%.*
- *Show me news about oil from the Financial Times over the last two months.*

Our QA (question answering [1, 8]) systems map structurally complex questions like the above to a logical meaning representation which can then be translated to an executable query language (such as SQL or SPARQL). At that point we can execute the queries against a suitable back end, obtain the results, and present them to the users.

Adding a natural-language interface to a data repository introduces usability challenges of its own, chief amongst them being this: How can the user know what the system can and cannot understand and answer (without needing to undergo extensive training)? We can unpack this question into two separate parts:

(1) How can we convey the full range of the system's abilities?
(2) How can we convey its limitations?

We use *auto-complete* [4] as a tool to help meet both challenges.

Specifically, the first question pertains to the general issue of *discoverability*: We want at least some of the suggested completions to act as vehicles for discovering data and functionality of which users may have not been previously aware. The second question pertains to *expectation management*. Naturally, no QA system can attain perfect performance; limiting factors include representational shortcomings and various kinds of incompleteness of the underlying data sources, as well as NLP technology limitations. We want to stop generating completions as a signal indicating that we are not able to understand and/or answer what is being typed.

At minimum, then, we want our AC (auto-complete) systems to be both *sound* and *complete*. Soundness here means that if we do provide a completion q' for a given partial question (a question prefix) q, then q' is in fact both understandable (semantically parsable) and answerable. This being a conditional statement, soundness would be easy to achieve if we never provided any completions. Completeness is the needed dual: If q *can* in fact be extended to *some* semantically parsable and answerable question q' that is a suffix of q, then our AC system should provide completions for q.

These two properties are minimal desiderata because we want the list of completions to attain a number of additional properties:

(1) The top completion should be *predictive* of the user's intended query. After all, saving keystrokes was and remains the main objective of all auto-complete systems.
(2) The completion list should be *diverse*. For instance, if we are dealing with a QA system for news and a user types the letter *i*, then our completions should not be limited to companies whose names start with an *i* (such as IBM); it should include people (such as Icahn), sectors (such as insurance), countries (Ireland), and so on. Diversity is key for discoverability.
(3) The completions should be *propositional*, which is to say that they should have full propositional semantics: They should be full clauses that map to true or false. For example, considering the first sample question above, if the user typed everything up to ... *companies with e*, then ... *companies with eps at least 4* is an acceptable propositional completion but ... *companies with eps* is not. The former is a complete thought, so to speak, a query that can be run against a data back end; the latter is not.
(4) The completions should be *grammatical*. While the QA system may be able to understand queries formulated in a telegraphic style [5], we want to ensure that the completions we provide are linguistically well-formed. By suggesting well-formed questions we ensure that the meaning of the completions is unambiguous to the user, and we encourage users to move away from less expressive telegraphic queries to more expressive natural language questions where appropriate for their information needs.

These are the most important constraints we need to satisfy, but there are others, such as personalization and popularity [9]. The former means that the history and profile of a user should have an impact on the completions they receive. The latter means that popular queries (posed by large numbers of other users) should make for more likely completions.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5657-2/18/07.
https://doi.org/10.1145/3209978.3210204

Note that our criteria are markedly different from—and more challenging than—those of more conventional AC systems in the setting of more traditional keyword-based search [2, 7]. Those systems are based almost entirely on large logs of preexisting user queries, so query popularity is the main driving factor that determines what completions are generated, in tandem with some degree of personalization. The existence of large query logs not only shapes the problem, but also solves it to a large extent. With such logs at hand, it is not particularly challenging to build efficient prefix-based data structures and algorithms for selecting popular queries that complete a given input.

By contrast, auto-complete in our setting is characterized by a *cold-start* problem. We want to build and deploy the QA and AC systems simultaneously, which means that we have to build both from scratch and without any preexisting logs of user queries. This problem has been explored for keyword querying [3, 6], but in the setting of QA the presence of semantics combined with the above constraints introduces new challenges. These problems become more acute when we deal with infinite sets like numbers and dates, which are particularly important in the financial domain.

As the IR community has seen increasing interest in QA and chatbots, our adaption of AC to improve the usability of our QA systems, and the concrete industrial-strength solutions and insights we have developed, will be of wide interest. Our solutions rely on a plethora of approaches. We touch on two approaches here and elaborate on these and others in our presentation. In the first, we exploit our QA framework for parsing questions into a structured meaning representation as a language generator in order to do on-the-fly question generation for AC. The second approach exploits natural language compositionality to mix and match completions from any meager question logs we may have from internal annotators, or shortly after a QA system is deployed, thereby multiplying their effective size and coverage several-fold. In both cases semantics plays a key role across the board, from enabling smart deduping (based on meaning rather than surface form) to ensuring diversity. We make use of various machine learning models, statistics, and metadata to ensure morphological, grammatic, and semantic coherence of completion suffixes with the typed prefixes, as well as the diversity of the completion lists. Finally, we ensure that our top-level AC component meets stringent performance requirements: Completions must be generated with each new keystroke and typically in no more than 60 to 80 milliseconds (including time for spelling correction).

KEYWORDS

Auto-completion; question answering; auto-complete

ACM Reference Format:
Konstantine Arkoudas and Mohamed Yahya. 2018. Auto-completion for Question Answering Systems at Bloomberg. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3209978.3210204

1 ABOUT THE PRESENTERS

Konstantine Arkoudas is a senior research scientist and software architect at Bloomberg, where he leads the question answering project. Before joining Bloomberg, Konstantine was a senior research scientist at Telcordia Research, where he worked on access control, cognitive radios, data privacy, question answering in large RDF knowledge bases, and on efficient implementations of relational machine learning and AI planning. He holds a PhD in computer science from MIT and has published dozens of papers.

Mohamed Yahya is a senior software engineer at Bloomberg, where he works on question answering and auto-completion. Before joining Bloomberg, Mohamed was a doctoral student at the Max-Planck Institute for Informatics in Germany, where he worked at the intersection of IR, NLP, and DB systems. His main focus was question answering through semantic parsing over structured knowledge sources, overcoming incompleteness in structured sources by combining them with textual ones, query processing in this hybrid setting, and adding humans to the question answering loop. His work on question answering won an honorable mention at CIKM 2013. Previously, Mohamed worked on question answering at Siemens Corporate Research and Technology and on information extraction at Google Research.

2 ABOUT THE COMPANY

Bloomberg is the world's leading provider of financial insights, data, news and information since 1981. At the core of the company is the Bloomberg Professional Service (The Terminal), which provides real time financial information to approximately 325,000 subscribers globally. Bloomberg's enterprise solutions build on the company's core strength, leveraging technology to allow customers to access, integrate, distribute and manage data and information across organizations more efficiently and effectively. At its heart, Bloomberg is a technology company, with more than a quarter of its 19,000 employees working in engineering. As an information provider, Bloomberg has a strong focus on data science, with focus on IR, NLP, and data mining, and contributes back to the community through projects like Solr, Hadoop, and IPython/Jupyter.

REFERENCES

[1] Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on Freebase. In *CIKM 2015*. 299–304.
[2] H. Bast and Ingmar Weber. 2006. Type less, find more: fast autocompletion search with a succinct index. In *SIGIR 2006*. 364–371.
[3] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *SIGIR 2011*. 795–804.
[4] Fei Cai and Maarten de Rijke. 2016. A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval* 10, 4 (2016), 273–363.
[5] Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP 2014*. 1104–1114.
[6] Bhaskar Mitra and Nick Craswell. 2015. Query auto-completion for rare prefixes. In *CIKM 2015*. 1755–1758.
[7] Bhaskar Mitra, Milad Shokouhi, Filip Radlinski, and Katja Hofmann. 2014. On user interactions with query auto-completion. In *SIGIR 2014*. 1055–1058.
[8] Denis Savenkov and Eugene Agichtein. 2017. EviNets: Neural networks for combining evidence signals for factoid question answering. In *ACL 2017, Volume 2: Short Papers*. 299–304.
[9] Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *SIGIR 2013*. 103–112.