PeARS: a Peer-to-peer Agent for Reciprocated Search

Aurélie Herbelot
University of Trento, Centre for Mind/Brain Sciences
Palazzo Fedrigotti, Corso Bettini 31
38068 Rovereto, Italy
aurelie.herbelot@cantab.net

ABSTRACT

This paper presents PeARS (Peer-to-peer Agent for Reciprocated Search), an algorithm for distributed Web search that emulates the offline behaviour of a human with an information need. Specifically, the algorithm models the process of 'calling a friend', i.e. directing one's query to the knowledge holder most likely to answer it. The system allows network users to index and share part of their browsing history in a way that makes them 'experts' on some topics. A layer of distributional semantics agents then performs targeted information retrieval in response to user queries, in a fully automated fashion.

Keywords

Web search; distributional semantics; distributed systems

1. INTRODUCTION

The Web hosts billions of documents. Recording the content of those documents and providing the ability to search them in response to a specific information need within a mere couple of seconds is considered a challenging big data problem. In this paper, we propose that searching the Web does not necessarily require a large infrastructure. We redesign the notion of search as a distributed process mirroring the offline behaviour of a human agent with an information need and access to a community of knowledge holders.

Our implementation deals with the size of the search space in the way humans deal with the complexity of their environment: by focusing their attention on relevant sources of information. In our scenario, where representations of a large number of Web documents are spread out across a network of human users with specific browsing habits, queries are matched with those user records that are most likely to hold relevant information, thus considerably reducing the search space. Every semantic element in the network, from words to documents to user profiles, is modelled as a distributional semantics (DS) vector in a shared space.

2. INTUITION

Let's imagine a person with a particular information need, for instance, getting sightseeing tips for a holiday destination. Let's also imagine that this person does not have access to the Internet. How might she gain access to the information she seeks? Typically, she will identify the actors that may hold the answer to her query within her community: she might call a friend who has done the trip before, or her local travel agent. The *relevance criterion* used in this process ensures that she does not waste time talking to actors who are less likely to be able to help, such as the local baker or a poorly-travelled uncle.

PeARS reproduces this process by creating a distributed network layer over a community of real Internet users. Each peer in the network corresponds to a user and models that human's online behaviour – in particular their primary interests. A keen traveller is likely to know about the most informative travelling sites while a dog trainer may repeatedly visit the online shops which they consider reliable for buying training equipment. PeARS creates artificial agents – one per human user – which can query each other about the topics they are 'specialists' for (see §6 for details on respecting the user's privacy). These agents make decisions using *distributional semantics* models of both documents and users.

3. DISTRIBUTIONAL SEMANTICS (DS)

DS [2] is an approach to computational semantics actively researched within linguistics, cognitive science and neuroscience. A DS system analyses large corpora to build word meaning representations in the form of 'distributions'. In their most basic form, such distributions are vectors in a so-called *semantic space* where each dimension represents a term in the overall system vocabulary. The value of a vector along a particular dimension expresses how characteristic the dimension is for the word modelled by the vector (as calculated using e.g. Pointwise Mutual Information). It will be found, typically, that the vector *cat* has high weight along the dimension *meow* but low weight along *politics*. Actual implementations vary, from the basic setup described here to multi-modal, dimensionality-reduced models. DS relates to, but is distinct from, the use of vector spaces in classic information retrieval – in particular in its focus on cognitive plausibility.

One of the research areas in DS concerns compositionality, i.e., how words combine together to form phrases and sentences. It has repeatedly been found that simple addition of vectors performs well in modelling the meaning of larger constituents (i.e., we express the meaning of *black cat* by simply summing the vectors for *black* and *cat*). This paper expands on this result by positing that the (rough) meaning of a document is similarly the addition of all characteristic words for that document. Further, we can sum

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2889369 .

the distributions of the documents in a user's search history to get a single vector modelling that user. So in a single semantic space, we may model that *cat* and *dog* are similar, that two documents on classic cars belong to the same topic, and that two users who browse programming forums may have relevant information for each other (even if they do not necessarily browse the same sites).

4. SYSTEM ARCHITECTURE

A PeARS network consists of n peers $\{p_1...p_n\}$, corresponding to n users $\{u_1...u_n\}$ connected in a distributed typology (all peers are connected to all other peers). Each peer p_k has two components: a) an indexing component I_k; b) a query component Q_k. All peers also share a common semantic space $\mathcal{S}$ which gives DS representations for words in the system's vocabulary. In our current implementation, $\mathcal{S}$ is given by the CBOW semantic space of [1], a 400-dimension vector space of 300,000 lexical items built using a state-of-the-art neural network language model.

Indexing: I_k builds vector representations for each document in u_k's browsing history. For instance, if u_k visits the Wikipedia page on Bangalore, the URL of that page becomes associated with a 400-dimension vector produced by summing the distributions of the 10 most characteristic words for the document (these are identified by comparing their document frequency to their entropy in a large corpus). At regular interval, I_k also updates u_k's profile by summing the vectors of all indexed documents, outputting a 400-dimension vector $\vec{u_k}$ which inhabits an area of the semantic space related to their interests (i.e., the type of information they have browsed).

As a result of the indexing process, two types of information are made freely available across the network: the user profile $\vec{u_k}$ and the individual document vectors $D_k = \{d_1...d_n\}$ used to build $\vec{u_k}$ (at a particular URI, or in the form of a distributed hash table). Periodically, each peer $p_1...p_n$ scans the network to collect all profiles $\vec{u_1}...\vec{u_n}$ and stores them locally.

Querying: Q_k takes a query q and translates it into a vector $\vec{q}$ by summing the words in the query. It then goes through a 2-stage process: 1) find relevant peers amongst $p_1...p_n$ by calculating the distance between $\vec{q}$ and all users' profiles $\vec{u_1}...\vec{u_n}$ (vector distance is operationalised as cosine similarity); 2) on the m relevant peers, calculate the distance between $\vec{q}$ and all documents indexed by the peer. Return the URLs corresponding to the smallest distances, in sorted order.

5. PERFORMANCE

Speed: Q_k involves two stages: 1) the computation of cosine similarities between a query (one vector with 400 dimensions) and all the peers on the network (a matrix with dimensionality $n \times 400$); 2) calculating cosine between the query and the documents hosted by the most relevant peers, as identified in the first stage. For the purpose of assessing system speed, we generate random vectors and perform cosine over the generated set. Our current implementation, running on a 4GB Ubuntu laptop under normal load, performs the calculation over batches of n=100,000 peers at stage 1. Each batch is computed in around 350ms. At stage 2, assuming an average of 10,000 documents per node, the computation time is 45ms for each peer.

This preliminary investigation indicates that on a home machine, the system covers up to 200,000 peers × 10,000 = 2 billion documents in around a second (we must subtract potential redundancies between peers from this figure). Note that in a 'real-life' system, we would need to include additional time to retrieve the indices of the remote peers. However, we can also increase efficiency by sorting the list of known peers as a function of their similarity to the user's profile and caching the most similar nodes. The premise is that a user will very often search for information related to their interests and thus require access to peers that are like their own profile.

Accuracy: Measuring the search accuracy of the system is ongoing work. We are testing the system's architecture on real user queries from the search engine Bing, as available – together with the Wikipedia page users found relevant for the respective queries – from the WikiQA corpus [3]. Our current simulation is a network of around 4000 peers covering 1M documents, modelled after the publicly available profiles of Wikipedia contributors. Preliminary results indicate that our system, when consulting the $m = 5$ most promising peers for each query, outperforms a centralised solution, as implemented by the *Apache Lucene* search engine[1] (Herbelot & QasemiZadeh, in prep.).

6. CONCLUSION

We have presented an architecture for a user-centric, distributed Web search algorithm that utilises the inherent 'specialisms' of individuals as they browse the Internet.

We should note that our system relies on the willingness of its users to share some of their search history with others. We alleviate the privacy concerns associated with this requirement in three ways: a) the user can create a blacklist of sites that will never be indexed by the system; b) before making an index available, the agent clusters documents into labelled topics and presents them to the user, who can decide to exclude certain topics from the index; c) there is no requirement for the shared index to be linked to a named and known user.

PeARS is under active development and code is regularly made available at `https://github.com/PeARSearch`.

7. ACKNOWLEDGMENTS

Grateful thanks to Hrishikesh K.B., Veesa Norman, Shobha Tyagi, Nandaja Varma and Behrang QasemiZadeh for their technical contributions to the project, and to the anonymous reviewers for their helpful feedback. The author acknowledges support from the ERC Grant 283554 (COMPOSES).

8. REFERENCES

[1] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL2014*, pages 238–247, 2014.

[2] Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.

[3] Yi Yang, Wen-tau Yih, and Christopher Meek. WIKIQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP2015*, 2015.

[1] `http://lucene.apache.org/`