A Versatile Hypergraph Model for Document Collections

Andreas Spitz
École polytechnique fédérale de Lausanne
andreas.spitz@epfl.ch

Dennis Aumiller
Heidelberg University
aumiller@informatik.uni-heidelberg.de

Bálint Soproni
Heidelberg University
soproni@stud.uni-heidelberg.de

Michael Gertz
Heidelberg University
gertz@informatik.uni-heidelberg.de

ABSTRACT

Efficiently and effectively representing large collections of text is of central importance to information retrieval tasks such as summarization and search. Since models for these tasks frequently rely on an implicit graph structure of the documents or their contents, graph-based document representations are naturally appealing. For tasks that consider the joint occurrence of words or entities, however, existing document representations often fall short in capturing cooccurrences of higher order, higher multiplicity, or at varying proximity levels. Furthermore, while numerous applications benefit from structured knowledge sources, external data sources are rarely considered as integral parts of existing document models.

To address these shortcomings, we introduce *heterogeneous hypergraphs* as a versatile model for representing annotated document collections. We integrate external metadata, document content, entity and term annotations, and document segmentation at different granularity levels in a joint model that bridges the gap between structured and unstructured data. We discuss selection and transformation operations on the set of hyperedges, which can be chained to support a wide range of query scenarios. To ensure compatibility with established information retrieval methods, we discuss projection operations that transform hyperedges to traditional dyadic cooccurrence graph representations. Using PostgreSQL and Neo4j, we investigate the suitability of existing database systems for implementing the hypergraph document model, and explore the impact of utilizing implicit and materialized hyperedge representations on storage space requirements and query performance.

CCS CONCEPTS

• **Information systems** → **Document collection models**; • **Theory of computation** → *Data modeling*.

ACM Reference Format:
Andreas Spitz, Dennis Aumiller, Bálint Soproni, and Michael Gertz. 2020. A Versatile Hypergraph Model for Document Collections. In *32nd International Conference on Scientific and Statistical Database Management (SSDBM 2020), July 7–9, 2020, Vienna, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3400903.3400919

SSDBM 2020, July 7–9, 2020, Vienna, Austria

1 INTRODUCTION

What makes a good document representation? The answer to this question, of course, heavily depends on the context. First and foremost, different applications require different information needs to be taken into account. Second, it must be possible to effectively realize the basic methods that operate on the documents on some computing infrastructure. To address the diversity of information and application needs, many document models and storage solutions have been proposed, ranging from simple bag-of-word models to the more sophisticated word embeddings [6]. Built on top of these models, there is a plethora of applications designed in support of diverse information retrieval (IR) tasks, such as search, query expansion, text summarization, or document classification, to name but a few [28, 49]. A closer inspection of these IR tasks reveals the central role of word dependency. That is, information about word cooccurrences in textual documents is a key concept of models such as latent semantic indexing, topic models, or word embeddings. As a result, word cooccurrence statistics have been studied extensively in the past [17, 44]. Although many such models were initially based on the concept of words, they can also be considered on the more general basis of *terms*, which include both individual words and multi-word expressions such as named entities.

As is reflected in J.R. Firth's well known quote *"you shall know a word by the company it keeps"*, some of the most discriminative features of a term are the other terms that occur in its proximity [18]. The use of *company* in his quote also expresses an aspect of connectedness and has a social connotation, which immediately suggests a network of sorts. After all, a network (or, formally, a graph) is the natural model of choice whenever a set of things is connected. It thus comes as no surprise that this notion term context has led to numerous approaches in which cooccurrence information is modeled in the form of graphs [9, 10, 26]. In a similar vein, this thought leads to the idea of heterogeneous information networks, which employ knowledge bases to associate external information with terms in documents [14, 31, 37, 40]. While such graph structures naturally provide a topological embedding of terms or entities in a textual context, the modeling of this context is constrained to pairwise relationships and term cooccurrences due to the limitations of binary (or *dyadic*) edges in the employed graphs. It is well known that such dyadic graph models suffer from several shortcomings in modeling higher-order relationships with more than two participating terms or entities. As a result, such models tend to be tailored to a specific task and are notoriously difficult to generalize.

To address these shortcomings of existing models, we propose a document model that is based on the concept of hypergraphs [7, 8].

In contrast to dyadic graphs, in which an edge always connects exactly two nodes, hypergraphs allow the representation of higher-order relationships as hyperedges that connect an arbitrary number of nodes. Furthermore, we include not just terms as nodes, but also structural components of the document, such as sentences, annotations, and the document itself. Together with the linking of nodes to entities of different classes in external knowledge bases, this leads to the notion of a hypergraph over a heterogeneous set of nodes. While this model for higher-order term cooccurrences is valuable in itself, our primary contribution and key to the practical application of the model is a core set of operators that support a variety of IR tasks. In the spirit of the base operators of relational algebra [11], we introduce a generalized set of operators on hypergraphs that allow the selection and transformation of hyperedges to satisfy information needs and retrieval operations with respect to the documents. The operators are designed to seamlessly integrate structured data from external knowledge bases into the unstructured text.

Contributions. In summary, we make five primary contributions in an extension of a prior conceptual model [38]:

(*i*) We propose a hypergraph-based document model for describing higher-order cooccurrences among a set of heterogeneous nodes that represent the documents' components.
(*ii*) The model is designed to represent text annotations together with data from knowledge bases and thus bridges the gap between structured and unstructured data.
(*iii*) We propose fundamental operators on such heterogeneous hypergraphs for filtering and selecting nodes and (subsets of) hyperedges to support a wide range of applications.
(*iv*) We show how the model can be used in combination with the proposed operators to realize key tasks in IR.
(*v*) We compare implementations of the system in a relational database and a dedicated graph database with respect to their memory consumption and query performance.

Structure of the paper. In Section 2, we review related work for term cooccurrences and hypergraphs. In Section 3, we introduce the hypergraph document model, and describe fundamental operations on such graphs in Section 4. In Section 5 we show how key IR methods can be realized with our model. Section 6 compares practical implementations with respect to their performance.

2 RELATED WORK

We split the related work into two broader categories: (1) network-based modeling of term cooccurrences and (2) hypergraphs.

2.1 Network-based Term Cooccurrences

Modeling and analyzing word cooccurrences has a long tradition and was likely first considered formally by Van Rijsbergen, who proposed to drop the term independence assumption and measure term dependencies with non-linear weighting functions [44]. Since then, a multitude of approaches have built on term cooccurrence information and statistics (for a comprehensive overview, see [17]). Unsurprisingly, term cooccurrence information has applications in a variety of IR settings, including similarity measures for words [12, 48], query expansion [2, 32], extracting keywords from documents [27, 29], or constructing low-dimensional vector embeddings from term cooccurrences [33].

Some recent works model term cooccurrences as networks to describe term relationships in a more context-oriented framework and employ network analysis for the derivation of measures or to compare language specific networks [9, 10, 22, 25, 26]. Others exploit the properties of such networks to learn document representations and context-dependent relationships through embeddings [43].

The above networks include words and terms without considering external information sources. More recently, typed cooccurrence networks have been introduced, which include cooccurrences of named entities that are detected and extracted from the documents. Nodes still represent terms, but are also associated with entity types, such as *person* or *location*. Examples of such networks include the LOAD approach for cross-document extraction and summarization of events [39, 40], entity graphs used for identifying entities in trending events [36], and time-term association graphs to estimate the focus time of documents [23].

While these works incorporate named entity recognition and cooccurrence information, many other approaches additionally harness structured knowledge bases such Wikidata [45], either to construct networks directly, or to link extracted terms to entries in the knowledge base. Examples include coreference resolution of entity mentions [14], query feature expansion from links to knowledge bases [13], and determining the semantic relatedness of documents [31]. On this side of the spectrum of term networks, one eventually encounters (heterogeneous) information networks [37]. Such networks, however, are focused less on term cooccurrences and more on knowledge-base like relationships. General models that fully merge structured knowledge and unstructured text data independently of the application are still missing.

2.2 Hypergraphs

A major shortcoming of the above approaches to modeling term relationships is the restriction to a dyadic graph model, which is insufficient for modeling the joint cooccurrences of multiple terms.

To address these shortcomings, recent approaches increasingly utilize hypergraphs. Hypergraphs have been studied extensively in graph theory [7, 8], but have not seen frequent use in practice due to their computational complexity and difficult realization based on existing data management infrastructure. However, as is evident from recent publications, hypergraph implementations scale well on novel computing infrastructures. Heintz et al. discuss several challenges and opportunities when realizing hypergraph management systems [19] and present a flexible, distributed and scalable processing system for hypergraphs called MESH [20]. Huang et al. introduce the HyperX system, which supports efficient learning on and processing of distributed hypergraphs in Spark [21].

Nowadays, one can find emerging approaches that successfully employ hypergraph models for diverse graph management and analysis tasks. On the one hand, some approaches extend traditional network analysis tasks to hypergraphs, such as clustering coefficients [16], centrality measures [24], or spectral clustering [47]. On the other hand, hypergraph-based approaches have also found use in traditional document-based IR settings, such as sentences as hyperedges of words to enable random-walk based metrics for sentence ranking [3, 4], or hyperedgs over the set of sentences in a document for semi-supervised extractive summarization [46].

Bendersky and Croft utilize hypergraphs to model queries (but not documents) in a way that includes term and phrase dependencies and improves subsequent retrieval operations [5]. Hypergraphs have also been proposed as a basis for recommendation systems, for example in tagging data [50] and music recommendation [42].

Almost all the above approaches come with their own specialized hypergraph model, ranging from a pure document-oriented view to modeling complex relationships between terms or features within the documents. Furthermore, none of these approaches utilize hypergraphs for generalized term cooccurrences to support other application frameworks. Given the plethora of information retrieval and exploration approaches based on term cooccurrences, it is pertinent to have a single yet flexible document model that can support the majority of these applications. In the following, we present such a document model based on hypergraphs, including necessary operators to enable a wide range of applications.

3 HYPERGRAPH DOCUMENT MODEL

Before we introduce the hypergraph model, we define the underlying concepts and discuss document segmentation strategies.

3.1 Preliminaries

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_\mathcal{G})$ is a tuple consisting of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}_\mathcal{G}$. In most commonly used graphs, edges connect exactly two nodes, which means that $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V} \times \mathcal{V}$. In the following, we refer to such graphs as *dyadic graphs*. Dyadic graphs may be weighted, meaning that a weight is associated with each edge, or directed, meaning that the order of nodes in an edge is of relevance.

In contrast to dyadic graphs, edges in a hypergraph consist of an arbitrary subset of nodes. Thus, for a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_\mathcal{H})$, the set of edges is a subset of the power set of nodes $\mathcal{E}_\mathcal{H} \subseteq 2^\mathcal{V}$. Such edges are called *hyperedges*, but we refer to them as edges when the meaning is clear from context. Similar to edges in dyadic graphs, hyperedges may be weighted or directed, but we focus on unweighted, undirected hypergraphs in the following. A graph or hypergraph is called *heterogeneous* if the set of nodes can be partitioned according to some distinct attribute (consider, for example, entities in a knowledge graph representing persons or places).

In the following, we discuss how to segment documents in such a way that they can be represented as a type of heterogeneous hypergraph. External knowledge such as node type hierarchies are then easily included as dyadic graphs over the same set of nodes.

3.2 Document Segmentation

As unstructured text input, we consider a document collection $\mathcal{D}$. Each document $D \in \mathcal{D}$ may have associated metadata (e.g., an author or a publication date), and can be segmented into smaller units. Specifically, we use terms $\mathcal{T}$ as atomic units and sentences $\mathcal{S}$ as groups of terms. Thus, we consider a document $D \in \mathcal{D}$ to be a subset of sentences $D \subseteq \mathcal{S}$. In turn, each sentence $S \in \mathcal{S}$ is a subset of terms $S \subseteq \mathcal{T}$. In the following, we consider a term to be a word or a multi-word expression with a specific meaning in the given sentence. For example, both *apple* and *apple tree* could constitute terms, even though the latter consists of two words.

Note that additional granularity levels in this segmentation hierarchy are possible. For example, *phrases* can serve as groups of terms that are parts of sentences, *paragraphs* allow the modeling of groups of sentences, whereas *volumes* could represent sets of documents within a collection. For the sake of brevity, we do not include them here, but it should be obvious how they can be formalized as a possible hierarchy of sentences, in analogy to the following.

3.3 External Term Augmentations

The segmentation of documents into terms instead of words is both aided and required by the use of a model that transcends document knowledge and integrates external structured information into the unstructured text. To this end, we design the model to support the augmentation of identified terms with external knowledge, such as (named) entity information from knowledge bases. Since terms constitute the majority of nodes of the hypergraph, knowledge about terms can be modeled as node attributes, while ontological or hierarchical information provides dyadic graphs over the set of nodes. For example, the term *apple* can be tagged as part-of-speech *noun* or, depending on the context in the sentence, could be linked to an entity in a knowledge base representing the fruit or the technology company. Based on these links to knowledge bases, terms can be classified into categories and hierarchies. For example, the company *Apple* would be classified as an organization. Thus, additional term information constitutes attributes that are associated with the corresponding nodes. Naturally, named entity recognition and linking tools are a rich source of augmenting information.

Targets for entity linking could be gazetteers or knowledge bases such as Wikidata [45]. On a linguistic level, it is also possible to link terms based on lexical networks like WordNet [30]. Ideally, it should be possible to link any term to an underlying knowledge base or lexical resource. In reality, since information is often missing or incomplete, the resulting set of nodes is heterogeneous with regard to the available information. For terms that cannot be linked, it is reasonable to assume some form of lemmatization or stemming to ensure that distinct terms and lexemes with identical meaning are also mapped to the same node in the graph, similar to the linking of entities. Alternatively, terms can be clustered and linked according to (pre-trained) vector embeddings to represent dyadic semantic relations that are present within the document collection or a reference corpus, for example by using GloVe [33] or ELMo [34]. While the possibilities of linking terms to external sources are numerous, the approach can always be modeled as (heterogeneous) nodes that are linked to external dyadic graph structures for which numerous querying and reasoning approaches exist. In the following, we thus focus on representing the document collection itself. For this task, we put a focus on term cooccurrences, which cannot be handled adequately by dyadic approaches.

3.4 Hyperedge Composition

To introduce the construction principle of hyperedges, we first define the set of nodes and a system for describing term positions within the sentence structure of the documents. Ultimately, we obtain a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. External knowledge base structures can then be considered as dyadic graphs $\mathcal{G}_{KB} = (\mathcal{V}, \mathcal{E}_{KB})$ on the same set of nodes or a subset thereof.

3.4.1 Term and Sentence Position. To represent the occurrence of terms in sentences as well as the cooccurrences of terms, we

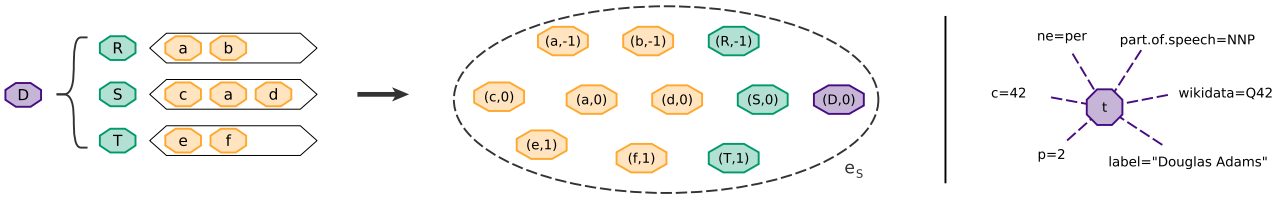

Figure 1: Left: document D with sentences R,S,T and terms $a-f$, along with the resulting hyperedge e_S for representing the primary sentence S at a window size of $w=1$. Right: example of a term occurrence t with attributes core $t.c$ and position $t.p$, as well as added part-of-speech and named entity annotations, and external Wikidata knowledge base attributes.

introduce the concept of relative term and sentence position as a function $p:(\mathcal{T}\cup\mathcal{S})^2\rightarrow\mathbb{Z}$. To this end, we assume that there exists a monotonic, consecutive numbering of sentences such that each sentence S has an identifier $id(S)\in\mathbb{N}$. For two sentences S_1 and S_2, we define their relative position as $p(S_1,S_2):=id(S_2)-id(S_1)$. Analogously, we define positions for terms. For a given occurrence of term t, let $s(t)\in\mathcal{S}$ denote the containing sentence. For two term occurrences t_1 and t_2 we then define their relative position as $p(t_1,t_2):=id(s(t_1))-id(s(t_2))$. Note that positions may be negative, that they are symmetric for inverse arguments, and that the absolute value $|p(t_1,t_2)|$ of a position score is a proper distance in the number of sentences between the term occurrences. For ease of notation, we also include a relative position between documents and sentences, by setting the position of a document D with regard to a sentence S to $p(D,S):=0$ iff $S\in D$ and $p(D,S):=\infty$ otherwise.

Since sentences represent coherent units of linguistic structure, we consider the above position scheme on the sentence level to be the most useful approach, and it is used in some of the typical IR applications that we replicate in Section 5. Of course, position functions based on paragraph or term distances are equally viable. While a monotonic and consecutive numbering of sentences is not strictly necessary, it is beneficial for materializing hyperedges from the underlying data at query time, as we discuss in Section 6.

3.4.2 Graph Nodes $\mathcal{V}$. To construct the set of graph nodes, we define the set $Core:=\mathcal{T}\cup\mathcal{S}\cup\mathcal{D}$ of terms, sentences, and documents. Based on unique identifiers for terms, sentences, and documents, this allows us to identify individual nodes. To satisfy the requirement of modeling cooccurrences across sentence boundaries, we include the position of the node core in the node representation. Thus, the set of nodes is given by $\mathcal{V}\subseteq Core\times\mathbb{Z}$. The first component denotes the identifier, while the second denotes the relative position with respect to the considered sentence.

3.4.3 Node Attributes. Each node as defined above is a tuple with the two primary components *core* (the term, sentence, or document identifier) and *position* (the relative position within the document). In the following, we refer to the core of a node $v\in\mathcal{V}$ as $v.c$ and use $v.p$ to denote the position. Note that the content of each node is uniquely identified by the core component (i.e., the word or sentence). In practice, this can be any unique identifier. For term- and entity-centric analyses, terms can be mapped to additional attributes that are stored in respective lookup tables (e.g., named entity types). We represent these optional attributes with the same component notation. An important attribute in this context is the type of a node, denoted as $v.type$, which classifies it into a document, sentence, or term. Other useful attributes include $v.ne$, the named entity type of a term. For an example, see Figure 1 (right).

3.4.4 Node Equivalences. To compare graph nodes, we introduce node equivalences. We say that two nodes $v,w\in\mathcal{V}$ are equal and write $v=w$ iff their two primary components are identical. That is,

$$v=w:\Leftrightarrow v.c=w.c\wedge v.p=w.p \tag{1}$$

Intuitively, v and w are identical if they share both content and relative position value. Since terms, sentences, and documents are uniquely identified by the core c, we also consider approximate equivalence $\approx$ if only the occurrence position deviates. Thus,

$$v\approx w:\Leftrightarrow v.c=w.c \tag{2}$$

Further relations are viable, such as the less-than and more-than relations $\leq_n$ and $\geq_n$, in which the core is identical and nodes are ordered by their position component. These can be used to induce a partial order on the set of hyperedges, but are not required here.

3.4.5 Hyperedges $\mathcal{E}$. Following the segmentation of documents, we construct hyperedges to represent the document collection around the cooccurrences of terms. Based on the set of nodes $\mathcal{V}$ of the graph, we obtain the set of all possible hyperedges over these nodes as $\Sigma:=2^{Core\times\mathbb{Z}}$, i.e., all sets that can be constructed from all possible nodes. From these, we can identify a subset $\mathcal{E}\subseteq\Sigma$ that represents the input document collection and allows us to define a hypergraph $\mathcal{H}=(\mathcal{V},\mathcal{E})$, in which each edge $e\in\mathcal{E}$ is constructed around a sentence S_e in the document collection. We call S_e the *primary sentence* of e. To model the content and context of sentences, each edge is composed of nodes as defined above. Formally, each edge constitutes a set $e\subset\mathcal{V}$ that contains the terms in and around the primary sentence S_e. Let $w\in\mathbb{N}$ denote the size of a suitable *context window*, measured in sentences. Then $e:=\mathcal{V}^e_{\mathcal{T}}\cup\mathcal{V}^e_{\mathcal{S}}\cup\{(D_e,0)\}$ is a set of terms in S_e and in nearby sentences along with their relative positions, where D_e is the document that contains S_e and

$$\mathcal{V}^e_{\mathcal{T}}:=\{(t,p(t,S_e)):t\in\mathcal{T}\wedge|p(t,S_e)|\leq w\} \tag{3}$$

$$\mathcal{V}^e_{\mathcal{S}}:=\{(S,p(S,S_e)):S\in\mathcal{S}\wedge|p(S,S_e)|\leq w\} \tag{4}$$

Intuitively, each sentence is represented by one hyperedge that contains its contents as well as the contents of surrounding sentences based on the window size w. The parameter w is thus directly related to the size of the context window around a sentence that induces the term cooccurrences. In Section 6, we discuss the practical implications of this aspect for the storage size and show how physical replication of the data can be avoided. For an overview of the model, see Figure 1 (left). Since hyperedges constitute sets of nodes, we write $v\in e$ to denote that edge e is incident on node v.

3.4.6 Edge Relations. In the following, we use relations between hyperedges to simplify the derivation of cooccurrence relations and statistics between terms. Most fundamentally, we define the notion of edge equality based on set semantics. That is, two edges are equal iff node equality as defined above is a bijection between the two edges. Similarly, we say that edges e and f are approximately equivalent and write $e \approx f$ iff a bijection between the two sets can be defined on the basis of approximate node equivalence.

Finally, we say that edge e is contained in edge f and write $e \sqsubseteq f$, if all nodes in e have an approximately equivalent node in f whose absolute position is at most as large as in e. Formally, we have

$$e \sqsubseteq f :\Leftrightarrow \forall v \in e \; \exists \hat{v} \in f : (v \approx \hat{v}) \wedge (|\hat{v}.p| \leq |v.p|). \tag{5}$$

We also refer to e as a *subedge* of f. Intuitively, this notion of edge containment requires that all terms are at least as closely positioned in the containing edge as they are in the subedge.

4 DOCUMENT HYPERGRAPH OPERATORS

Based on the hypergraph document model, we now introduce the base operators that can be used to select and transform the hyperedge representations. We begin by discussing the notation of propositional expressions as a basis for selection operations.

4.1 Propositional Expressions

In analogy to relational algebra, we rely on propositional expressions for the selection of nodes from an edge or of edges from a set of edges. Formally, a propositional expression (called *expression* in the following) can be any syntactically adequate unary formula that maps a node or edge to a truth value. With node attributes being the most discriminative feature of hyperedges, most relevant expressions rely directly on attribute values and are of the form $v.att \, \phi \, x$, where att is some node attribute, x is a value from the domain of this attribute (or a subset thereof), and $\phi \in \{=, \neq, \leq, \geq, <, >, \in\}$. In the following, we consider expressions θ that contain node attributes to be true for an edge if the edge contains at least one node for which the expression is true. That is, for an edge $e \in \mathcal{E}$,

$$\theta(e) = true :\Leftrightarrow \exists v \in e : \theta(v) = true \tag{6}$$

To keep the notation concise, we use an abbreviated notation and write $\theta(e)$ instead of $\exists v \in e : \theta(v)$ when using the expression.

4.1.1 Distance. As a special shorthand, we use the concept of distance d instead of position p where the sign of the position value does not matter. Specifically, we define the distance as

$$d := |v.p| \tag{7}$$

and use it in expressions of the form $d \, \phi \, k$, where $k \in \mathbb{R}$ and ϕ is some valid relation over the real numbers. The expression is true if ϕ holds for d and k. Note that d may not always be a proper distance metric since the identity of indiscernibles is violated when sentence or paragraph distances are used.

4.1.2 Existence. Since nodes may or may not possess a certain attribute due to heterogeneity, an important distinction criterion is the existence of an attribute, regardless of the value. For example, it may be relevant to distinguish between named entities and other terms. Here, we simply denote with $\exists v.att$ an expression that is true if node v has attribute att and false if it does not.

4.2 Closed Operators on Hyperedges

We first introduce a number of operations that are closed on sets of hyperedges, i.e., both the input and the output are sets of hyperedges. Since isolated nodes are edges with one element, all sets of hyperedges can w.l.o.g. be considered to represent hypergraphs.

4.2.1 Set Operators. Three trivial binary operators are the asymmetric set minus $-$, the union $\cup$, and the intersection $\cap$. They conform to their usual semantics. However, note the difference between set operations on two hyperedges (which merge sets of nodes), and on sets of hyperedges (which merge entire graphs).

4.2.2 Selection σ. The selection is defined in analogy to the definition of a subhypergraph and equates to the selection of all hyperedges from an input set that satisfy some selection expression. For example, a subset of hyperedges can be selected based on certain nodes that these edges contain, or on attributes of those contained nodes. Formally, for some expression θ, we define the selection $\sigma_\theta : 2^\Sigma \rightarrow 2^\Sigma$. Thus, for a set of input edges $E \subseteq \mathcal{E}$, let

$$\sigma_\theta(E) := \{e \in E : \theta(e)\}. \tag{8}$$

In terms of relational algebra, if we were to relate hyperedges to tuples, then the selection of edges from a set of edges is semantically similar to the selection of tuples from a table.

4.2.3 Projection π. The projection of hyperedges can be defined in analogy to partial hypergraphs, which is to say it handles the removal of nodes from hyperedges based on the provided expression. That is, all nodes that do not satisfy a given condition are removed from the input hyperedges. For example, all nodes of the sentence type or all nodes with a given attribute in an external knowledge base could be removed from the input edges. For the sake of notation, we first define the projection for a single hyperedge and then generalize. Formally, for some expression θ, we define the projection $\pi_\theta : \Sigma \rightarrow \Sigma$. Thus, for a given input edge $e \in \mathcal{E}$, let

$$\pi_\theta(e) := \{v \in e : \theta(v)\}. \tag{9}$$

On this basis, for a set of edges $E \subseteq \mathcal{E}$, we can define the more general projection function $\pi_\theta : 2^\Sigma \rightarrow 2^\Sigma$ as

$$\pi_\theta(E) := \{\pi_\theta(e) : e \in E\}. \tag{10}$$

In terms of relational algebra, if we were to relate nodes of hyperedges to attributes of a tuple in a table, then the projection is defined similarly. However, note that edges do not necessarily need to contain nodes with every attribute that occurs in θ.

For simplicity, we use three shorthand notations for frequently used projections. Specifically, we use π_{term}, π_{sen}, and π_{doc} to project hyperedges to the most common structural components of text by removing all nodes from all input edges that are not of the the type term, sentence, or document, respectively.

4.2.4 Reduction r. An important aspect of a hypergraph model is its capability to represent higher-order cooccurrences. However, many existing models use dyadic graph representations, so the inclusion of an operator that transforms hypergraphs into dyadic graphs by creating dyadic edges between all nodes in a hyperedge is required. Note that the resulting list of edges can still be represented as a 2-uniform hypergraph (i.e., all edges have a cardinality of two). We refer to this operation as *reduction* (sometimes also called a

clique expansion in the literature), which we first define for a single hyperedge e as a function $r : \Sigma \to 2^{\Sigma}$ such that

$$r(e) := \{\{v, w\} : v \neq w \wedge v, w \in e\}. \tag{11}$$

Note that we specifically exclude self edges, which serve no purpose in this context. Based on this, we then obtain a reduction function $r : 2^{\Sigma} \to 2^{\Sigma}$ for sets of hyperedges E as

$$r(E) := \bigcup_{e \in E} r(e). \tag{12}$$

For simplicity, we use $r_{\neq}$ as shorthand notation for a reduction in which edges in the resulting dyadic graph are discarded if they connect nodes of the same type (term, sentence, or document). Similarly, we use $r_{=}$ to denote a resulting dyadic graph in which only edges between nodes of the same type are retained.

More generally, we may also consider an operator that extracts k-uniform hyperedges as subsets of fixed size k. If we denote with $[A]^k$ the set of all subsets of A of size k, then r^k is defined as

$$r^k(e) := [e]^k \qquad \text{and} \qquad r^k(E) := \bigcup_{e \in E} r^k(e). \tag{13}$$

The dyadic reduction is then a special case of r^k for $k = 2$ and we write r instead of r^2 where it is clear from context. Similar to the case above, we use $r^k_{\neq}$ and $r^k_{=}$ as shorthand for output graphs in which all nodes of an edge are of different or identical type, respectively. In Section 5, we give examples of applications in which these hypergraph-to-hypergraph operators are of practical use.

Finally, we observe that lower-order edges in the resulting graphs are not necessarily distinct. Due to set semantics, these duplicate edges are lost after the reduction unless multi-sets are used. Alternatively, an aggregation weighting function ω can be used to assign a weight to the resulting aggregated edges (which is, to the best of our knowledge, how this is predominantly handled in practical applications). As a simple example, the total number of all such edges could be assigned as a weight, which surmounts to counting the duplicates. In the dyadic case, edges between terms in the resulting graph would be assigned their cooccurrence count. More complex weight functions are of course possible. Formally, we assume a function r_m that behaves equivalently to r but uses multiset semantics. For a multiset M of edges, let $\{\!\{M\}\!\}_e := \{e' \in M : e \approx e'\}$ denote the subset of edges that are approximately equivalent to a given edge e. We can then regard the reduction with an aggregation weight function as a family of functions $r_\omega : \Sigma \to \mathbb{R}$ such that

$$r_\omega(E) := \{(e, \omega(\{\!\{r_m(E)\}\!\}_e)) : e \in r_m(E)\}. \tag{14}$$

Thus, any conceivable function ω that takes a set of edges between a fixed set of nodes and computes a weight for the aggregated edge can be used in this context.

4.2.5 Join $\bowtie$. As a final operator, we introduce the join of hyperedges, which is inspired by the concept of the join in relational algebra, but is semantically distinct. Unlike edges in a dyadic graph, hyperedges can be extended to include additional nodes. Thus, we consider the extension of edges with nodes from other edges that overlap on some subset of nodes. In terms of a dyadic graph, this translates to the construction of growing paths from starting nodes, or to growing clusters. From a retrieval perspective, this allows the (context-sensitive) expansion of relevant cooccurrences around some set of terms. Conceptually, we distinguish between joins around a specified subedge and joins around shared subedges.

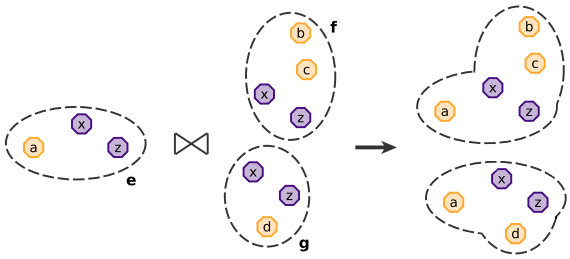

Figure 2: Example join $\{e\} \bowtie \{f, g\}$ on the subedge $\varepsilon = \{x, z\}$.

The first case we consider is the join on a specified subedge ε, which we call the ε-join $\bowtie_\varepsilon$. For two sets of hyperedges $E, F \in \mathcal{E}$, we extend each edge in E that contains the given subedge with each edge in F that also contains the given subedge. Formally, we let

$$E \bowtie_\varepsilon F := \{e \cup f : e \in E \wedge f \in F \wedge \varepsilon \sqsubseteq e \wedge \varepsilon \sqsubseteq f\}. \tag{15}$$

In an application scenario, this join allows, for example, the identification and exploration of common or distinct cooccurrences of terms in the graph with a set of query terms.

Expanding on this specific join on fixed subedges, we define the more general j-join $\bowtie_j$, which joins all edges that contain any shared subedge of size j or larger. Thus, for some $j \geq 1$, let

$$\begin{aligned} E \bowtie_j F := \{&e \cup f : e \in E \wedge f \in F \wedge \\ &(\exists \varepsilon : |\varepsilon| \geq j \wedge \varepsilon \sqsubseteq e \wedge \varepsilon \sqsubseteq f)\}. \end{aligned} \tag{16}$$

In contrast to the ε-join, the j-join does not expand all edges in a common direction but rather expands each edge in all (possibly distinct) suitable directions. Both joins rely on the relative position and thus allow a restriction of cooccurrences to a desired proximity level. For a schematic overview, see Figure 2.

Note that for hyperedges resulting from a join operation, position values are not necessarily well defined. The edge join operation is thus powerful but, similar to a join in relational algebra, not every possible join result is semantically meaningful. In Section 5, we discuss how edge joins can be used for path and context explorations. In addition to the two operators defined above, further joins such as a general θ-join on arbitrary expressions are conceivable.

4.3 Closure under Operators

Most retrieval operations on the hypergraph structure require the combination (or chaining) of multiple operators on the set of input edges to obtain the desired result. Thus, it is important that the set of possible sets of hyperedges 2^{Σ} is closed under the operators as introduced above. In the following, we briefly discuss and prove this property for sets of hyperedges.

Lemma 4.1 (Closure under set operations). *The set of possible sets of hyperedges 2^{Σ} is closed under the set operators set minus $-$, union $\cup$, and intersection $\cap$.*

Proof. Since power sets are closed under the basic set operators and 2^{Σ} is a power set, it follows that 2^{Σ} must be closed under the minus, union, and intersection operators. □

Lemma 4.2 (Closure under selection). *The set of possible sets of hyperedges 2^{Σ} is closed under the selection operator σ.*

Proof. Given a subset of edges $E \subseteq \Sigma$ and some expression θ, it must hold that $\sigma_\theta(E) \subseteq E$ and $\sigma_\theta(E) \subseteq \Sigma$ by transitivity of the set containment. Therefore, the set of possible sets of hyperedges is closed under selection. □

Lemma 4.3 (Closure under projection). *The set of possible sets of hyperedges 2^Σ is closed under the projection operator π.*

Proof. Given a subset of edges $E \subseteq \Sigma$ and some expression θ, let $\pi_\theta(E)$ denote the set of edges that is obtained by projecting E according to expression θ. Then for all edges $e \in \pi_\theta(E)$, there exist an edge $e' \in E$ such that $e \subseteq e'$. Since $e' \in \Sigma$ and Σ is also a power set that is closed under the set minus operation, it must hold that $e \in \Sigma$. Thus, $\pi_\theta(E) \subseteq \Sigma$, meaning that 2^Σ is closed under the projection operator. □

Lemma 4.4 (Closure under reduction). *The set of possible sets of hyperedges 2^Σ is closed under the reduction operator r.*

We observe that the reduction operator effectively creates the set of all k-uniform subedges for each hyperedge in the input set. Therefore, we omit the proof of closure since it is analogous to the proof provided above for the projection operator.

Lemma 4.5 (Closure under join). *The set of possible sets of hyperedges 2^Σ is closed under the join operators $\bowtie_\varepsilon$ and $\bowtie_j$.*

Proof. To show the closure of the join operation, we can consider both the ε-join and the j-join simultaneously. W.l.o.g. let $f = \{e_1\} \bowtie \{e_2\}$ denote the single hyperedge that results from the join of edges $e_1 \in \Sigma$ and $e_2 \in \Sigma$ on some common subedge. Then, $f = e_1 \cup e_2$, from which directly follows that $f \in \Sigma$ since Σ is closed under union. Given that the above observation holds for any two edges, it must also hold for the join $F = E_1 \bowtie E_2$ of any two sets of edges $E_1 \subseteq \Sigma$ and $E_2 \subseteq \Sigma$ that $F \subseteq \Sigma$. Thus, 2^Σ is closed under the join operator. □

4.4 Non-closed Operators

In addition to the five types of basic operations defined in the previous section, some applications may require additional operators that map sets of hyperedges to scalar values instead of other sets of hyperedges. A prominent example that we use in the following is the counting operator that returns the number of edges in a graph. Formally, we define it as a function $count : 2^\Sigma \to \mathbb{N}$ such that for an edge set $E \in 2^\Sigma$, we obtain $count(E) := |E|$. While the operator is trivial, it also forms the basis for the large collection of statistical methods that rely on counts of term cooccurrences.

5 HYPERGRAPH MODEL APPLICATIONS

The literature knows a multitude of methods that rely on the extraction of term cooccurrences for tasks so diverse as exploratory search, event detection, or summarization. To highlight the versatility of the hypergraph model, we show how the model can be used to reproduce and support existing IR techniques for a number of typical and essential applications. An exhaustive coverage of techniques is beyond the scope of this (or any single) paper, so we focus on a selection that employs diverse operators or reproduces well known baselines. We show how the model emulates and supports a wide range of approaches with the operators defined in Section 4.

5.1 Exploratory Search

We begin with a number of cooccurrence query operations that serve as examples of initial investigations into a document collection. Based on a query term t, a basic search operation is realized by the selection of terms that cooccur with the given term in a window of size at most k, which we can represent as

$$\mathcal{E}_{cooc}(t,k) := \pi_{v.c\neq t}(\pi_{d\leq k}(\sigma_{v.c=t}(\pi_{term}(\mathcal{E})))) \tag{17}$$

By adjusting the projections, we can also retrieve sentences or documents as source information of the cooccurrence instances. Going beyond single-term queries, if we are interested in sets T of terms as query input, we can combine the above operations for individual terms by intersection to obtain

$$\mathcal{E}_{cooc}(T,k) := \bigcap_{t\in T} \mathcal{E}_{cooc}(t,k). \tag{18}$$

Similar to the case above, retrieving source information may support additional document or sentence retrieval tasks. For further exploration or to obtain count statistics (e.g., for rankings), both of the above results can also be reduced to a dyadic graph.

To explore more complex cooccurrence patterns, joins are helpful. Consider, for example, a query in which we aim to extract location mentions (i.e., toponyms) at which a specified person is mentioned together with groups of other persons (i.e., that person's meeting places). For a given term t_p of named entity type person, and a minimum group size j, we can formulate the operation as

$$\begin{aligned} E_1 &:= \sigma_{v.c=t_p}(\pi_{v.ne=per}(\mathcal{E})) \\ E_2 &:= \pi_{v.ne\in\{per,loc\}}(\mathcal{E}) \\ \mathcal{E}_{places}(t_p,j) &:= \pi_{v.ne=loc}(E_1 \bowtie_j E_2) \end{aligned} \tag{19}$$

that returns all such place mentions. If we are instead interested in places where a given person was mentioned with a specific group of other persons, an ε-join could be used.

5.2 Vector Space Model

The vector space model is a classic representative of document models and is based on the bag-of-words representation for sentences, which is easily emulated through hyperedges. Established methods include numerous variations of *tf-idf* or the BM25 metric [35] that are based on the term count statistics term frequency *tf* (the frequency of a term in a document) and document frequency *df* (the number of documents in which the term occurs) of documents or sections of documents. The frequency of term t in document D can be obtained from the hypergraph model as

$$tf(t,D) := count(\sigma_{v.c=t}(\pi_{d=0}(\sigma_{v.c=D}(\mathcal{E})))) \tag{20}$$

by counting the sentences in the document that contain the term. Similarly, the document frequency is given by

$$df(t) := count(\pi_{doc}(\sigma_{v.c=t}(\mathcal{E}))). \tag{21}$$

Note that these are the commonly used definitions. However, other variations of these measures can be formulated analogously, which shows that the proposed model includes these baselines. Obviously, the computation of such metrics can be combined with subsequent explorations within the same framework.

5.3 Graph-based Summarization

Automatic text summarization is a large area of research in which a multitude of methods employ dyadic graph representations of the documents. Here, we consider LexRank [15] as a well known example for such an approach. Common to the majority of summarization methods is the representation of sentences as nodes of a graph. Nodes are then connected by edges that encode some form of sentence similarity. Subsequent steps extract representative sentences from this graph, for example through centrality computations or random walks. Based on a hypergraph representation of the input documents, a sentence graph $\mathcal{G}_{sen}$ can be generated as

$$\mathcal{G}_{sen} := r_{=}\left(\bigcup \pi_{d=0}(\pi_{sen}(\mathcal{E}))\right) \tag{22}$$

such that a sentence similarity function $sim : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ allows the derivation of sentence similarity from the context of a sentence S, i.e., $\pi_{term}(\pi_{d=0}(\sigma_{v.c=S}(\mathcal{E})))$. Subsequent graph centrality computations can be performed directly on the reduced graph (e.g., based on the vector space model in the case of LexRank).

More recent summarization approaches that rely directly on a hypergraph representation of sentences [4] can be replicated even more easily. In these cases, a projection $\pi_{d=0}$ to the primary sentence level along with a set union are sufficient to represent the underlying data in the model of the summarization approach.

5.4 Event Extraction and Detection

Based on the definition of an event as *something that happens at a specific date and location and involves an actor* [1], event extraction is ultimately aimed at the efficient detection of actor-location-date triples (or their subsets in the case of partial mentions), along with a suitable context. Similarly, much of event detection is based on tracking evolving statistics of entity or term cooccurrences, which are easily extractable from the hypergraph model. For example, extracting actor-location-date triples is equivalent to the reduction of edges to a 3-uniform hypergraph in which edges constitute triples of entity nodes with distinct type. Thus, for an occurrence context window of size k, this extraction can be formulated as

$$\mathcal{E}_{triples}(k) = r^{3}_{\neq}(\sigma_{term}(\pi_{d\leq k}(\pi_{v.ne\in\{loc,per,dat\}}(\mathcal{E})))). \tag{23}$$

A weighting function can be used to extract counts or relevance scores for the occurrences. For more involved approaches that subsequently perform linguistic analyses on the sentence level, retrieving provenance information for the mentions then constitutes an inclusion of sentence nodes with distance zero in the expression. Thus, entity triples and occurrence statistics are easily extracted and serve as seeds for any more specialized approach.

5.5 Query Hypergraph Support

Query hypergraphs were introduced by Bendersky and Croft to model higher-order term dependencies in queries [5]. In principle, they are based on the same intuition as our hypergraph document representation, but are specifically limited to modeling only the dependency of query terms (or more complex query concepts) as hypergraphs, while the document representation itself is not considered. Naturally, a hypergraph formalization of the documents directly enables the efficient processing of similarly structured hypergraph queries and constitutes the logical next step. Specifically, query hypergraphs as introduced by Bendersky and Croft model hyperedges between terms and documents to facilitate document ranking and retrieval. Thus, retrieving relevant edges from the document collection is enabled implicitly by use of the edge containment relation $\sqsubseteq$ in our model (see Section 3.4.6).

By modeling queries as a set of query concepts $\kappa \in \mathcal{K}^{Q}$, query hypergraphs are constructed from *local edges* and *global edges*. Local edges have the formal structure $e_l = \{\kappa, D\}$ and simply link each concept to the document D. As the authors observe themselves, these edges are not hyperedges but dyadic edges and thus structurally similar to traditional bag-of-word representations. As a result, relevance computations can be processed on the hypergraph document representation according to Equations 20 and 21, albeit by using count statistics that are different from *count* where necessary. In contrast, global edges link the entire set of query concepts to the document and are formalized as $e_g = \mathcal{K}^{Q} \cup \{D\}$. To retrieve documents that are perfect matches, we can thus construct a global query edge e_q in our notation as $e_q := \{(v, 0) \mid v \in e_g\}$ and retrieve the set of matching hyperedges $\mathcal{E}_{global}(e_q) = \{e \in \mathcal{E} \mid e \sqsubseteq e_q\}$. Obviously, deriving document rankings then also requires the retrieval of partial matches, a process which can be formalized based on the hypergraph operators as

$$\mathcal{E}_{global}(\mathcal{K}^{Q}) = \pi_{v.c\in\mathcal{K}^{Q}\vee v.type=doc}(\sigma_{v.c\in\mathcal{K}^{Q}}(\mathcal{E})). \tag{24}$$

Note that this formulation does not restrict the cooccurrences of terms to the sentence level, since cross-sentence adjacencies are required for proximity-structure query hyperedges [5]. Intuitively, we are retrieving hyperedges from the hypergraph representation that (partially) match the query hyperedges and then use them for scoring. Adding further query restrictions such as maximum cooccurrence windows through chained projection or selection operations is then a trivial matter.

5.6 Implicit Entity Networks

Implicit entity networks have recently been proposed as flexible data representations for diverse IR tasks related to entity cooccurrences. As an example of such a network we use the LOAD model by Spitz and Gertz [39, 40] since it considers cooccurrences beyond sentence boundaries. Obviously, networks based on term distances are equally viable as discussed in Section 3.4. The implicit network model is based on a dyadic entity graph for a context window size of k sentences that can be represented in the hypergraph model as

$$\mathcal{G}_{LOAD}(k) = r_{\omega,\neq}(\pi_{d=0}(\mathcal{E}) \cup \pi_{0\leq p\leq k}(\pi_{\exists v.ne}(\mathcal{E}))). \tag{25}$$

Note that we ignore negative positions to prevent counting cooccurrences between sentences twice. The edge weights ω used in LOAD are then recreated by using the aggregation function

$$\omega(E) := \sum_{(v,w)\in E} \exp(-|\max\{v.p - w.p\} - \min\{v.p - w.p\}|). \tag{26}$$

The resulting dyadic graph is equivalent to the implicit network and supports all extraction and ranking methods proposed for such a network. Similar approaches that do not use edge weights but rely on discrete edge attributes for the extraction of information networks are equally viable, albeit longer in a formal representation.

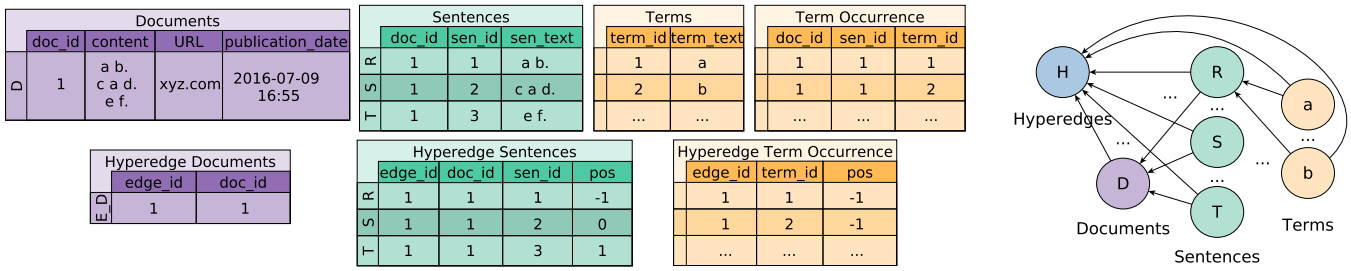

Figure 3: Conceptual representation of the schema in the PostgreSQL relational model (left) and the graph schematics in Neo4j (right). The data used in this example corresponds to the example document D with sentences S, T, and R in Figure 1.

6 IMPLEMENTATION AND EVALUATION

So far, we have given a description of the model and its operators, and demonstrated its applicability to wide range of typical tasks in IR, which puts a full evaluation of all aspects well outside the scope of this paper. However, we provide two initial implementations of the proposed model to investigate the impact of design decisions on the performance of such a system and demonstrate its viability.

For our implementation of the model, three aspects are of primary concern, namely the choice of the underlying database systems, the degree to which hyperedges need to be materialized, and the amount of cooccurrence information that is stored in the system. In the following, we evaluate implementations of the model with a focus on these three aspects of *materialization*, *database system*, and *data*. We conduct all experiments on a commodity hardware system, using an Intel Core i7-7700 CPU, 32 GB RAM, and a 1 TB HDD. The code for our experiments is available online[1].

6.1 Hyperedge Materialization

The efficient storage of dyadic graphs is already no simple task, and arbitrary hypergraphs are justifiably known for inducing an even more daunting complexity. However, in the case of our document representation, hypergraphs primarily serve as a formalization of the inherent structure of natural language, which puts bounds on the size of hyperedges in practice. In particular, applications typically consider term cooccurrences within a fixed window, meaning that the sequential structure of sentences within a document matters. This raises the question whether the hyperedges have to be represented explicitly, or if they can be implicitly generated at query time with limited overhead to the storage requirements.

6.1.1 Explicit representation. As the most direct approach, we consider a naive baseline implementation in which all hyperedges are precomputed and materialized in the database. While we expect the replication of content across adjacent hyperedges to create a significant increase in the amount of required storage space in comparison to storing only the documents, this approach may provide solid query performance for smaller document collections or small window sizes. We refer to this as the *explicit* representation.

6.1.2 Implicit representation. The replication of data that is stored in the explicit representation scales linearly with the considered window size for cooccurrences and is likely excessive for large window sizes. However, recall that the replication of nodes in the hyperedges of adjacent sentences is purely conceptual. In practice, the sentences of the documents are likely stored sequentially or within close proximity in memory. In contrast to the explicit storage of hyperedges, we thus also consider an *implicit* representation in which the hyperedges are generated from the stored sentences at query time. While this avoids the storage overhead of explicitly storing the edges, it may increase the processing time for queries.

6.1.3 Dyadic representation. Since many IR applications are graph-based, we also consider a physical representation of the dyadic graph structure of cooccurrences. This is conceptually similar to using the reduction operator r on a hypergraph representation (see Section 4), but obviously prohibits the application of the more involved hyperedge join operators. However, it may be sufficient for applications that only require a dyadic graph structure. Since this model represents each hyperedge by $|E| \cdot (|E| - 1)/2$ dyadic edges, the required storage space grows quadratically with the size of hyperedges. We refer to this as the *dyadic* representation.

6.2 Database Systems

The design of the hypergraph operators and their applications in Sections 4 and 5 indicate that a translation of the hypergraph into a relational database is an intuitive modeling decision. Thus, we implement the model in PostgreSQL as a representative of relational database systems. On the other hand, since the model is inherently graph-based, graph databases might potentially be considered as a suitable alternative due to their native support of graphs and their optimized performance for graph operations. As a representative of graph databases, we use Neo4j. For an overview of the baseline schema that we use to represent data in these systems, see Figure 3.

6.2.1 PostgreSQL. As relational database system, we use PostgreSQL 11.1. Our baseline data model in the relational implementation represents the content of the document collection in four tables, three of which store the hierarchical containment information of *documents*, *sentences*, and *terms*, while the fourth contains the *term occurrence* information. Additional tables serve to store the hyperedge information. For the explicit representation, we furthermore store the respective *hyperedge document, hyperedge sentences, and hyperedge term occurrences*. Each of these three tables contains the edge identifier, as well as a unique identifier for each individual occurrence (i.e., *doc_id* for documents, *sen_id* and *pos* within the hyperedge for sentences, and *term_id* and *pos* for terms). The separation of tables with respect to the hierarchical structure of the text allows the efficient execution of type-projection queries (π_{term}, π_{sen}, π_{doc}), and minimizes the size of indices and intermediate

[1]Code available at https://github.com/dennlinger/hypergraph-document-store

table representations, while maintaining a homogeneous data definition. For the implicit representation, no additional tables are used beyond the baseline since hyperedges are generated dynamically. For the dyadic representation, the *hyperedge term occurrence* table is replaced by a table for dyadic edges in which we store the *edge_id* and the two incident nodes of the edge.

To boost query performance, we add primary key indices over the ID columns of every table (for example, the primary key for *sentences* consists of *doc_id* and *sen_id*). Secondary indices are available for all *term_id* columns, including the *hyperedge* table. Aside from regular B-trees, we use a general inverted index for the *term_text*. Clustering of tuples for all tables is according to the primary key index (alternatives did not improve query performance).

6.2.2 Neo4j. To implement the model, we use Neo4j Community Edition 3.5.11, which is designed to natively handle graph data, but does not include direct hypergraph support. Nevertheless, modeling in Neo4j still results in a very interpretable schema, as shown in Figure 3 (right). We store the baseline data model in separate node types that include unique identifiers and respective meta-information, but split the containment relation into three separate relationships *term_in_sentence, sentence_in_document* and *term_in_document*. Hyperedges are modeled via node types, where hyperedge-specific information (such as the relative position) is modeled as a property of the edges between nodes of type *hyperedge* and nodes of type *term* or *sentence*. The dyadic representation is modeled as a self-relation on term occurrences, which contains a unique identifier (*edge_id* and *pos*) for each edge.

To ensure comparable performance to PostgreSQL, indices were added to identifier columns across all node types (*document, sentence, term, hyperedge*), and further on the *term_text* property. Additional indices showed no improvement during evaluation.

6.2.3 Query Generation. In the following, we briefly introduce our approach to translating the operators from Section 4 into actual queries based on our implemented database schemas.

For PostgreSQL, the translation of hypergraph operators into queries is trivial by design due to the semantic similarity between the hypergraph model and relational algebra, which thus allows for a seamless translation. To optimize the generated query plans, we experimented with basic CTE fencing.

In contrast, the translation of hypergraph operations to queries in Neo4j's query language Cypher is less direct since it does not rely on relational algebra. However, our model formalizations with an explicit *hyperedge* node type allows us to express the set operators as well as the atomic operations σ and π in a single MATCH expression, which enables the easy generation of any general operator.

More complex queries arise only if intermediate hyperedge results need to be generated ad-hoc (e.g., in the implicit model), but can be modeled within a single subquery in both implementations.

6.3 Data

As evaluation data, we utilize news articles as a typical use case of entity-annotated Web documents. We use a set of English news articles that are annotated for parts of speech and named entities [41]. Named entity mentions in the documents are disambiguated and linked to Wikidata as an external knowledge base.

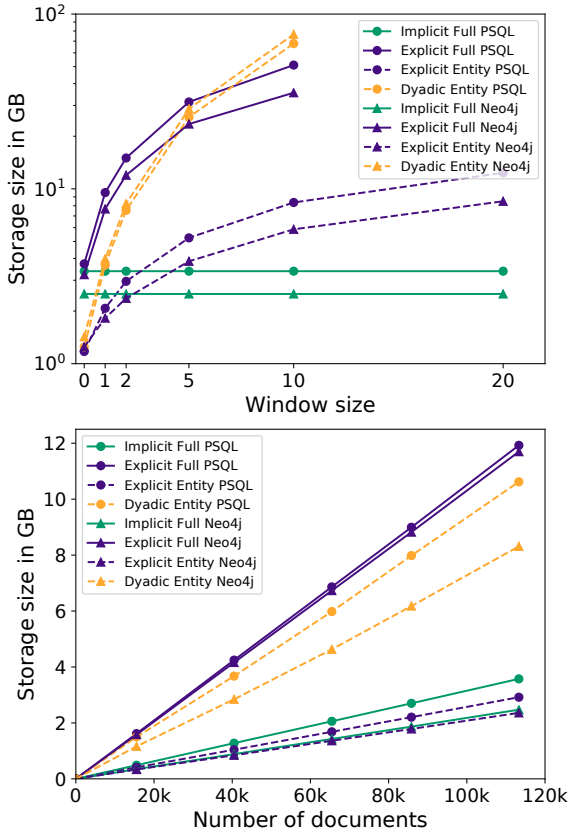

Figure 4: Storage space requirements for all models in PostgreSQL and Neo4j for varying sizes of the context window w (top), and for varying numbers of documents in the collection and a fixed size of the context window $w = 2$ (bottom).

6.3.1 Full data set. The collection of news articles consists of 113,312 distinct documents containing 2,746,875 sentences. After the removal of stopwords, there are 31,631,317 distinct occurrences of 390,486 terms. Furthermore, there are 122,153 distinct named entities that have 3,121,492 combined overall occurrences. In the following, we refer to this as the *full* data set.

6.3.2 Entity subset. While many applications require the retention of cooccurrence information for all terms, named entities are of special interest in many IR applications, such as the extraction of entity relations. Due to the sparsity of entity mentions, this subset differs in not just its size but in structural properties (such as a reduced size of hyperedges for constant window size). To assess the performance for entity-centric tasks, we include a subset of the data in which all non-entity terms are removed, and refer to it as the *entity* data set. This shrinks the number of hyperedge-term tuples by approximately 90%, but retains the relational data structure.

6.4 Evaluation: Storage Space Requirements

By combining the three degrees of freedom for implementing the model discussed above, we obtain 12 possible model configurations *(explicit, implicit, dyadic)* × *(full data, entity data)* × *(PSQL, Neo4j)* to evaluate. Of these 12, two are redundant (the implicit representation has a constant storage size that is identical for both data sets) and

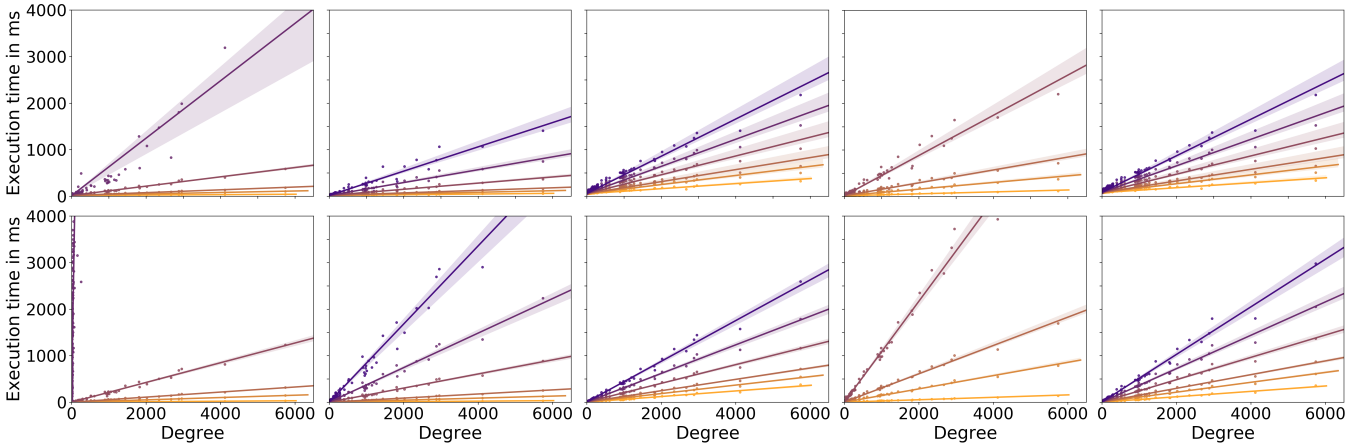

Figure 5: Query performance of all models for 2,000 randomly sampled entities in PostgreSQL (top row) and Neo4j (bottom row). The evaluated window sizes are 20 (●), 10 (●), 5 (●), 2 (●), 1 (●), and 0 (●). Shaded areas represent 95% confidence bands.

two could not be implemented (the memory consumption of the dyadic representations for the full data is prohibitive), which leaves eight combinations to consider. Storage requirements in PostgreSQL are measured as tables and corresponding indices. In Neo4j, we report the size of the `graph.db` folder for the equivalent export.

In Figure 4 (top), we show the storage size requirements for varying window sizes w (measured in sentences to either side of a primary sentence), and the storage size over the number of stored documents in Figure 4 (bottom). We were unable to obtain the values for the explicit representation on the full data for window sizes beyond $w = 10$ due to insufficient memory, indicating that the explicit representation is only suitable for small window sizes. The dyadic representation is capped at $w = 10$ for the same reason, which is a strong indication that the storage efficiency of a hypergraph representation is preferable over the dyadic format.

We find that the memory footprint of implicit models scales best since they effectively ignore the window size and materialize edges only at query time. The comparatively large storage size for small windows is the result of a more expensive representation of term occurrences in the schema. For explicitly stored hypergraphs, all representations scale poorly with the window size due to the increasingly large overlap of hyperedges. We observe that the increase in memory consumption slows down for large window sizes that exceed the overall size of shorter documents (for $w = 20$, the context window is already larger than 85.6% of documents).

Surprisingly, the comparison between implementations in PostgreSQL and Neo4j reveals that Neo4j is at a minor advantage. While Neo4j has a slightly higher cost for representing nodes, it also enables an overall cheaper implementation of edges and thus performs comparatively better for larger window sizes in which nodes are dominated by the entries for hyperedges. In PostgreSQL, the major source of memory consumption is the creation of indices.

6.5 Evaluation: Query Performance

To provide a first impression of the differences in performance between the different model configurations, we focus on the extraction of term cooccurrences $\mathcal{E}_{cooc}$ as a core concept. Thus, our evaluation metric differs from regular graph benchmarks in that it is specific to the IR-related tasks that the hypergraph model is designed to support. We generate a set of queries by randomly sampling 2,000 entities from the set of term nodes, whose occurrence counts range from 1 to 5,800 (we refer to this as the degree of the entities). Due to the underlying Zipfian distribution of terms in documents, the majority of these entities have a relatively low degree. To determine the query performance for each model and window size, we record and average the query execution time for retrieving all cooccurrences of a given query entity as returned by `EXPLAIN ANALYZE` (Postgres) and `PROFILE` (Neo4j) over five iterations (after an initial iteration for cache warmup to ensure a fair comparison).

In Figure 5, we show the query performance of all model constellations over varying window sizes. In a comparison between Neo4j and PostgreSQL, the latter has a better overall performance, while the differences between the model implementations are consistent across database systems. The dyadic representation in Neo4j suffers from the extremely large edge tables for larger window sizes and no longer benefits fully from caching effects, which is detrimental to its performance. We observe a similar effect for larger window sizes in the explicit representations. PostgreSQL alleviates this problem due to its ability of caching only the corresponding index in memory, which results in a better scaling performance for large hyperedges.

Most importantly, the implicit models offer a competitive runtime even for large window sizes, and outperform the explicit representation in Neo4j, even on the reduced *entity* dataset. For small window sizes, the implicit models incur the expected overhead of computing the hyperedges on the fly, but provide a surprisingly effective performance, especially if the enormous reduction in storage space requirements is considered. For window sizes above $w = 2$, the implicit representation is always the superior choice, regardless of the database system, which affirms the primary motivation behind the use of the hypergraph model over dyadic graphs.

7 SUMMARY AND CONCLUSION

In this paper, we introduced a hypergraph model for representing and querying term cooccurrences in large document collections, not

as a replacement or improvement for any *single* IR approach, but rather as a versatile and unified framework that natively supports a wide *variety* of tasks in text mining and information extraction. By utilizing hyperedges as sets of heterogeneous term, sentence, and document nodes, the graph enables the inclusion of external knowledge bases and thus bridges the gap between structured information and unstructured text data. To formalize queries to this model, we introduced a set of edge operators that allow the representation of numerous fundamental information retrieval methods in one universal notation. Based on these operators, we discussed a range of example applications in which they enable the retrieval and extraction of information from the underlying document collection both through exploratory search and established IR approaches.

Our empirical evaluation of the hypergraph model shows that it is not only competitive with existing dyadic graph representations, but that it is preferable due to its reduced storage space requirements and better query performance for longer cooccurence distances. While the implicit representation of hyperedges is simultaneously efficient in terms of memory usage and query speed, it is also more versatile in its support of downstream applications. Thus, our findings indicate that it is entirely possible to benefit from the expressiveness of formally modeling term cooccurrences in large document collections as overlapping hyperedges, without the drawback of having to physically replicate the data.

Ongoing work. The natural next step is a native implementation of the hypergraph model in a suitable database system with full support for all operators in lieu of relying on high-level queries.

REFERENCES

[1] James Allan. 2002. *Topic Detection and Tracking: Event-Based Information Organization.* Vol. 12. Springer Science & Business Media.

[2] Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. 2005. Query Expansion Using Term Relationships in Language Models for Information Retrieval. In *CIKM*.

[3] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2014. HG-Rank: A Hypergraph-based Keyphrase Extraction for Short Documents in Dynamic Genre. In *WWW Companion*.

[4] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2014. Multi-document Hyperedge-based Ranking for Text Summarization. In *CIKM*.

[5] Michael Bendersky and W. Bruce Croft. 2012. Modeling Higher-order Term Dependencies in Information Retrieval Using Query Hypergraphs. In *SIGIR*.

[6] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A Neural Probabilistic Language Model. In *NIPS*.

[7] Claude Berge. 1973. *Graphs and Hypergraphs.* North-Holland Publishing.

[8] Claude Berge. 1984. *Hypergraphs: Combinatorics of Finite Sets.* Vol. 45. Elsevier.

[9] Roi Blanco and Christina Lioma. 2012. Graph-based Term Weighting for Information Retrieval. *Inf. Retr.* 15, 1 (2012), 54–92.

[10] Monojit Choudhury, Diptesh Chatterjee, and Animesh Mukherjee. 2010. Global Topology of Word Co-occurrence Networks: Beyond the Two-regime Power-law. In *COLING*.

[11] Edgar F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (June 1970), 377–387.

[12] Ido Dagan, Shaul Marcus, and Shaul Markovitch. 1993. Contextual Word Similarity and Estimation from Sparse Data. In *ACL*.

[13] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity Query Feature Expansion Using Knowledge Base Links. In *SIGIR*.

[14] Sourav Dutta and Gerhard Weikum. 2015. Cross-Document Co-Reference Resolution using Sample-Based Clustering with Knowledge Enrichment. *TACL* 3 (2015), 15–28.

[15] Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *J. Artif. Intell. Res.* 22 (2004), 457–479.

[16] Ernesto Estrada and Juan A. Rodríguez-Velázquez. 2006. Subgraph Centrality and Clustering in Complex Hyper-networks. *Physica A* 364 (2006), 581 – 594.

[17] Stefan Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* Ph.D. Dissertation. University of Stuttgart, Germany.

[18] John Rupert Firth. 1957. *Papers in Linguistics, 1934-1951.* Oxford University Press, London.

[19] Benjamin Heintz and Abhishek Chandra. 2014. Beyond Graphs: Toward Scalable Hypergraph Analysis Systems. *SIGMETRICS Perform. Eval. Rev.* 41, 4 (2014), 94–97.

[20] Benjamin Heintz, Shivangi Singh, Corey Tesdahl, and Abhishek Chandra. 2016. *MESH: A Flexible Distributed Hypergraph Processing System.* Technical Report. Department of Computer Science and Engineering, University of Minnesota.

[21] Jin Huang, Rui Zhang, and Jeffrey Xu Yu. 2015. Scalable Hypergraph Learning and Processing. In *ICDM*.

[22] Ramon Ferrer i Cancho and Richard V. Solé. 2001. The Small World of Human Language. *Proc. Royal Soc. B* 268, 1482 (2001), 2261–2265.

[23] Adam Jatowt, Ching-man Au Yeung, and Katsumi Tanaka. 2013. Estimating Document Focus Time. In *CIKM*.

[24] Komal Kapoor, Dhruv Sharma, and Jaideep Srivastava. 2013. Weighted Node Degree Centrality for Hypergraphs. In *Network Science Workshop*.

[25] Wei Liang. 2017. Spectra of English Evolving Word Co-occurrence Networks. *Physica A* 468 (2017), 802 – 808.

[26] Wei Liang, Yuming Shi, K Tse Chi, Jing Liu, Yanli Wang, and Xunqiang Cui. 2009. Comparison of Co-occurrence Networks of the Chinese and English Languages. *Physica A* 388, 23 (2009), 4901–4909.

[27] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to Find Exemplar Terms for Keyphrase Extraction. In *EMNLP*.

[28] Christopher D. Manning, Prabhakar Raghavan, and Hinrich SchÃ¼tze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

[29] Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword Extraction from a Single Document Using Word Co-occurrence Statistical Information. *Int. J. Artif. Intell. Tools* 13, 1 (2004), 157–169.

[30] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[31] Yuan Ni, Qiong Kai Xu, Feng Cao, Yosi Mass, Dafna Sheinwald, Hui Jia Zhu, and Shao Sheng Cao. 2016. Semantic Documents Relatedness Using Concept Graph Representation. In *WSDM*.

[32] Helen J Peat and Peter Willett. 1991. The Limitations of Term Co-occurrence Data for Query Expansion in Document Retrieval Systems. *J. Assoc. Inf. Sci. Technol* 42, 5 (1991), 378.

[33] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

[34] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*.

[35] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389.

[36] Anish Das Sarma, Alpa Jain, and Cong Yu. 2011. Dynamic Relationship and Event Discovery. In *WSDM*.

[37] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *IEEE Trans. Knowl. Data Eng.* 29, 1 (2017), 17–37.

[38] Andreas Spitz. 2019. *Implicit Entity Networks: A Versatile Document Model.* Ph.D. Dissertation. Heidelberg University, Germany.

[39] Andreas Spitz, Satya Almasian, and Michael Gertz. 2017. EVELIN: Exploration of Event and Entity Links in Implicit Networks. In *WWW Companion*.

[40] Andreas Spitz and Michael Gertz. 2016. Terms over LOAD: Leveraging Named Entities for Cross-Document Extraction and Summarization of Events. In *SIGIR*.

[41] Andreas Spitz and Michael Gertz. 2018. Exploring Entity-centric Networks in Entangled News Streams. In *WWW Companion*.

[42] Shulong Tan, Jiajun Bu, Chun Chen, Bin Xu, Can Wang, and Xiaofei He. 2011. Using Rich Social Media Information for Music Recommendation via Hypergraph Model. *ACM Trans. Multimedia Comput. Commun. Appl.* 7S, 1 (Nov. 2011).

[43] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding Through Large-scale Heterogeneous Text Networks. In *KDD*.

[44] Cornelis Joost Van Rijsbergen. 1977. A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval. *J. Doc.* 33, 2 (1977), 106–119.

[45] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[46] Wei Wang, Furu Wei, Wenjie Li, and Sujian Li. 2009. HyperSum: Hypergraph Based Semi-supervised Sentence Ranking for Query-oriented Summarization. In *CIKM*.

[47] Michael M. Wolf, Alicia M. Klinvex, and Daniel M. Dunlavy. 2016. Advantages to Modeling Relational Data Using Hypergraphs Versus Graphs. In *HPEC*.

[48] Jinxi Xu and W. Bruce Croft. 1998. Corpus-based Stemming Using Cooccurrence of Word Variants. *ACM Trans. Inf. Syst.* 16, 1 (Jan. 1998), 61–81.

[49] ChengXiang Zhai and Sean Massung. 2016. *Text Data Management and Analysis.* Morgan & Claypool.

[50] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous Hypergraph Embedding for Document Recommendation. *Neurocomputing* 216 (2016), 150–162.