Roussanka Loukanova · Peter LeFanu Lumsdaine ·
Reinhard Muskens
Editors

Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)

Springer

Editors
Roussanka Loukanova
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria

Reinhard Muskens
Faculty of Science
ILLC—Institute for Logic, Language
and Computation
University of Amsterdam
Amsterdam, The Netherlands

Peter LeFanu Lumsdaine
Department of Mathematics
Faculty of Science
Stockholm University
Stockholm, Sweden

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-031-21779-1 ISBN 978-3-031-21780-7 (eBook)
https://doi.org/10.1007/978-3-031-21780-7

White Roses, Red Backgrounds: Bringing Structured Representations to Search

Tracy Holloway King

Abstract Search has become a key component of many on-line user experiences. Search queries are usually textual and hence should benefit from improvements in natural language processing. However, many of the NLP algorithms used in production systems fail for queries that require structured understanding of the query and document or that require reasoning. These issues arise because of the way information is stored in the search index and the need to return results quickly. The issues are exacerbated when searching over non-textual documents, including images and structured data. The use of embedding-based techniques has helped with some types of searches, especially when the query vocabulary does not match that of the documents and when searching over images. However, these techniques still fail for many searches, especially ones requiring reasoning. Simply combining classic word-level search and embedding-based search does not solve these issues. Instead, in this position paper, I argue that we need to create hybrid systems from traditional search techniques, embedding-based search, and the addition of structured data and reasoning. Enabling such hybrid systems will require a deep understanding of linguistic representations of meaning, of information retrieval optimization, and of the types of information encoded in the queries and documents. It is my hope that this paper inspires further collaboration across disciplines to improve these complex search problems.

Keywords Search · Information retrieval · Semantic search · Query understanding

1 Introduction

Search has become a key component of many on-line user experiences. These include large web search engines (e.g., Google, Bing, Baidu, Yandex), eCommerce (e.g., Amazon, eBay, Rakuten, Taobao), and search within websites and within documents.

T. H. King (✉)
Adobe Inc., San Jose, CA, USA
e-mail: tking@adobe.com

R. Loukanova et al. (eds.), *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, Studies in Computational Intelligence 1081,
https://doi.org/10.1007/978-3-031-21780-7_9

Search queries are usually textual and hence should benefit from natural language processing (NLP) and especially from the rapid improvements in NLP over the past decades. This is especially true in situations where there is not enough behavioral data from past users to "memorize" the top search results and so a deeper understanding of the query and documents is required.

However, many of the NLP algorithms currently used in production systems fail for queries that require a structured, whether syntactic or logical, understanding of the query and document, or that require reasoning. Examples of such queries include *dresses between $50 and $100* (eCommerce search), *cake recipes without wheat flour* (web search and recipe site-internal search), and *photo white rose red background* (image search). Even simple queries that only require an understanding of word-order based relations such as *chocolate milk* vs. *milk chocolate* often include many irrelevant search results.

These issues arise because of the way information is stored in the search index, often as single words for text-based documents, and the need for extremely fast computation in order to return search results quickly to the user. The issues are exacerbated when searching over non-textual documents, including images and structured data (e.g., prices for eCommerce products). The use of embedding-based techniques has helped with some types of searches, including when the query vocabulary does not match that of the documents (e.g., misspellings, synonyms) and when searching over images. However, these techniques still fail for many searches, especially ones requiring reasoning. Simply combining classic word-level search and embedding-based search does not solve these issues. There remain classes of queries which require information beyond the representation of words and multi-word expressions stored in an inverted index. These are the structured representations referred to in the title of this paper. They involve associating typed information with the content (e.g., prices) and storing relationships among the entities (e.g., in an image the rose is white, the background is red). These structured representations in turn must support reasoning (see the *snacks without nuts* example query below). Although some of this reasoning can be computed off-line and stored for commonly queried information, to support the broad range of search queries, fast and accurate reasoning at query time is required.

Three Example Queries To understand the scope of the issue, consider three real-world example queries that require information beyond simple keyword matching.

First consider a search for images: *white rose red background*.[1] This query is looking for images of white roses shown on a red background. However, there are many more images of red roses than white roses because red is such a popular color for roses and there are a huge number of images with white backgrounds because these are used when compositing images. This means that there are many more images of red roses on white backgrounds than white roses on red backgrounds and that those images are much more popular (e.g., viewed, downloaded, or purchased

[1] There is no preposition in this query (cf. *white rose on red background*), but the search results are similar even with the preposition. The preposition helps to delineate the two noun phrases, but the techniques to improve the results with the preposition also help when the preposition is absent.

Fig. 1 Results for the query *white rose red background* with (upper) and without (lower) sufficient query and document understanding for search

more). If the words in the query are treated as a bag of words without reference to word order or syntactic structure, then any image associated with roses, backgrounds, and the colors red and white will be returned and the more popular images of red roses on white backgrounds will rank higher. Example search results for this query are shown in Fig. 1. One way to handle this query using structured representations will be discussed in detail in Sect. 4.3.

Next consider an eCommerce search with negation: *snacks without nuts*. The preposition *without* encodes a negation of containing nuts. This is a common query by users who do not like nuts or are allergic to them. However, when treated as a bag of words, the preposition *without* is either dropped entirely because it is so frequent as to be considered useless[2] or will match many items where the *without* applies to some other text in the product description. To make matters worse, the search engine will try to match the word *nuts* and so will return snacks which specifically contain nuts. As a result, the search results both miss relevant items (snacks without nuts that

[2] Such words are referred to as stopwords in search. See Sect. 2.2 on text processing.

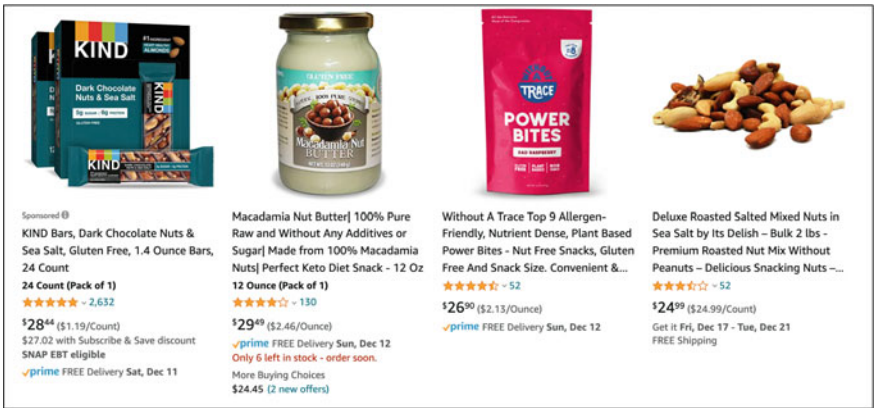

Fig. 2 Results for the query *snacks without nuts* where most of the results clearly contain nuts

did not mention the exclusion of nuts) and include many items with nuts (where the word *without* referred to something other than nuts). An example set of search results are shown in Fig. 2. The search results are much better when rephrasing the query as *nut free snacks* but even then many relevant results are missed because only snacks that overtly state they are nut free are returned. In search, checking for the absence of something (nuts in this example) is complicated because there are so many things that could be absent in a document that they cannot be listed: their absence has to be determined at search time.

Finally consider an eCommerce query that requires simple reasoning: *dresses over $100*. Here the price phrase requires the search engine to constrain the price of the returned items (the dresses) to be greater than $100. If the query is treated as a simple textual query, the dresses returned will either be exactly $100 if the dollar sign is searched for or, more likely, any price but containing the word *100* somewhere in the product description.

Given the frequency of price-related queries on eCommerce sites, many of sites have special query processing to detect the price constraint and map it to structured data on each item. In this case, the phrase *over $100* would be mapped to the price information and checked that the value is greater than $100. Example search results where this reasoning has been applied are shown in Fig. 3.

The remainder of this paper examines why search works as it does and how this is being improved. Section 2 provides the basics on how search works, focusing first on inverted indices, then introducing text processing, the use of user behavioral data, and structured data, and finally describing search ranking. Section 3 discusses why pure inverted indices are not sufficient. Section 4 introduces techniques to enhance traditional search to provide more accurate results and provides a detailed example. Finally, Sect. 5 concludes.

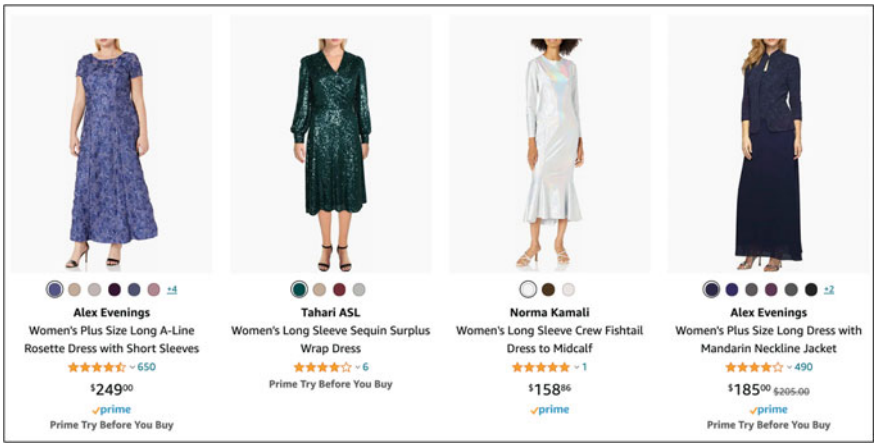

Fig. 3 Results for the query *dresses over $100* where the price constraint has been properly applied. The word *100* does not appear in the titles or prices

2 How Search Works

This section provides a high level overview of how search works, focusing on traditional inverted indices but also introducing other aspects of search used in production search engines. Those familiar with search can skip this section.

2.1 Inverted Indices

Most search engines are based on an inverted index. An inverted index is similar to the index at the end of a book. It allows you to look up a word or phrase and see what pages are associated with it. In search, these pages are documents (e.g., web pages in web search, product listings in eCommerce, documents in document collections like arXiv).

To create a search index for a set of documents, first the system identifies all the words associated with each document. Then it builds an index of the words and associates them with an identifier for relevant documents. Table 1 shows four simple one-sentence documents and the inverted index that is created from them.

When the user issues a query, the query is broken into words. All of the documents associated with each word are found by looking up the word in the index. The document lists for each word are compared. The documents that appear on all the lists are returned. This set of documents is referred to as the recall set or match set. These documents are then ranked so that the most relevant one is first, then the next more relevant, etc. Four sample queries and the documents returned for them are shown in Table 2.

Table 1 Sample inverted index based on 4 documents, each with only one sentence

Documents	
Doc. Id	Text
1	Cats are furry
2	Mice are furry
3	Dogs chase cats
4	Cats chase mice

Inverted index	
Word	Doc. Ids
cats	1, 3, 4
furry	1, 2
mice	2, 4
dogs	3
chase	3, 4

Table 2 Sample queries and the documents they return for the documents and inverted index in Table 1

Query	Initial documents	Final documents
cats	1, 3, 4	1, 4, 3
cats chase	cats: 1, 3, 4; chase: 3, 4	3, 4
cats birds	cats: 1, 3, 4; birds:—	—
mice dogs	mice: 2, 4; dogs: 3	—

The one-word query *cats* returns all the documents with the word *cats* in them. For the ranking, here we ranked the two documents with the word *cats* as the subject higher than the one with it as an object. For the query *cats chase*, there are two documents which contains both words and so those two are returned. We ranked the document which matches the word order in the query higher. In contrast, the query *cats birds* does not return any results because there is no document that contains the word *birds*. Similarly, the query *mice dogs* also returns no results. In this case, there are two documents which contain the word *mice* and one which contains the word *dogs*, but no documents that contain both. Many search engines return "partial matches" where not all the query words are matched if there are no documents that match all the words. This is not shown in Table 2.

2.2 Beyond Inverted Indices

Text Processing In the inverted index example above, the words were indexed in their inflected forms (e.g., plural *cats*). In search, the processing of the text in documents and queries strongly affects search result quality. In general, text is uniformly lower cased, even for proper nouns where it can be distinguishing. This is because queries do not contain canonical capitalization (e.g., English proper nouns are usually lower cased; users may have the caps-lock key on). Accent marks may be removed, mapping accented letters to their unaccented counterpart. Whether to de-accent depends on

the language and in particular on whether users reliably type queries with accent marks. Punctuation is stripped except when crucial for meaning (e.g., decimal points within numerals are not removed but sentence- and abbreviation-final periods are).

Stemming[3] or lemmatization is usually applied. Queries often include plural nouns even when the user is looking for a singular instance. For example, in eCommerce, the query *hammers* has more purchases for single hammers than for hammer sets or multiple hammers. The same is true for the query *dresses*: even though many people own multiple dresses, they purchase them one at a time.

Finally, words that are extremely common in the document collection are not indexed. These are referred to as stop words. The linguistically closed class words (e.g., prepositions, determiners) are usually stop words. These are not indexed because they are so frequent as to not discriminate between documents. However, as discussed in Sect. 3, they can be crucial for certain types of reasoning (e.g., *snacks without nuts*) and for media queries (e.g., bands like The Who and The The), which will not be discussed further here.

See [17] for a detailed overview of text processing for search.

Behavioral Data Most search engines have some queries that are much more popular than others. These popular queries are referred to as head queries, and the rare ones as tail queries. In web search and larger eCommerce sites [24], the search engine can effectively memorize the best results for head queries. Although this can be done by editorially providing results, it is usually learned from aggregated user behavior. That is, the search result that users click on the most is put in first position, the next most popular result in second position, etc. In web search, this can be seen with navigational queries where the query is looking for a particular popular site (e.g., the query *bbc*, where most users are looking for the BBC on-line news site). Even when a query is not frequent enough to memorize the top results, user behavioral data can provide signal as to which documents are most popular and which queries are associated with these popular documents.

Meta-data and Structured Data Most search engines involve documents which contain more than just text, or images in the case of image search. This data can provide valuable information for search, both for identifying relevant documents and for ranking them. Web pages have titles and urls (site addresses). In addition, web documents often include data about what queries led to clicks on them and what text was used to link to them from other documents (referred to as anchor text). Products in eCommerce search contain a large amount of structured data (e.g., price, brand, size) [15]. Similarly, publications contain author, year, and publisher information.

Our example query *dresses over $100* demonstrates the importance of structured data. Knowing that a query is restricting the price of the search results is not actionable unless the price is available in the documents in a way that the search engine can

[3] Stemmers remove the endings of words, leaving a stem that can be indexed. This stem may or may not be a word in the language [20]. Lemmatizers map inflected forms to a linguistic or dictionary base form. The difference between these is most obvious with highly irregular inflected forms where stemming cannot normalize a word to a citation form.

apply the restriction. In the case of prices, this means having them in a format that allows basic arithmetic operations to be performed. The prices are also used when ranking the products by price instead of relevance, an option that is available on most eCommerce sites.

2.3 Ranking Features

The set of documents retrieved has to be ranked for display to the user. The order of results is important for ensuring users have easy access to the most relevant results. Users rarely look beyond the first page of results and have a tendency to click on the top result. Here we mention some key factors in search ranking because even if we solve the core relevancy issues described in the introduction around queries like *white roses red background*, *snacks without nuts*, and *dresses over $100*, there is still significant work to be done to order those results to provide an optimal search experience.

Some ranking features focus only on the document, in particular on document popularity and reliability. For example, in web search, results from Wikipedia are often ranked highly for entity queries like proper names (e.g., *tbilisi*, *ruth bader ginsburg*). Other ranking features involve query-document affiliation. For example, where the query word matches within the document matters: documents which match the query words in the title are ranked more highly than ones which match lower in the document. Relatedly, if the query contains multiple words, then documents where the two words are near to each other are ranked more highly. A special subcase of this is multi-word expressions (MWE) where the words should be a MWE both in the query and the document. There is a strong preference in the ranking for MWE to be adjacent in the document text. High-frequency MWE may be indexed as single words (e.g., *san francisco*, *hot dog*), i.e., they are treated as words with spaces for purposes of the index.

In this paper, we focus on the match set since the integration of structured representations in search is fundamental for retrieving all and only documents that match the intent of the user's query. For more on ranking see [7, 17, 24].

2.4 An Example: Milk Chocolate *Versus* Chocolate Milk

This section walks through an example of how search works, moving beyond the simplistic furry cat example in Sect. 2.1. Consider the queries *milk chocolate* vs. *chocolate milk*. Assume that neither *milk chocolate* nor *chocolate milk* are MWE in the search index.

For the document processing, lower-casing allows us to match titles and section headers which contain initial or all capital letters (e.g., *Berkeley Farms Chocolate Milk*). Lemmatization allows us to match documents with plural words like

assortment of milk chocolates. After the text normalization, documents that mention the normalized form of *chocolate* or *milk* will have entries in the inverted index for those words.

When the query is issued, it is lower-cased, lemmatized, and stop words are removed in exactly the same way that the documents were. Stop words are not an issue here since neither query contains one. Lemmatization maps *milk chocolate* and *chocolate milk* to themselves since none of the words are inflected. Lower-casing maps any capitalization in the query to the lower-case forms used in the index.

Using a standard inverted index and no MWE for these words, the two queries will include precisely the same documents in the match set. This is because although the word order differs, they contain the same words.

However, ranking will treat the queries differently and rank different documents highly. In both web and eCommerce search, these queries are likely to have been seen before and have user click data associated with them. So, documents that were clicked for those queries by other users will be ranked more highly. In addition, word proximity, including word order, is key to distinguishing the queries: documents with the exact query phrase in the document are much more likely to be relevant and should be ranked higher. A Google search for the queries demonstrates the strength of the phrase matching and user behavioral data. At least in the US (google.com) shopping results are shown at the very top of the page; the Wikipedia article with the exact phrase title is shown as the first result; there is an entity box with a definition of chocolate milk (for *chocolate milk*) and milk chocolate (for *milk chocolate*) as well as additional entity-related information (e.g., wine pairings for *milk chocolate* and serving size for *chocolate milk*).

3 Why Inverted Indices Are Not Enough

In Sects. 1 and 2, we alluded to the fact that inverted indices are not enough to ensure relevant search results for all queries. Even with basic text processing, many relevant documents can be missed and many irrelevant documents can be included. This section describes how these issues arise and some ways to address them. As we will see, even these capabilities are not enough to provide relevant results for many types of queries. Those will be addressed in Sect. 4.

3.1 Vocabulary Mismatches

User queries may contain synonyms and paraphrases of the words used in the documents. These documents should be returned in the search results since they are relevant to the query: they just use different words with the same semantic meaning. For example, there is a piece of furniture that can be referred to in English as a *sofa*, *couch*, or *chesterfield*. In an eCommerce scenario, if a query contains any of these

words, all the relevant products should be returned, regardless of which word appears in the product title and what the name of the category that contains the products is.[4]

There are two ways to handle synonyms in inverted indices. The first is to normalize to a specific form (e.g., mapping *couch* and *chesterfield* to *sofa*). This is similar to using lemmatization to normalize singular and plural forms of nouns. Normalization applies to both the index and the query so that they can match. Unlike lemmatization, synonym normalization may add additional forms instead of replacing the form. This allows the ranker to use information about exact form matching in addition to retrieving all the documents via the normalization. When using normalization while maintaining the original form, the query treats the two forms as alternatives. For example, the query *leather couch* would match documents with either the words *leather* and *couch* or with the words *leather* and *sofa*. Table 3 shows the two treatments of normalization. Misspellings are always handled as normalization.

The other way to handle synonyms in inverted indices is via expansion. In expansion, a word is expanded to all of its synonyms, as opposed to normalization where one of the synonyms is the form that all the words map to. Expansion can occur either in the index or the query. If done in the index (Table 4), whenever one of the words is found, all the synonyms are indexed. Expanding in the index has two advantages. First, more context is available. This means the system is more certain that the word should be expanded (e.g., it is not some other meaning of the word which would not have the synonym expansion). Second, the search is faster because the number of words being searched from the query is smaller. If done in the query (Table 5), whenever one of the words is found, all of the synonyms are looked for as alternatives. Query expansion has the advantage that if there is a problem with a synonym it can be quickly fixed because there is no need to reindex the documents.

3.2 Mapping to Structured Data

Many documents contain structured data. Sometimes the structured data is part of the original document (e.g., prices on products in eCommerce, date stamps on papers in document collections). In other cases, the structured data is created as part of the document processing. For example, a named entity detector can extract all the

[4] There are two interesting variants of the synonym problem. The first is misspellings, which are a different, albeit technically incorrect, way to refer to a word. Search engines usually employ a separate speller module to correct misspellings in queries [10]. Given how short queries are, spell correction can be difficult, especially in the broad domain of web search. In addition, the autocomplete (also referred to as type-ahead or query suggestion) [8] feature provided in many search engines helps to guide users to properly spelled queries. In domains where there are likely to be misspellings in the documents, the speller may also be applied during document indexing. The second variant of the synonym problem is providing cross-lingual search, i.e., having a search query in one language find documents in another language [19]. In this case the synonyms are the words in the different languages with the same meaning. Cross-lingual search occurs for certain document collections, especially domain-specific archives, and when searching for images, where the images may be tagged in one language, usually English, but searched for in many languages.

Table 3 Normalization approach to synonyms: Documents with any of *sofa*, *couch*, or *chesterfield* will index *sofa* (indexing of the words *brown* and *leather* is not shown). Queries with any of *sofa*, *couch*, or *chesterfield* will search for *sofa*. The pipe (|) indicates an alternative (logical OR). All three documents will be returned for all three queries. The original words can be used in ranking

Document synonym normalization			
Document text	Original word	Indexed words	
		Normalized only	Normalized+original
Brown Leather Sofa	Sofa	sofa	sofa
Brown Leather Couch	Couch	sofa	sofa, couch
Brown Leather Chesterfield	Chesterfield	sofa	sofa, chesterfield
Query synonym normalization			
Original query	Normalized query		
	Normalized-only query	Normalized+original query	
sofa	sofa	sofa	
couch	sofa	sofa|couch	
chesterfield	sofa	sofa|chesterfield	

Table 4 Document expansion approach to synonyms: Documents with any of *sofa*, *couch*, or *chesterfield* will index all three forms (indexing of the words *brown* and *leather* are not shown). All three documents will be returned for queries with any of those words in them

Document synonym expansion		
Document text	Original word	Indexed words
Brown Leather Sofa	Sofa	sofa, couch, chesterfield
Brown Leather Couch	Couch	sofa, couch, chesterfield
Brown Leather Chesterfield	Chesterfield	sofa, couch, chesterfield

Table 5 Query expansion approach to synonyms: Queries with any of *sofa*, *couch*, or *chesterfield* will search all three words. The pipe (|) indicates an alternative (logical OR). So, documents with any of the three synonyms will be returned for all three queries

Query Synonym Expansion	
Original query	Expanded query
sofa	sofa|couch|chesterfield
couch	sofa|couch|chesterfield
chesterfield	sofa|couch|chesterfield

people, locations, and organizations mentioned in a document and provide canonical forms for these and even links to a knowledge graph nodes for the entities. Failing to recognize entities and treat them as such can result in irrelevant search results. For example, a search for my name *tracy king* on Amazon returns many books where either the author's first or last name is Tracy and the title of the book contains the word *king*. If you include my middle name *tracy holloway king*, more of the results are relevant, but many are not. Clearly I need to work on my popularity as an author in order to rank more highly.

These entities can be indexed as special fields so that the fact that they are entities and the type of entity are recorded. This is similar to how words in the document title are indexed distinctly from those in the rest of the document so that ranking can put more weight on matching words in the title. These fields may be text fields that allow for standard inverted index retrieval (e.g., brand names in eCommerce, author names for books and documents). Some entities are not text and so are stored and searched differently, generally in ways similar to databases. Examples of such entities include prices in eCommerce and dates for documents. By having these in entity-specific formats, it is possible to reason over them, as required for queries like *dresses over $100* or *19th century poems*.

3.3 Negation and Syntactic Structure

The last category of issues affecting the quality of search results from inverted indices are for queries where finding relevant results requires an understanding of syntactic or semantic structure.

In the realm of semantics, negation is particularly complex for search. Search with inverted indices works by finding documents which contain combinations of specific words. Determining whether a document does not contain a word, much less the semantic concept that corresponds to the word in the query is much less efficient. In the *snacks without nuts* example (Fig. 2), search has to interpret this as a query for the ingredients of the snack not including nuts. Simply excluding documents without the word *nut* can exclude relevant documents: A document might contain the word *nut* if it is in a phrase like *nut free*, *no nuts*, or even *my kids are nuts about this snack* in a review of the product. If the product contains a special ingredients field, then the negation can take scope only over that field. However, checking that the word *nut* does not occur in the ingredients field is not enough since different types of nuts also should not appear there (e.g., words like *almonds*, *walnuts*, or *pecans* also have to be absent).[5] Creating the product data, query understanding [1], and search facets (e.g., left rail filters for various attributes) to handle negation requires detailed domain knowledge and systems. As a result, negation is currently rarely addressed systematically in search engines.

[5] See [5, 6] on entailment and contradiction detection more generally and [14] on hybridizing natural language inference systems.

As opposed to the semantics required for negation, information from syntactic structure is easier to capture in search [25, 28, 29]. Modifier-head relationships, such as those in the *chocolate milk* vs. *milk chocolate* example (Sect. 2.4) are of this type. If these occur frequently enough in queries, they can be treated as MWE even if they are compositional or the modification structure can be stored as additional data in the index. When processing the documents to find the words and structures to index, simply scanning for the particular phrase (referred to as an ngram) is not always enough. The ngram may occur in contexts where it does not refer to the modifier-head relationship. In addition, this simple string adjacency will miss instances where the modifier is not strictly adjacent to the head (e.g., *milk and dark chocolate assortment* or *plain, chocolate, and strawberry milk*) [22]. More complex syntactic analysis is needed to find documents which answer queries such as *who acquired PeopleSoft* versus *who did PeopleSoft acquire*. The words *PeopleSoft* and *acquire* (and its inflected forms and possibly words like *acquisition*) will match the same documents. The ranking can treat these differently using word order, but syntactic understanding is necessary as search comes closer to being question answering and not just document retrieval.

3.4 Robustness Through Embeddings

Several of the techniques discussed above involve improving recall, i.e., improving the robustness of the system to mismatches between user queries and documents. Text normalization and synonyms are of this type. In addition to the methods described above to handle these, with the advent of scalable deep-learning (DL) systems, instead of using traditional word-based inverted indices, some search engines integrate embedding-based search. DL models map text and images to arrays of numbers, referred to as embeddings, in an abstract semantic space. Words that have identical or similar meanings (e.g., *sofa, couch, chesterfield*) are close to one another in this space. Words that are related to one another (e.g., *woman, girl*) are close, but less close than the synonymous words. Words that are unrelated (e.g., *girl, chesterfield*) are far apart. Search can map the documents to embeddings and search over those.[6] The user query is mapped into the same embedding space and the closest documents are retrieved. The score that indicates how close the documents are in the semantic embedding space is used in ranking and as a threshold to decide which documents should be in the result set and which not. The remainder of this section outlines where embedding-based search works well and where it does not.

Where Embeddings Work Well DL embeddings often work well to bridge vocabulary mismatches between user queries and documents [4, 9, 18]. Instead of explicitly using synonyms via normalization or expansion, the semantic space puts words that are similar close together. A strength and weakness of this approach is that the line between true synonyms and related words is blurred with no clear cut-off

[6] Indexing and retrieving embeddings efficiently is a major challenge. It will not be discussed here.

Fig. 4 Results for the query *cougar* using embedding based search. The queries *puma* and *mountain lion* have almost identical results

between the two. This provides robustness in finding documents but can also result in related but non-exact documents being returned. Many embedding models also handle misspellings. Although embedding-based search can return relevant results for misspelled queries, it is not a replacement for a spell corrector, which can be used to message the user for confirmation and correct queries into forms that will have more accurate results and improved ranking.

Some of the greatest potential for embedding-based search is for text-based search over images. Several recent DL models map text and images into the same semantic space [12, 21]. Images are mapped into embeddings, which are indexed. The user's text query is then mapped into an embedding and the semantically closest images are returned by the search engine. Ranking features other than the semantic similarity can be used to determine the final ranking. This ensures that high-quality or popular images are returned and helps provide more visual diversity in the images. An example of a search for *cougar* is shown in Fig. 4. Results are almost identical for the queries *puma* and *mountain lion*, which are synonyms for *cougar*, even though no synonyms were added to the system.

In many cases the DL embedding models are trained on large, general domain data. This provides broad coverage, but can have issues for more specialized queries and unusual domains. The embeddings can be customized, referred to as fine-tuning, for a particular domain such as search on a fashion eCommerce site.

Where Embeddings Fail There are situations where embedding-based search does not work well. Some of these are classes discussed above which require more structure, including negation and syntactic structure. However, structure in the form of ngrams (e.g., *milk chocolate* vs. *chocolate milk*) are captured well in many embedding models. So, in these cases embeddings are not worse than inverted index-based search, but they are not always improvements.

However, embedding-based models perform much worse than inverted indices when words distribute very similarly but have meanings that are crucially distinct for

search. Model numbers in eCommerce search are an example of these. When a model number occurs in a query, users want exactly that model, but embedding-based search will often return a seemingly random set of model number results. In fact, numbers in general behave in this way since they occur in similar environments (e.g., modifying the same nouns with the same adjectives and verbs surrounding those nouns) and so the embeddings treat them like synonyms. This means that results for queries like *dresses over $100* will not be relevant unless techniques as described in Sect. 3.2 are used. Names of people can also behave this way where names of the same gender and ethnicity incorrectly distribute like synonyms for one another.

4 Towards a Solution: Incorporating Structure

Section 3 outlined where neither inverted indices nor embedding-based search solve issues with search result quality. In this section we first discuss how more structured data can be used to solve these (Sect. 4.1). We then show how robustness techniques can be integrated to further enhance the results (Sect. 4.2). Finally we discuss a detailed example of combining these techniques, using color-object queries to search over images as an example (Sect. 4.3) and then sketch techniques for addressing negation in search (Sect. 4.4).

4.1 Structure Where It Matters

In the examples in Sect. 1, we saw how structured data was need to provide accurate search results for certain classes of queries. Why isn't structured data of this type used more pervasively? The first reason is that although there are established techniques for creating and searching over structured data, models for query and document understanding have to be build for the specific domain and use case. This means that effort is focused on the most important uses cases. The second is that searching over structured data is generally slower than search over inverted indices: The query understanding models are run and then the search over the structured data, often in addition to the standard search. Finally the ranking models have to take these new structure-matched features into consideration.

In general domain search, including web search, special data around entities is commonly used, including information about relations between entities [3]. The entities are identified in the documents and linked to canonical forms. Documents with matches to the query entities are ranked highly. Information learned about the entities can be used to create search page features such as entity boxes, answer boxes, highlighting in search result captions, and autocomplete [8] suggestions that map directly to the entity, thereby avoiding spurious matches (e.g., so that *tracy king* does not retrieve books about kings by authors with Tracy as one of their names). eCommerce often makes use of special entity and structured data mappings and even

simple reasoning (e.g., for price queries like *dresses over $100*). Identification of brands, categories and departments, sizes, and prices can all be canonicalized and mapped to structured data [15, 23]. Often these involve straight-forward matching once the type of entity and its canonical form is identified.

Syntactic structure is used less frequently in search engines. However, it can be used to identify MWEs that might otherwise be missed. For example, if *Mickey Mouse* is treated as a MWE, it is necessary to analyze queries and titles like *mickey and minnie mouse* in order to determine that the MWE *mickey mouse* is correctly included [22]. Similarly, price queries cannot be processed with simple entity detection of prices but instead require basic syntactic or at least ngram understanding because the preposition used indicates the range (e.g., *over $100* vs. *under $100* vs. *from $100 to $250*).

4.2 Robustness Where Needed

Inverted indices make use of text normalization, stemming and lemmatization, synonyms, and spell correction to improve the robustness of search, especially in mismatches between the words used in the user's query and the documents. In addition, DL embeddings can further improve robustness. A specialized example of robustness with embeddings is color matching. As highlighted by Fig. 5, the color blue and hence the images to which the word *blue* refers can occur to a broad range of shades, including ones that merge into greens and purples. By mapping the word *blue* into a color embedding, this embedding can then be matched against color embeddings of images to be searched. Images closer to blue will be closer to the core *blue* embedding and so can be ranked higher. In addition, the color wheel shown in Fig. 5 can be used to let the user select an exact shade of blue to match, something which is difficult to do with text queries.

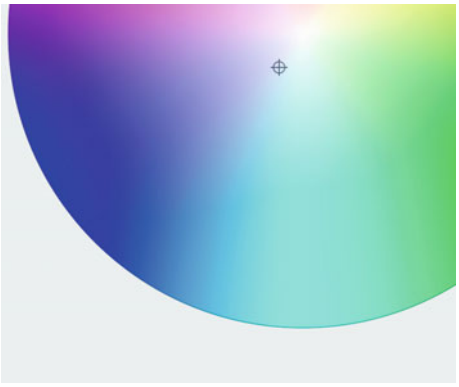

Fig. 5 Shades of blue: Color embeddings can allow the search engine to match into the blue, blue-green, and blue-purple color space

The issue with robustness is that in addition to retrieving relevant documents, it can also retrieve related and even irrelevant ones when the expansion compared to the query words is too great. This is especially the case for embeddings when the similarity score is the only control over what to match. As a result, robustness techniques are sometimes reserved for when a query has zero (referred to as null) or low results. Spell correction is often used in this way, only applying when the original query returns fewer than a fixed threshold of results. Similarly, embedding-based search may be reserved for null and low result queries [26] or for longer queries, which by definition are more likely to have fewer results because all the words have to match, thereby reducing the number of possible matches.

4.3 Color-Object Queries

In this section, we examine color-object queries when searching over stock images.[7] These queries are ones where a color word modifies an object. There may be more than one color-object pair in a query. The results for the query *white rose red background* were shown in Fig. 1. Color-object queries are relatively common in image search since users are looking for images to match their exact needs, including branding and marketing materials. When incorrectly matched results are shown, such as red roses on white backgrounds for *white rose red background*, the error is particularly jarring since the images look much different than expected. In addition, there is no way for the user to refine their query in order to see only relevant results. Finally, there is a long tail of color-object queries, including less frequent color names that overlap with non-color uses of the words (e.g., *salmon*).

The reason that irrelevant results are returned is that the inverted index contains words from the image tags and captions and there is no way to capture word order for the tags since they are just a set of words associated with the image. Among the images returned by matching words in the tags and captions, images with red roses on white backgrounds are much more common than ones with white roses on red backgrounds. So, by random selection, irrelevant results will be more common. This is exacerbated by the fact that queries for red roses are more common than for white ones and queries for images with white backgrounds are also common. This in turn results in more clicks and purchases for those images and higher popularity features in the ranking.

The treatment of color-object queries for image search requires improved document and query understanding for matching and ranking. For the images, prominent objects, including the background, have to be identified. The colors of these objects are determined. The information about the objects, colors, and the relations between the two are stored in the index as structured representations. For the queries, the color words and the objects they modify have to be identified and canonicalized. The representations for the colors in the index and the query have to be compared,

[7] The approach described here is based on techniques used in Adobe Stock search.

as do the objects and the color-object relations. Once we know which images match the query with respect to the color-object relationships, this information can be used in matching to return only images where the color-object relationships are identical or can be used in ranking to rank images with the requested color-object relationships higher in the result set. The matching approach is more aggressive, leading to higher precision results but risking excluding some relevant results. The ranking-only approach is more conservative, returning all the results the non-structured data would have returned but thereby including irrelevant results. The decisions for each of these steps can be complex. They are described in more detail below as an example of how powerful but complex incorporating structured data into search can be.

Color-object Queries For the queries, first we use a named entity recognition (NER) model to identify color phrases in queries. Depending on what types of colors are used in queries, the model can cover "kindergarten" colors (the approximately ten most basic colors, e.g., *red rose*), modified colors (e.g., *pale yellow rose*, *hot pink rose*), or the long tail of colors (e.g., *salmon rose*, *chartreuse rose*). The NER model should handle coordinated colors as well (e.g., *pink and yellow roses*), including determining their logical meaning (e.g., each rose is both pink and yellow; some roses are pink and some are yellow but there must be at least one of each). Finally, the NER must avoid detecting false positives where a color word occurs but does not refer to a color in a color-object construction. False positives include ethnicities (e.g., *black nurse*, *white nurse*), proper nouns (e.g., *snow white*), and styles (e.g., *black and white portrait*). Once a color is detected in the query, the object it modifies has to be determined. A dependency parser that is custom-trained to work on short text like queries can be used for this [25, 28]. The color NER model and the dependency parser are language dependent and so models have to be created for each language.

Color-object Images For the images, first we need to identify the prominent objects in the image and mask them. Masking determines the edges of the object so that we can extract the dominant color for each object. Each object has to be labeled as to what it is (e.g., a rose, a dog, a ball). The list of potential object labels can be extracted from the queries, in particular from the objects that occur in color-object queries.[8] The object labeling is done by an autotagger model and the confidence of the model can be used to adjust how accurate the labels are. The image background is a special case. Backgrounds are commonly referred to in color-object queries (e.g., *banana blue background*, *rose white background*). The background can be identified by masking out all the prominent objects: what is left is considered background. The background color is determined the same way that object colors are. The background label is simply as *background*; no autotagger is needed. The color-object pairs have to be stored in the index as structured data so that the specific color is associated with its object.

Color and Object Representations Consider how to represent the colors. For the index, one possibility is to use a model that maps from an image to the colors used by the query color NER model. However, this becomes difficult to manage if

[8] If the object extraction and labeling will be used for other features, then the model should label a broader set of objects.

the range of colors goes beyond the unmodified kindergarten colors. For example, for modified colors (e.g., color words modified by *light*, *pale*, *dark*, *neon*, *hot*), the index would have to list both the modified version (e.g., *dark blue*) and the simple version (e.g., *blue*) so that either type of query color could match (e.g., the queries *dark blue ball* and *blue ball* match a dark blue ball, but the query *light blue ball* does not). As discussed in Sect. 3.4, embeddings are an effective representation for colors due to their continuous nature. Instead of storing the colors as text, the embeddings can be used. These embeddings are language-independent because they represent the colors of the pixels in the image. If the colors are stored as embeddings in the index, the color words detected by the query NER have to be mapped to color embeddings. This can be done with a multi-modal text-to-color model. This model is language dependent due to the query text component.

Next consider how to represent the object labels. These could be English words. If this is the case, then for search in other languages the object words have to be translated into English. This is similar to synonym normalization (Sect. 3) only between languages instead of synonyms within a language. Alternatively, the language variants for each object label could be stored in the index, which could result in many entries for each object (i.e., one for each language). This is similar to synonym expansion only across languages instead of across synonyms within a language. An alternative is to map into language-independent concepts, either to concepts in a taxonomy or knowledge graph or to an embedding representation [11]. If the concept approach is chosen, the image autotagger has to map into these concepts so that they can be stored in the index. Then the query processing maps the object words identified by the dependency parser into the concepts.

Finally, the relationship between the color and object has to be represented and associated with the image in which they occur. This can comprise a dedicated index field for the objects in the image. Each object is then associated with at least one structured attribute, namely its color.

Color-Object Example Consider the image in Fig. 6. Two objects are detected in the image: the cup and the pastry. The remainder of the image is considered the background. The objects are labeled by an image autotagger, shown in Fig. 6 as concepts associated with English words to make them readable. Each object and the background are then associated with a color embedding that is derived from the object image, shown in the figure as a color swatch but in fact represented as a vector. The object concept and its associated color embedding are in turn associated with the document id. A list of words from tags or a caption is also associated with the image.[9]

Consider the query *pastry and red cup*. Color NER detects the color word *red*. The query dependency parser determines that the color modifies the object *cup*. The text-to-color multi-modal model maps the word *red* to an embedding. This embedding is not an exact match to the one for the coffee cup in Fig. 6 because the word *red* maps to the most canonical representation of the color, while the coffee cup has shadowing

[9] These are shown as English words in Fig. 6 but could be concepts similar to the ones used for the color-object representation. The same holds for the word *pastry* in the query in Fig. 7.

Structured Color-object Field Image Data		
Doc. Id	**Object (Concept)**	**Color (Embedding)**
1	Concept_123_cup	:
1	Concept_456_pastry	:
1	Concept_789_background	:

Words Associated with the Image	
Doc. Id	**Words**
1	cup, saucer, pastry, croissant, coffee, yellow, red, tasty, breakfast

Fig. 6 Image with two objects (a red cup and a brown pastry) on a yellow background. The index information is shown in the table below the image. The objects are represented as concepts (represented as Concept_*num_word* for exposition) and an associated color (represented as a color swatch corresponding to a color embedding). *Image licensed from Adobe Stock*

User query:	*pastry and red cup*
Color-object Field:	Concept_123_cup :
Words:	pastry

Fig. 7 Mapping of the query *pastry and red cup* for retrieval with color-object structured data and an inverted index

on it. The object word *cup* is mapped to a concept and associated with the color embedding. The word *and* is dropped as a stop word. The word *pastry* is treated as any other word for search since it is not part of a color-object relationship. The resulting query is shown in Fig. 7. When the query is matched against the index, the word *pastry* matches against the words in document 1. The concept Concept_123_cup matches with the object concept in document 1. The color embeddings for Concept_123_cup in the query and image are then compared. These are not an exact match since the shades of red are different. However, since they are within a pre-defined similarity threshold, they are considered a match and document 1 will be returned for the query.

The treatment of color-object queries for image search shows the power behind using structured data in conjunction with embeddings and inverted indices for search.

However, it also demonstrates the complexity of creating such structured data and integrating it into the search matching and ranking.

4.4 Towards Negation

Section 3.3 discussed why negation is complex for search, using the example query *snacks without nuts* illustrated in Fig. 2. Here, we discuss potential approaches to handling negation in search.

Negation in Queries To handle queries with negation like *snacks without nuts*, search has to determine that the query contains a negation (e.g., *without*) and the scope of that negation (e.g., *nuts*). Most queries are relatively short and there are a limited number of ways to indicate negation, which makes the detection and scope of negation easier to determine than with standard long text such as that found in documents.[10] However, once the negation and its scope is determined, search then has to correctly and efficiently retrieve documents, e.g., ones which are snacks but which do not contain nuts.

An overly simplistic approach is to retrieve all documents matching *snacks* and then exclude documents which contain the word *nut* or to rank documents with the word *nut* lower than documents without it. Unfortunately, this approach still includes many results with nuts, i.e., documents which refer to specific instances of nuts (e.g., *almonds, walnuts*). Conversely, relevant documents are excluded or ranked unduly lowly if they contain the word *nut* in phrases like *nut free*, *no nuts*, or *kids are nuts about this snack*.

Given the importance of ingredients in food items, a specialized solution can be implemented, as was done for color-object search for photos (Sect. 4.3). The ingredient lists of food items can be enhanced offline with relevant hypernyms (e.g., *nut* for *pecan*, *dairy* for *milk*). These hypernyms can be treated as keywords or as part of the structured data for the ingredients, which would also enable filters for or against specific ingredients. If search correctly identifies the query *snacks without nuts* as a food query with negation, it can match all snack documents that do not contain the word *nut* in the ingredient field or structured data. If it does not identify the query as a food query, search can back off to the approach of returning snack documents which do not contain the word *nut* in any of the text fields.

The specialized solution approach requires anticipating which classes of attributes will occur with negation. However, users can apply negation to many types of attributes (e.g., *lamp without shade*, *cup no handle*, *cities without rent control*, *plastic free restaurants*). Handling negation more broadly requires a more systematic approach. Once the negation and its scope are detected in the query, the documents

[10] Negated phrases add an extra layer of complexity in determining the scope of negation and matching that against the documents. For example *dresses without red stripes* should match dresses with blue stripes or with red polka dots since the negation applies to the concept denoted by the phrase *red stripes*. See [5, 6] for more discussion with a particular focus on question answering and textual inference.

matching the non-negated parts of the query can be retrieved. Documents mentioning the negated words can then be excluded or demoted. The remaining documents can then be checked for hyponyms of the negated word (e.g., *almonds*, *walnuts* etc. for *nuts*) and documents containing these hyponyms can be excluded or demoted. Special care has to be taken to allow for phrases like *nut free* which capture the negation within the document and hence are excellent matches for the corresponding negative query. The above approach requires more calculation at query time because a potentially large list of hyponyms has to be checked against a potentially large number of documents. To avoid this, hypernyms can instead be added to the index. This is similar to the specialized ingredient solution but on a larger scale for all of the non-stop words in the document. This has the downside of adding to the size of the index, especially to the number of document ids associated with the more abstract hypernyms. A similar trade-off was discussed in Sect. 3.1 for synonyms and other vocabulary mismatches.

Negation in Documents The above discussion outlined some methods for addressing negation in queries. However, negation also exists in documents. The search document processing and indexing must handle negation in documents when providing results for queries, whether negated or not. A simple example of this is the query *nut snacks* which should not match documents that have phrases like *a nut free snack*, *contains no nuts*, or *does not contain nuts*. A more subtle example is the query *leather jacket* which often matches faux leather jackets, where the word *faux* indicates that the jacket is not made of leather. If the user is looking for jackets in the style of leather jackets, these non-leather results are fine, but if they want a jacket made of leather, these results are irrelevant and it is difficult to construct a query to eliminate them. Often the only way to exclude these non-leather results is to use a filter for *material=leather*, if available.

Inverted indices are not well designed for encoding words as negated concepts since they are optimized to provide word-document lists (Sect. 2.1). Deciding not to index negated terms runs two risks. First, these can be perfect matches for negated queries since they are explicit about not involving the negated attribute. Second, if the processing incorrectly identifies the word as being negated, then that document will never be returned for that concept since the word is not indexed. Instead, the negated concept or words need to be associated with the information that they are negated. This takes the form of a basic structured representation. This representation allows negated queries to match directly against similarly negated document information. It also allows non-negated queries (e.g., *nut snacks*, *leather jacket*) to avoid matching against documents which negate words that should be matched (e.g., *nut* in *nut free*, *leather* in *faux leather*). This information is more costly to compute, store, and match against, but is necessary to effectively handle negation in documents and corresponding queries. Once such information is available, search can then use it for matching or, more conservatively, for ranking documents with the negated words lower than ones with the non-negated words.

This section briefly explored the complexity of handling negated queries and documents in search and some potential solutions. There is no single, simple solution,

especially when large numbers of documents have to be searched over quickly. So, handling negation remains an unsolved, but crucial research problem in search.

5 Conclusion

Search historically depends on inverted indices for matching user queries to relevant documents. User behavioral data provides additional signals for query-independent and query-dependent relevance of documents. These inverted indices enable rapid search over large document collections. Optimizations in text processing, normalization, and expansion have improved search's ability to return relevant documents. However, these approaches have issues around robustness when there are mismatches between users' queries and the documents, something which is highlighted when using textual queries to search for images. Moving to representations like embeddings from DL models improves robustness for some types of data and are particularly useful for image data (Sects. 4.2 and 4.3).

However, there remain classes of queries which require information beyond the representation of words and multi-word expressions stored in an inverted index or as an embedding. These are the structured representations referred to in the title of this paper. They involve searching over structured information about the content and relationships among the entities. These structured representations in turn must support reasoning (e.g., *snacks without nuts*).

Given the specialized knowledge and models needed to create and search over this structure and given the increased cost in latency and space to search and reason over the structured data, these approaches are reserved for high-value queries which have low quality results with traditional techniques. A simple example of this is identifying brands in eCommerce queries, canonicalizing them, and then matching them against the structured brand data on the products. More complex structured data is required for representing document-internal relationships. The color-object image search queries are an example of this, where the images have to have color data associated with the objects in order to avoid cross-talk bag-of-words results where the color is present in the image but on the wrong object (e.g., for the query *white rose red background* showing red roses on a white background). The step beyond the matching to structured data and relations across entities is to be able to reason about the documents in order to provide relevant results and even answers to the users' queries. Price queries in eCommerce search (e.g., *dresses over $100*) are an example of these. For price queries, not only does the price in the query have to be identified and matched to the product price, but the word *over* (or *under* or *around* or *from . . . to*) has to be identified and associated with the correct reasoning (arithmetic operation).

The future of search is going to be hybrid, involving techniques from classic information retrieval, embedding-based search, and reasoning. These hybrid search techniques will be increasingly powered by improved query and document understanding via the integration of structured data into search matching, ranking, and

reasoning itself. Enabling such hybrid systems will require a deep understanding of linguistic representations of meaning, of information retrieval optimization, and of the types of information encoded in the queries and documents. It is my hope that this paper inspires further collaboration across disciplines to improve search.

For readers interested in learning more about search, there are several excellent textbooks available. References [2, 3, 17] focus on the fundamentals of search, especially inverted indices and text processing (e.g., normalization, expansion, stop words, stemming). Balog [3] examines entity-based search and how deeper understanding of entities can be used to improve all aspects of search. Baeza-Yates [1] focuses on semantic query understanding. SIGIR (Special Interest Group—Information Retrieval) is the annual conference focused on search with a published proceedings and the ACM SIGIR Forum (https://sigir.org/forum/) publishes additional papers on search. There are two annual workshops focused on eCommerce search: ECOM [13] and ECNLP [16], both of which publish proceedings. Tsagkias et al. [27] provides an overview of search issues as pertain to eCommerce.

Acknowledgement I would like to thank Roussanka Loukanova for inviting me to present at Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021) and to contribute to this volume. I would also like to thank the audience of LACompLing2021, four anonymous reviewers, and Annie Zaenen for insightful questions and comments.

I would like to thank the Adobe Sensei and Search team who developed the color-object search techniques discussed in Sect. 4.3: Baldo Faieta, Ajinkya Kale, Benjamin Leviant, Judy Massuda, Chirag Arora, and Venkat Barakam.

References

1. Baeza-Yates, R.: Semantic query understanding. In: Proceedings of SIGIR. ACM (2017)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval: The Concepts and Technology Behind Search, 2nd edn. Addison-Wesley (2011)
3. Balog, K.: Entity-Oriented Search. The Information Retrieval Series, vol. 39. Springer (2018)
4. Bianchi, F., Tagliabue, J., Yu, B.: Query2Prod2Vec: grounded word embeddings for eCommerce. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers, pp. 154–162. Association for Computational Linguistics (2021)
5. Bobrow, D., Condoravdi, C., Crouch, R., Kaplan, R., Karttunen, L., King, T.H., de Paiva, V., Zaenen, A.: A basic logic for textual inference. In: Proceedings of the AAAI Workshop on Inference for Textual Question Answering, pp. 47–51 (2005)
6. Bobrow, D., Crouch, D., King, T.H., Condoravdi, C., Karttunen, L., Nairn, R., de Paiva, V., Zaenen, A.: Precision-focused textual inference. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pp. 16–21 (2007)
7. Buttcher, S., Clarke, C.L.A., Cormack, G.V.: Information Retrieval: Implementing and Evaluating Search Engines. The MIT Press (2016)
8. Cai, F., de Rijke, M.: A survey of query auto completion in information retrieval. Found. Trends Inf. Retr. **10**, 1–92 (2016)
9. Chang, W.C., Jiang, D., Yu, H.F., Teo, C.H., Zhong, J., Zhong, K., Kolluri, K., Hu, Q., Shandilya, N., Ievgrafov, V., Singh, J., Dhillon, I.S.: Extreme multi-label learning for semantic matching in product search. In: Proceedings of KDD2021 (2021)

10. Chen, Q., Li, M., Zhou, M.: Improving query spelling correction using web search results. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 181–189 (2007)
11. Chen, X., Cardie, C.: Unsupervised multilingual word embeddings. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 261–270 (2018)
12. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q.V., Sung, Y., Li, Z., Duerig, T.: Scaling up visual and vision-language representation learning with noisy text supervision. In: Proceedings of the 38th International Conference on Machine Learning PMLR (2021)
13. Kallumadi, S., King, T.H., Malmasi, S., de Rijke, M. (eds.): Proceedings of the SIGIR 2021 Workshop on eCommerce. CEUR-WS (2021)
14. Kalouli, A.L., Crouch, R., de Paiva, V.: Hy-NLI: a hybrid system for natural language inference. In: Proceedings of the 28th International Conference on Computational Linguistics, pp. 5235–5249. International Committee on Computational Linguistics (2020)
15. Kutiyanawala, A., Verma, P., Yan, Z.: Towards a simplified ontology for better e-commerce search. In: Proceedings of ECOM2018. CEUR-WS (2018)
16. Malmasi, S., Kallumadi, S., Ueffing, N., Rokhlenko, O., Agichtein, E., Guy, I. (eds.): Proceedings of The 4th Workshop on e-Commerce and NLP. Association for Computational Linguistics (2021)
17. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
18. Mohan, V., Song, Y., Nigam, P., Teo, C.H., Ding, W., Lakshman, V., Shingavi, A., Gu, H., Yin, B.: Semantic product search. In: Proceedings of KDD2019 (2019)
19. Peters, C., Braschler, M., Clough, P.: Multilingual Information Retrieval: From Research to Practice. Springer (2012)
20. Porter, M.F.: An algorithm for suffix stripping. Program **14**, 130–137 (1980)
21. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision (2021). ArXiv:2103.00020
22. Senthil Kumar, P., Salaka, V., King, T.H., Johnson, B.: Mickey Mouse is not a phrase: improving relevance in E-commerce with multiword expressions. In: Proceedings of the 10th Workshop on Multiword Expressions (MWE), pp. 62–66. Association for Computational Linguistics (2014)
23. Skinner, M., Kallumadi, S.: E-commerce query classification using product taxonomy mapping: a transfer learning approach. In: ECOM SIGIR Workshop. CEUR-WS (2019)
24. Sorokina, D., Cantú-Paz, E.: Amazon search: the joy of ranking products. In: Perego, R., Sebastiani, F., Aslam, J.A., Ruthven, I., Zobel, J. (eds.) Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, pp. 459–460. ACM (2016)
25. Sun, X., Wang, H., Xiao, Y., Wang, Z.: Syntactic parsing of web queries. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1787–1796 (2016)
26. Trotman, A., Degenhardt, J., Kallumadi, S.: The architecture of eBay search. In: Degenhardt, J., Kallumadi, S., de Rijke, M., Si, L., Trotman, A., Xu, Y. (eds.) Proceedings of the SIGIR 2017 eCom workshop. CEUR-WS (2017)
27. Tsagkias, M., King, T.H., Kallumadi, S., Murdock, V., de Rijke, M.: Challenges and research opportunities in eCommerce search and recommendations. ACM SIGIR Forum **54**, 1–23 (2020)
28. Wang, Z., Wang, H., Hu, Z.: Head, modifier, and constraint detection in short texts. In: Proceedings of the International Conference on Data Engineering, pp. 280–291 (2014)
29. Wang, Z., Zhao, K., Wang, H., Meng, X., Wen, J.R.: Query understanding through knowledge-based conceptualization. In: Proceedings of IJCAI, pp. 3264–3270 (2015)