Chapter XXXIII

Question Answering Using Word Associations

Ganesh Ramakrishnan
IBM India Research Labs, India

Pushpak Bhattacharyya
IIT Bombay, India

ABSTRACT

Text mining systems such as categorizers and query retrievers of the first generation were largely hinged on word level statistics and provided a wonderful first-cut approach. However, systems based on simple word-level statistics quickly saturate in performance, despite the best data mining and machine learning algorithms. This problem can be traced to the fact that, typically, naive, word-based feature representations are used in text applications, which prove insufficient in bridging two types of chasms within and across documents, viz., lexical chasm and syntactic chasm. The latest wave in text mining technology has been marked by research that will make extraction of subtleties from the underlying meaning of text, a possibility. In the following two chapters, we pose the problem of underlying meaning extraction from text documents, coupled with world knowledge, as a problem of bridging the chasms by exploiting associations between entities. The entities are words or word collocations from documents. We utilize two types of entity associations, viz., paradigmatic (PA) and syntagmatic (SA). We present first-tier algorithms that use these two word associations in bridging the syntactic and lexical chasms. We also propose second-tier algorithms in two sample applications, viz., question answering and text classification which use the first-tier algorithms. Our contribution lies in the specific methods we introduce for exploiting entity association information present in WordNet, dictionaries, corpora and parse trees for improved performance in text mining applications.
1. INTRODUCTION

A QA system responds to queries like Who is the Greek God of the sea? with a precise answer like Poseidon. A slightly less ambitious goal is to identify short snippets or passages of up to several words which contain the answer.

QA has roots in classic AI-style inference engines, but in this work we focus on recent open-domain systems closer to the Information Retrieval (IR) community. Falcon Harabagiu et al., 2000a), Weblopedia (Hovy et al., 2000), AnswerBus (Zheng, 2002) and AskMSR (Dumais et al., 2002) are some well-known research systems, as are those built at the University of Waterloo (Clarke et al., 2000; Clarke et al., 2001b), and Ask Jeeves (http://ask.com).

Most QA systems are substantial team efforts, involving the design and maintenance of question taxonomies, question classifiers, and passage scoring heuristics. The intensity of human effort involved has limited state-of-the-art QA system development to a handful of groups. The current lack of clear separation between algorithms and knowledge bases makes it hard to gauge the benefits of new algorithmic ideas, and to generalize the tuning experience to new domains, new corpora, and new languages.

The description of a QA system is almost exclusively about how questions and passages are processed, how they are matched and scored, and how external knowledge bases (question taxonomy, dictionary, and thesauri) are exploited. Even if some strategies make intuitive sense, the treatment is predominantly operational (how) rather than declarative (what): it is rare to find a system-independent discussion of general properties that make one passage better answer a question than another.

Compare the QA situation with IR engines, which are available off the shelf (e.g., Lucene (Group, 2002)), require essentially no tuning, and can be deployed in minutes. The basics of the vector space model and tfidf ranking can be taught in an hour. In contrast, QA systems contain large pieces of software, lashed together in diverse ways with customized “glue” and many crucial knobs to turn. Naturally, these knobs are best turned by QA specialists rather than the end-user, which might explain in part why off-the-shelf QA packages are rare.

The broad architecture of QA systems (Clarke et al., 2000; Harabagiu et al., 2000a; Hovy et al., 2000; Zheng, 2002; Clarke et al., 2001b; Radev et al., 2002) has become standard. The corpus is indexed at the level of documents or passages chopped-up at a suitable size. A shallow entity extractor, supported by a large gazette, is sometimes run on the passages to identify people, places, organizations, etc. (Abney et al., 2000; Dill et al., 2003), which may also be indexed. A taxonomy of question types (where, when, who, how many, etc.) is built by hand, and rules tuned to map questions to types. Several QA systems assume that an answer type catalog is available. And if it is not already available, they build such catalogs with great care (Harabagiu et al., 2000b; Hovy et al., 2000) and classify each question into an answer type. Accordingly, the question is transformed into a keyword query to be submitted to the index. Responding passages are re-ranked using a variety of strategies.

The methodology of Agichtein et al. (Agichtein et al., 2001) is to learn from QA pairs how to transform a question into words and short phrases likely to appear in an answer. Frequent item-set and co-occurrence detection is at the core of their system. A related paper (Lam et al., 2003) explores how to pick one answer from a small set of candidate answers, as in the millionaire TV show.

Another well-known Web QA system by Dumais et al. (Dumais et al., 2002) use a seven-way classification of the question followed by a customized rewrite for each class. The transformed query is forwarded to Web search engines and the redundancy in responses help in ranking passages. Apart from query transformation, Dumais et al. point out the great value of proximity queries involving query terms