SimMeme: A Search Engine For Internet Memes

Tova Milo
Tel Aviv University
Tel Aviv, Israel
milo@post.tau.ac.il

Amit Somech
Tel Aviv University
Tel Aviv, Israel
amitsome@mail.tau.ac.il

Brit Youngmann
Tel Aviv University
Tel Aviv, Israel
brity@mail.tau.ac.il

***Abstract*—As more and more social network users interact through *Internet Memes*, an emerging popular type of captioned images, there is a growing need for users to quickly retrieve the right Meme for a given situation. As opposed conventional image-search, visually similar Memes may reflect different concepts. Intent is sometimes captured by user annotations (e.g., tags), but these are often incomplete and ambiguous. Thus, a deeper analysis of the relations among Memes is required for an accurate, custom search. To address this problem, we present SimMeme, a Meme-dedicated search engine. SimMeme uses a generic graph-based data model that aligns various types of information about the Memes with a semantic ontology. A novel similarity measure that effectively considers all incorporated data is employed, and serves as the foundation of our system. Our experimental results achieve using common evaluation metrics and crowd feedback, over a large repository of real-life annotated Memes, show that in the task of Meme retrieval, SimMeme outperforms state-of-the-art solutions for image retrieval.**

I. Introduction

With the ubiquity of social-media platforms like Flickr, TuDing and Instagram, *Internet Memes* have grown from a peculiar subculture to a widespread phenomena, dramatically affecting online-marketing and interpersonal communication[1].

Internet Memes are media items, mostly captioned images with a sarcastic or humorist intent that are proliferated across the internet (example Memes are depicted in Figure 1). The term "Meme" was originally coined by the evolutionary biologist R. Dawkins as the corresponding cultural unit to the biological gene [2]. Since an increasing amount of social networks users are interacting through Memes to express opinions and emotions [3], there is a growing need to quickly retrieve the right Meme for a given user query. To this end, we present SimMeme, a Meme-dedicated search engine.

Adequate retrieval of Memes, as opposed to regular images, is particularly challenging since: (i) They convey semantic meaning beyond their visual appearance; (ii) The associated text (caption) is often ironic and thus captions that contain common words may express different ideas; (iii) Since Memes typically propagate through social networks, they are often accompanied by user-annotated tags. While such tags provide meaningful semantic information, they are often incomplete or ambiguous, and therefore hard to utilize.

To illustrate, consider the following example.

Example I.1. *Consider the dashed part of Figure 1, depicting three Memes and their associated tags, in a form of a graph. While Meme C is more similar visually to Meme B than Mem A (identical background images, a common case for Memes), Meme A is more similar semantically, since both Memes are related to the act of feeding a baby. Also observe that Memes A and C are textually similar (starting with the same text, also typical for Memes), while semantically they describe different intent. This simple example demonstrates that different properties of the Memes (caption, image or tags) can be utilizes to define different search criteria (e.g., search by visual appearance, textual similarity, etc.).*

We argue that an ideal Meme search engine should (i) adequately record relevant information about the Memes (i.e., visual, semantic and textual information); (ii) provide an expressive interface that enables users to specify their search preferences (e.g., define the importance of visual appearance); and (iii) efficiently retrieve the relevant set of Memes for a given user query. SimMeme, as we will present next, has dedicated constructs to account for these needs.

To the extent of our knowledge, SimMeme is the first dedicated Meme search engine. Up until now, Meme-retrieval has been typically handled using general-purpose image search systems. General-purpose image retrieval has received much attention in the literature and can be divided into three main categories: (i) Content Based Image Retrieval (CBIR), which relies mostly on visual features [4]. This approach cannot be easily applied for Meme retrieval, as it suffers from the well-known "semantic-gap" problem, where visual features are not correlated with high-level semantic concepts (e.g., irony) that are especially prevalent in Memes. (ii) Text Based or Tag Based Image Retrieval (TBIR/TagIR) [5], [6], which emphasize external textual sources (e.g., the surrounding text of an image, tags). While the TBIR approach is highly useful when the images are taken from websites such as Wikipedia, it is less suited for Memes, which are often associated with a small number of users tags. On the other hand, the TagIR approach is mainly based on the relevance between the image tags and the query keywords. It tends to overlook the textual and visual properties, which may be more important to the user in a given context. (iii) Hybrid approaches [7], [8], which consider both textual and visual features. These works often rely on *representation learning* techniques (such as embedding) to obtain a unified representation of the images, comprising of features derived from both textual and visual properties. While the hybrid approach often outperforms the former three (or any naïve combination of them) in general image retrieval, such solutions do not allow for the flexibility and expressibility desired in Meme search (see Section IV).

In summary, the problem we consider in this paper is the *Meme retrieval task*, which, as opposed to image retrieval, requires to address user-defined search criteria. In contrast to previous work, SimMeme takes a holistic approach that

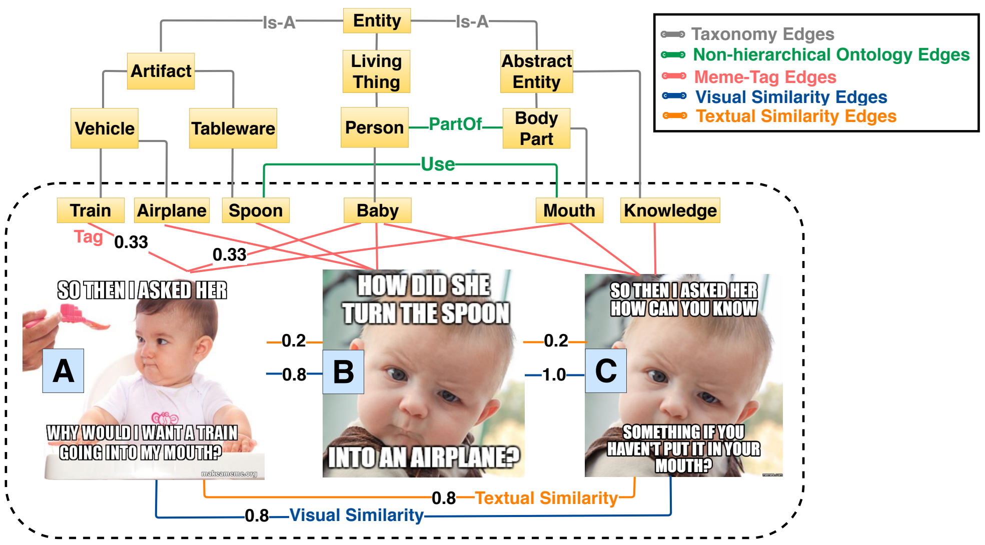


Fig. 1: Example HIN, aligning structural relations (among tags and Memes) with semantic ontology.

integrates all available data about the Memes, and allows users to indicate the importance of both the visual appearance and semantic meaning. Our experiments show that our approach outperforms general-purpose image-retrieval competitors in the Meme-retrieval task.

Our work is comprises of the following key contributions:

Heterogeneous Knowledge Representation: SimMeme models the various information on Memes using a simple yet rich and flexible graph-based data model. The visual and textual features extracted from the Memes (using external tools, e.g., [9], [4]) are all represented in a single Heterogeneous Information Network (HIN). To reveal the tags' semantics, we align the HIN with an external semantic knowledge base (i.e., an ontology), using standard alignment tools [10]. See Figure 1 for an example. Memes and tags are represented as nodes, and the various types of weighted-edges connecting them include: (1) Meme-Tag edges, where weights reflect the relevance between a tag and its associated Meme, if specified by users; (2) Visual and (3) Textual resemblance edges between Memes, where weights reflect the visual/textual resemblance; and (4) Ontology edges, which describe the semantic relations among the tags. The flexibility of this model, derived from its schemaless nature, allows to incorporate any other data related to the Memes (e.g., information about the Memes' creators).

Compound Meme-Similarity Assessment: Over the HIN, we devise a novel similarity measure that effectively considers all data incorporated, and serves as the main building block of our system. This measure is used to quantify the degree of relevance of a given Meme w.r.t. a user query. Our similarity function is declaratively defined, and thus, as opposed to most machine learning based measures [4], [5], [11], [12], can be easily interpreted and tuned according to the user requirements.

Our proposed measure refines SimRank [13], a well-studied similarity measure for information networks. Intuitively, SimRank quantifies the similarity of two network nodes based on the evaluation of the compound similarity of their network neighbors. However, SimRank does not consider semantic knowledge and edge weights. To overcome this, we refine SimRank by enriching its link-similarity definition with edge weights and semantic similarity. While this incorporation is simple and intuitive, it poses several performance challenges.

To maintain an efficient evaluation, we devise two optimization techniques, one is based on semantic pruning and the second is a probabilistic framework for computing approximated similarity scores, anchored in a careful modification of the underlying random surfer model of SimRank. For the latter, we define the *Semantic-Aware Random Walk Model*, which extends the standard uniform random walk model of SimRank. Then, we show how to harness *Importance Sampling* to attain an efficient computation in our setting. Importantly, our refined framework allows for a direct adaptation of existing optimizations previously proposed for SimRank (e.g., [14]).

Meme-Dedicated Search Interface: With SimMeme, users can query the Meme repository by providing a set of keywords, an example image/Meme, or both. Its advanced search features allow users to specify the weight of each keyword, and to alter the weights of textual and visual features, thus customizing the definition of *Meme-relevance*. Results are returned in semantically meaningful clusters (see Figures 2a and 2b), thereby enabling users to quickly focus on the group of Memes most relevant to their intent. SimMeme further provides users with an *explanation* for why a specific Meme is proposed, highlighting the semantic, visual and textual relations of the retrieved Meme to the given query (see Figure 3).

Experimental Evaluation: As Meme repositories are not yet publicly available for research purposes, we built a tagged-Meme database, comprising of 10K Memes collected by web crawling. We then used a crowdsourcing platform to tag the Memes. Since, to our knowledge, no other dedicated Meme-search systems are available, we compare our system's performance to alternative baselines, which include state-of-the-art methods for image retrieval and commercial image search engines. Our results show that SimMeme outperforms the competitors in the Meme-search task. Our query execution time experiments, performed over the Flickr datasets, demonstrate the efficiency and scalability of our approach.

A demonstration of our system was recently published [15]. The short paper accompanying the demonstration provides only a brief, high-level description of the system, whereas the present paper details our data model and algorithms. For space constraints, proofs are deferred to a technical report [16].

Paper outline: We present our data model and formally define the similarity notions in Section 2. In Section 3 we provide an overview of SimMeme and its components. The system implementation as well as our experimental study are described in Section 4. Related work is presented in Section 5 and we conclude in Section 6.

II. Preliminaries

We next describe how the data is modeled, then present the novel similarity measure employed by SimMeme. Particularly, we first recall the SimRank similarity measure, the main building block of our measure, then present our refined variant, which incorporates semantic knowledge and edge weights in the similarity assessment.

A. Knowledge Repository

Memes and tags form the basis of our data model, which is enriched by the visual and textual features of the Memes. We therefore use a Heterogeneous Information Network (HIN), a flexible graph-based model that can capture and integrate various types of data.

Definition II.1. ***Heterogeneous Information Network**. A HIN is a directed weighted graph $G = (V, E, \Phi, \Psi, W)$, where V is the set of nodes; E is the set of edges; $\Phi : V \to \mathcal{L}$ and $\Psi : E \to \mathcal{L}$ are the node/edge labeling functions and $\mathcal{L}$ is the labels domain, and $W : E \to R^+$ is the edge weight function.*

For a node $v \in V$, we denote by $I(v), O(v)$ the set of in and out neighbors of v, resp. An individual in-neighbor is denoted as $I_i(v)$, for $1 \leq i \leq |I(v)|$, if $I(v) \neq \emptyset$ (and $O_i(v)$, resp.). The node and edge labeling functions are used to describe different types of entities and links. Each node has an associated type: Meme, tag, or a user (if an information about the user who posted the Memes is available). Meme-Tag, the basic edge type, connects between a Meme and its tags, where the weights represent how relevant the tag is as defined by the annotator of the Meme. (A default weight is associated in case no such information is available.) Other types of edges between Meme-nodes represent *visual/textual* similarities. The edge weights reflect the corresponding resemblances, and are derived using designated tools [17], [11]. To avoid explosion of edges, we include only edges with similarity scores above a minimal threshold.

To further enrich the tags' semantics, we incorporate an *ontology* in the HIN. An ontology is a labeled graph consisting of general facts, e.g., that *Body-Part* "is a-part-of" a *Person*, or that *Train* "is-a" *Vehicle*. See Figure 1 for example. Indeed, several ontologies are publicly available (e.g., WordNet [18][1]). We embed the ontology in the HIN by aligning each tag with its corresponding ontology entity, using an entity-alignment tool (e.g., [10]). Ontologies typically contain a hierarchical *taxonomy* of concepts, e.g., that *Baby* "is-a" *Person* and *Person* "is-a" *Living-Thing*.

We will leverage the tags taxonomy relations in the following sections for (i) defining a particular semantic similarity measure (2) speeding up queries execution times (by semantic-based pruning), and (3) generating the result clusters' headers.

B. Similarity Notions

As mentioned, our HIN-based model contains various types of links between Memes and tags, as well as additional semantic information associated with the tags, as captured by the embedded ontology. We next devise a similarity measure between the nodes that effectively considers all information, and serves as the main building block of our system. As we explain in the next section, this measure serves to quantify the *degree of relevance* of each Meme, given a user query.

Our measure extends SimRank [13], a powerful and popular similarity measure for general (homogeneous[2] and unweighted) information networks. SimRank follows the intuitive assumption that: "two nodes are similar if they are related to similar nodes". Formally, given two nodes u and v in a network, their SimRank score is defined recursively as follows. If $u = v$ then $simrank(u, v) = 1$, else: $simrank(u, v)$ is given by the following formula *without the red colored parts*.

$$sim(u,v) = \qquad (1)$$
$$\frac{sem(u,v) \cdot c}{N_{u,v}} \sum_{i}^{|I(u)|} \sum_{j}^{|I(v)|} sim(I_i(u), I_j(v)) \cdot W(I_i(u), u) \cdot W(I_j(v), v)$$

where c is a decay factor $\in (0, 1)$, $N_{u,v} := |I(u)| \cdot |I(v)|$ and $sim(\cdot,\cdot)$ is the SimRank score of the corresponding neighboring nodes.

We enrich SimRank by weighting, at each step of the computation, the neighbors' link-similarity with the edge weights and their semantic similarity, which propagates from the tag-nodes to the Meme-nodes, and allows for a comprehensive assessment of the similarity between nodes. Formally, given a semantic similarity measure, denoted as $sem(\cdot,\cdot)$, the highlighted red parts indicate our refinements to SimRank's standard formula: (i) an additional semantic factor is added ($sem(u, v)$); (ii) the edge weights are taken into consideration when weighting neighbors similarity. Correspondingly, the normalization factor is set to:

$$N_{u,v} := \sum_{i}^{|I(u)|} \sum_{j}^{|I(v)|} W(I_i(u), u) \cdot W(I_j(v), v) \cdot sem(I_i(u), I_j(v))$$

where $sim(\cdot,\cdot)$ in the refined formula denotes the *refined* similarity of the neighbor-pairs. For both measures, if $I(u)$ or $I(v)$ are equal to $\emptyset$, then the scores are defined as 0.

Note that in the refined formula, the semantics of neighboring node-pairs propagates through the recursive computation, in particular, allowing the tags' semantic similarity (as defined by the incorporated semantic measure) to affect the Memes' similarity. Namely, the more similar semantic two Memes' tags are, the more similar the Memes will be according to our definition of similarity. As we will show in our experimental evaluation, these refinements are indeed critical, as SimRank and its other variants (e.g., a weighted variant of SimRank named SimRank++ [19]) are demonstrated to be significantly less suited for the Meme search task which requires to account for the semantics of tags.

[1]In our prototype implementation we used WordNet ontology.

[2]I.e., where all nodes/edges belong to a single type.

Semantic Similarity. Multiple semantic similarity measures have been studied in the literature [20], [21], [22]. In general, any semantic similarly function $sem(\cdot,\cdot)$ can be employed in our setting, as long as it satisfies the following three intuitive constraints: For all $u, v \in V$:

1) **Symmetry.** $sem(u, v) = sem(v, u)$
2) **Maximum self similarity.** $sem(u, u) = 1$
3) **Fixed value range.** $sem(u, v) \in (0, 1]$

Those requirements are necessary to prove the soundness of our measure (see [16]). The first two constraints are satisfied by all semantics similarity measures we are aware of (e.g., [20], [23]). As for the third constraint, scores can be normalized into a $[0 + \varepsilon, 1]$ range, for a small $\varepsilon > 0$ value.

For the completeness of this paper, we next present a simple and effective semantic measure that we have integrated in our prototype implementation (See Section V for a discussion on alternative measures). Lin [20] is a common *Information Content* (IC) based semantic measure that is defined over concepts' taxonomy (i.e. tags in our setting). The IC of a concept is quantified as the negative of its log likelihood, i.e., the more prevalent a concept is, the lower its IC value. Intuitively, the similarity between concepts measures the ratio of the amount of information needed to state their commonality to the information needed to describe them.

Definition II.2. ***(Lin)*** *Given two concepts, u and v in a taxonomy, their Lin semantic similarity score is defined as:*

$$Lin(u, v) = \frac{2 \cdot max_{IC(LCA(u,v))}(IC(LCA(u, v)))}{IC(u) + IC(v)}$$

where $LCA(u, v)$ denotes the set of the lowest common ancestors of u and v in the taxonomy.

Note that Lin is traditionally defined only for taxonomy node pairs. We extend the definition assignment for all other pairs of nodes, e.g., between Meme-nodes, to the constant value of 1, indicating that the semantic relations are unknown.

To estimate the IC values of the tags, we adopted the Seco formula [24], which provides a simple linear-time algorithm. Theses values are also useful for the presentation of search results (see Section III-C).

Properties. We can prove that like SimRank, our refined similarity measure is well defined and its equations can be solved using an iterative fix point computation (for proof see [16]). More specifically, the authors of [13] have proved that SimRank's iterative form has the following properties: (1) Monotonicity; (2) Existence: the solution always exists and converges to a fix-point, and (3) Uniqueness: for every choice of the decay factor $c \in (0, 1)$, the solution is unique. A difference from Simrank is that in our case, the uniqueness property holds only for c values s.t: $0 < c < min(argmin_{N_{u,v}}(N_{u,v}), 1)$. Yet, our experiments show that for real-life data, the upper bound is high enough to comfortably accommodate typical c values chosen for SimRank (e.g., 0.6, as used in [14]).

As for SimRank [13], the iterative procedure for computing similarity scores is expensive. Towards this end, we devise a probabilistic framework for computing *approximated similarity scores*, which allows for a direct adaptation of SimRank's existing optimizations (see Section III-B).

Node	**IC value**
Entity	0.001
Living Thing, Abstract Entity	0.01
Vehicle,Tableware, Artifact	0.03
Person, Body Part,	0.05
Baby	0.08
Mouth, Knowledge	0.29
Train, Airplane, Spoon	0.4
Meme A, Meme B, Meme C	1.0

TABLE I: IC values for Example I.1.

To complete the picture, we next provide the similarity computation of Example I.1 from the Introduction, computing the three Meme-nodes similarity scores according to our refined measure, using Lin as the integrated semantic measure. This example demonstrates the flexible nature of our measure.

Example II.3. *Consider again the simple network depicted in Figure 1. Using the nodes' IC values computed for our experimental study (see Section IV-A) as listed in Table I, we compute the similarity scores according to our refined measure. We set $c = 0.6$[3] and executed 4 iterations of the iterative procedure. As explained in Example I.1, every Meme-pair is either semantically, visually or textually similar, and thus the obtained similarity scores are rather close:*
$sim(A, B) \approx 0.058, sim(A, C) \approx 0.058, sim(B, C) \approx 0.057.$
However, if we "ignore", e.g., the textual similarity edges by setting the corresponding edge weights to 0, we get that:
$sim(A, B) \approx 0.064, sim(B, C) \approx 0.063, sim(A, C) \approx 0.057.$
On the other hand, we note that according to SimRank (using the same parameters), the removal of the textual/visual/Meme-tag edges does not change the Meme-pairs ranking. I.e., in all cases: $sim(A, C) \geq sim(A, B) \geq sim(B, C)$.

This simple example demonstrates that changing the edge weights can affect the similarity scores and hence the relevance ranking of the Memes. Indeed, as we explain in the following section, based on this observation, SimMeme enables users to specify their intent in Memes search by setting custom weights and thereby influence the results ranking.

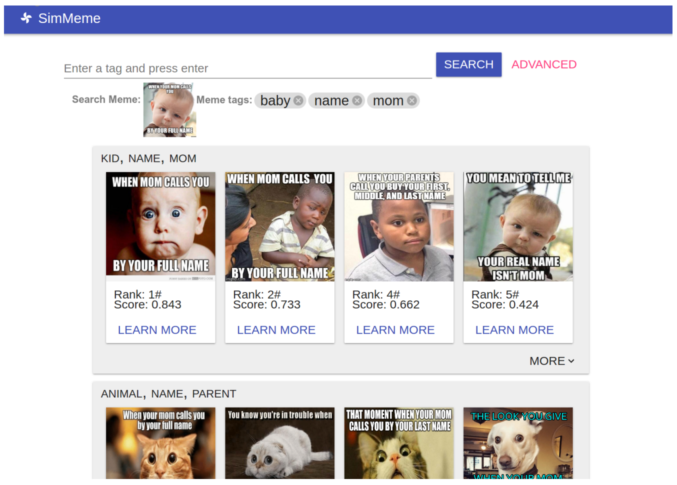
III. System Overview

Next, we present SimMeme workflow. We first explain how a user query is formulated, then describe its execution process. In particular, we focus on the two main optimizations SimMeme employs to achieve an efficient, scalable evaluation: semantic-based pruning and approximated similarity computation. Last, we briefly discuss the results presentation.

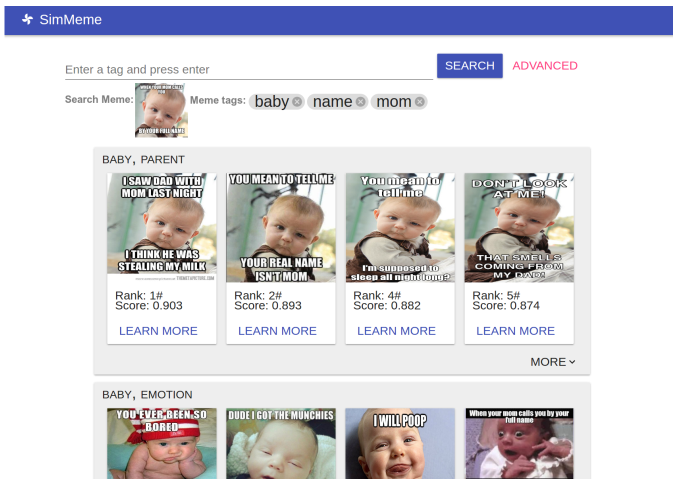
A. Query Formulation

SimMeme supports two types of search queries: (i) keyword-based queries, where the user specifics a set of search keywords, and (ii) image-based queries, where Meme/image is provided. SimMeme also allows users to provide both keywords and an image, and we refer to such combined queries as image-based queries as well. To ease the presentation, we assume in what follows that the image/Meme in a given image-base query exists in our repository. To support queries in which the user points to a new image/Meme, given the query, the

[3]A value that ensures the uniqueness of the similarity scores in this example.



(a) Results generated with visual factor set to 0.2.



(b) Results generated with visual factor set to 1.

Fig. 2: SimMeme UI: A Meme-search query with varying user-defined parameters.

system computes the relevant visual and textual edge weights, a process that can be efficiently done using solutions such as [25] for indexing.

SimMeme UI allows users to tune the importance of visual/textual features, and to set custom weights to the specified keywords (indicating the importance level of each keyword). The given parameters affect the similarity scores and hence the relevance ranking of the Memes. For example, Figure 2 depicts the results obtained for an image-based query, where the user provided an example Meme as the input and had set varying values for the visual factor. Particularly, in the query that resulted in the answer depicted in Figure 2a, the user had set a low visual factor (of 0.2), and hence the system displays semantically and textually similar Memes, while in Figure 2b, the user had set the visual factor to its maximal value (of 1), and hence the system ranks the visually similar Memes higher.

We model a user query as a (temporary) Meme-node, linking it to the relevant tag-nodes as provided in the query's keywords, with the corresponding custom weights (in case the user did not specify weights, we use default values). Given an image-based query, we simply consider the corresponding (image/Meme) node as the query node. Last, we update the graph edge weights according to the user's custom parameters. To illustrate, consider the following example demonstrating this process.

Example III.1. *Given the user query whose results are presented in Figure 2a, where the user pointed to an existing Meme and had set the visual similarity factor to* 0.2*, we consider this Meme-node as the query node, and multiply the weights of all edges in the graph with the label "visual_similarity" by a factor of* 0.2*. As another example, for the keyword-query Baby and Food, assuming no custom keywords' weights were provided, the system generates a temporary query node, connecting it to the relevant tags with the default edge weight value of* 1.

B. Query Execution and Optimizations

Given a query node, SimMeme retrieves its top-k most similar Meme-nodes, as determined by our similarity measure. To do this efficiently, it employs the following optimizations:

1) A pruning method that allows to restrict attention to a subset of potentially relevant Memes.
2) A probabilistic framework that allows for an efficient similarity estimation.

We next present more details about these techniques, and conclude this section with a brief discussion on additional possible execution times optimizations.

Pruning. We next describe the key principles of our method that constructs the initial set of candidate Memes to appear in the results set of a given user query. Consider again Equation 1. Given a query-node, a "promising" Meme-node to obtain a high similarity score is a one that: (i) its associated tags are semantically related to the query keywords, or (ii) is visually/textually similar to the query-node. A possible (but rather inefficient) pruning approach would be to consider each Meme-node separately, and discard those who have no semantically related tags (according to their semantic similarity scores), nor are visually/textually similar to the query node.

To speed up the computation and avoid considering each Meme-node separately, we leverage the following two observations. First, note that visually/textually related Meme-nodes are (by construction) already connected to the query-node, and can thus be identified immediately. Second, to identify Memes with semantically related tags, one can exploit the semantic hierarchy induced by the taxonomy. More specifically, we traverse the tags taxonomy in a top down manner, computing the tags semantic similarity scores to the query keywords. If a certain tag is semantically related to one of the query keywords, then its descendants in the taxonomy may be semantically related to this keyword as well. Consequently, all Meme-nodes connected to this tag-node or to one of its tag-node descendants, are added to the output set and we halt. Otherwise, we continue to explore the current tag direct tag-node descendants according to the taxonomy.

While not all selected tags, and hence their corresponding Meme-nodes, are bounded to be highly similar to the query's keywords (resp. query-node[4]), our experiments indicate that this method, nevertheless, has a significant affect on execution

[4]Unless the chosen semantic measure is monotone w.r.t. the partial order induced by the taxonomy

times. Note that the correctness of this procedure is always guaranteed, since all tags that were excluded in this process are those that have been explicitly checked and were found not to be semantically related. To further improve performances, SimMeme maintains a cache mechanism that maps between keywords mentioned in previous queries and their computed set of semantically-related tags.

Similarity Estimation. Given a set of candidate Meme-nodes, we next devise a probabilistic framework that efficiently estimates the similarity scores between each candidate and the query-node. This framework is based on previous developments for SimRank. We start by recalling SimRank's basic approximation framework, then show that a naïve solution of using this framework for our refined measure leads to a quadratic increase in the sample size. To overcome this, we employ *Importance Sampling*. Finally, we present our refined framework. For space constraints, we provide here only a brief overview of this framework. Full details are provided in [16].

Our probabilistic framework stems from an intuitive connection between SimRank to a "random surfer" model (established in [13]): The SimRank score between a pair of nodes measures how soon two random surfers are expected to meet, if they start at the two nodes and randomly walk on the graph backwards. Formally, given two random walks $t_1 = \langle u_1 = u, ..., u_k \rangle$ and $t_2 = \langle v_1 = v, ..., v_k \rangle$, from the nodes u and v resp. that intersect at the same node after exactly τ steps, it can be proved that $simrank(u, v) = E[c^\tau]$.

Utilizing the equality above, a *Monte-Carlo* (MC) framework that efficiently approximates SimRank scores was proposed in [26]. This framework first precomputes a set of reversed random walks from each node s.t. (i) each set has exactly n walks, and (ii) each walk is truncated at step t. Then, given two nodes u and v, the method estimates their SimRank score as:

$$simrank(u, v) \approx \frac{1}{n} \sum_{l=1}^{n_w} c^{\tau_l}$$

where τ_l denotes the number of steps prior to their first intersection and ∞ otherwise.

However, this framework can not be directly applied for our refined measure, as it assumes that the random surfers choose the next step *uniformly at random* and the choices of the surfers *are independent*. Contrarily, in our setting, the surfers must also consider the edge weights and the semantic similarity, and hence the two surfers choices *are dependent*. To facilitate efficient similarity computation, we first define a random walk model adequate for our setting, referred as the *Semantic-Aware Random Walk* (SARW), then describe how the approximated computation is performed using such walks.

Given two random walks t_1 and t_2 as defined above, let t denote the *coupled random walk* of t_1 and t_2, where $t = \langle (u_1, v_1), ..(u_k, v_K) \rangle$, and let $\tau(t)$ denote the prefix of t until the first meeting point of t_1 and t_2. Its probability, denoted as $P[t]$, is defined as the product of probabilities in each step. We define the *semantic-aware probability distribution* to be the probability that two random surfers in the current nodes u, v will next move to their neighbors x, y as follows:

$$P[(u, v) \rightarrow (x, y)] = \frac{W((u, v), (x, y)) \cdot sem(x, y)}{\sum_{k=1}^{|O((u,c))|} W((u, v), O_k(u, v)) \cdot sem(O_k(u, v))}$$

Let $t = \langle w_1, ..., w_k \rangle$ be a coupled random walk (i.e., w_i is a pair of nodes), the probability $P[t]$ of traveling within it is then defined as: $P[t] = \prod_{i=1}^{|t|-1} P[w_i \rightarrow w_{i+1}]$.

Let $t : (u, v) \rightarrow (x, y)$ denote a coupled walk from the nodes u, v to the nodes x, y resp., $l(t)$ denotes its length, and: $T = \{\tau(t) : (u, v) \rightarrow (x, x)\}$ is the set of all coupled walks starting from u and v that intersect after a finite number of steps. We can prove that:

$$\sum_{t \in T} sem(u, v) \cdot P[t] \cdot c^{l(t)} = sim(u, v)$$

where P is the semantic-aware distribution. Since such walks cannot be sampled independently (as it involves the semantic similarity of the next pair to be chosen), properly approximating similarity scores would require a much larger sample size of $n \cdot |V|^2$ walks, instead of $n \cdot |V|$ as in the simple uniform setting of SimRank. To overcome this, we employ *Importance Sampling*, a general technique for estimating properties of a particular distribution, while only having samples generated from *another* distribution.

For a single node-pair $u, v \in V$, we wish to estimate the expected value of the function $sem(u, v) \cdot c^{l(x)}$, where x is a coupled random walk drawn from the semantic-aware distribution P, namely:

$$sim(u, v) = E_P[sem(u, v) \cdot c^{l(x)}] =$$

$$sem(u, v) \cdot E_P[c^{l(x)}] = sem(u, v) \cdot \sum P(x) \cdot c^{l(x)}$$

Given n coupled random walks $x_1, ..., x_n$ drawn from P, an empirical estimator of $E_P[c^{l(x)}]$ is:

$$E_P[c^{l(x)}] \approx \frac{1}{n} \sum_{i=1}^{n} c^{l(x_i)}$$

We derive that:

$$E_P[c^{l(x)}] = \sum \frac{P(x) \cdot Q(x) \cdot c^{l(x)}}{Q(x)} \approx \frac{1}{n} \sum_{i=1}^{n} c^{l(x_i)} \frac{P(x_i)}{Q(x_i)}$$

where Q is a different distribution (with the same support as the distribution P). That is, we can obtain an unbiased estimator of the expected value of the function $c^{l(x)}$ under the distribution P, using only samples drawn from Q.

We can show that the expected value of the new estimator is indeed equal to the desired one, i.e., for every u, v:

$$sem(u, v) \cdot E_Q[\frac{P(t) \cdot c^{l(t)}}{Q(t)}] = sem(u, v) \cdot E_P[c^{l(t)}] = sim(u, v)$$

where t is a coupled random walk starting from u and v. Note that the latter inequality holds for any choice of the distribution Q. In our setting, we set Q to be the uniform distribution, i.e., we sample separated random walks with the underlying uniform distribution, to enable sampling the random walks at pre-processing time (instead of at query-time). That is, as the graph weights may changed according to user custom weights, the uniform distribution ignores the edge weights and hence allows us to sample the random walks without the need to re-sample them according to the updated edge weights.

Using our proposed framework, we can prove that the average time required to answer a single-pair similarity estimation is $O(n \cdot d^2 \cdot t)$, where n is the number of sampled random walks of length t from each node, and d is the average in-degree in the graph. For a detailed complexity and error analysis, we defer the interested readers to [16].

Supporting queries with user-defined weights: As in the original SimRank's MC framework, the (separated) random walks are sampled at pre-processing time. Given a query-node, we first sample a set of random walks starting from it, then compute the similarity scores between this query-node and all candidate Meme-nodes. Recall that the user may further provide custom visual/textual factors requiring to update the edge weights, i.e., to multiply the relevant edge weights by the corresponding factors. Since the (separated) random walks are sampled according to the uniform distribution that ignores the edge weights, there is no need to sample again the walks after updating the weights. However, as the weights do affect the coupled random walks' probabilities according to the semantic-aware distribution P, we dynamically update the edge weights while computing these probabilities during query-time.

Further optimizations We conclude with three remarks regarding further running times optimizations employed by SimMeme. First, to allow an efficient query evaluation, we employ a preprocessing which consists of: (1) Sampling the random-walks from all Meme-nodes; (2) Precomputing the IC values, and (3) The precomputation required for the incorporated semantic measure (e.g., processing the taxonomy using [27], to enable constant-time LCA computations, allowing constant-time Lin computations).

Second, a large portion of SimRank's optimizations are based on its MC framework (e.g., [28]) and can be adapted for our measure as well, using the adjusted MC framework. For example, in [14], the authors provided a new interpretation of SimRank incorporating $\sqrt{c}$-random-walks, which ensures that the expected length of walks is short. This interpretation can be immediately applied for our setting as well using $\sqrt{c}$-SARWs.

Last, we note that SimMeme is a highly parallelizable system, as the similarity scores are independent of one another and can be computed simultaneously. Hence, a natural speedup is achieved by distributing the similarity computations over multiple cores.

C. Results Presentation

Once the query is processed, SimMeme displays the top-k retrieved Memes in semantically related clusters. To illustrate, consider again Figure 2, depicting the results for an image-based query. In Figure 2a, the upper block contains only Memes semantically related to the tag *Kids*, while the lower block contains ones that are all related to the tag *Animals*. Furthermore, the system generates a meaningful header for each such cluster, namely, a set of tags describing it. Going a step further, the user may click on a result Meme to view an explanation about its retrieval process. See Figure 3 for example. This explanation provides an intuition about the relevant parts of the HIN that led to the selection.

To generate the clusters, our prototype engine uses *K-Means* (an effective clustering method), taking the proposed similarity measure as an input. We note that several other algorithms for image results summarization have been proposed (e.g., [29], [30]) and can be applied here as well. We next devise a simple method for generating the cluster headers, and shortly describe how the explanations are generated.

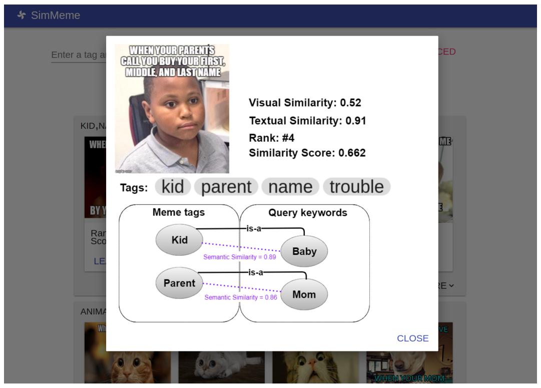

Visual Similarity: 0.52
Textual Similarity: 0.91
Rank: #4
Similarity Score: 0.662
Tags: kid parent name trouble
SimMeme
Enter a tag a
CED
WHEN YOUR PARENTS CALL YOU BUY YOUR FIRST, MIDDLE, AND LAST NAME
Meme tags
Query keywords
Kid
is-a
Baby
Semantic Similarity = 0.89
Parent
is-a
Mom
Semantic Similarity = 0.86
CLOSE

Fig. 3: An explanation popup.

Generating a Cluster Header. A cluster header in our system comprises of a set of concepts (tags or their higher level abstractions in the taxonomy), which describe the Memes within it. To generate the header, we use a simple procedure that utilizes the tags taxonomy and their IC values. Intuitively, our algorithm works as follows: for each Meme, we consider a set of tags consisting of its tags and all their taxonomical ancestors. From these sets, we identify the set of concepts that appear with frequency above a certain threshold, and select the most specific ones (i.e. frequent concepts that are not subsumed, according to the taxonomy, by other frequent concepts). Finally, we choose the ones having the highest IC values as the header[5]. For example, consider the upper cluster in Figure 2a. The tag *Name* was selected as it is associated with all Memes in this cluster and has a high IC value. Additionally, as all Memes in this cluster posses either the tag *Baby* or the tag *Kid* and according to the taxonomy *Baby* "is-a" *Kid*, the tag *Kid* gets a high frequency as well. Since its IC value is high, it also appears in the title.

Generating explanations. SimMeme provides an explanation for each retrieved Meme, highlighting the reasons for its selection. This feature assists the user to understand how the Meme's properties (e.g., tags, visual appearance) affected its ranking. The explanation consists of: (1) the Meme's tags; (2) the visual/textual similarity scores of the Meme, and (3) the semantic relations among the keywords and tags, as induced by the ontology. For example, Figure 3 displays an explanation for a resulted Meme. Although the query Meme and the result one are not visually similar, they are textually similar and their tags are semantically related: *Baby* "is-a" Kid, *Mom* "is-a" *Parent*, and they share the tag *Name*.

IV. Experimental Study

We conducted an empirical evaluation of SimMeme, studying three main aspects of our work:

1) ***Meme-retrieval quality.*** We assessed the adequacy of Memes returned by SimMeme to real-life user queries by

[5]In our implementation we have set the threshold so that $2-5$ tags are selected for each header.

conducting a user study. We compared our Meme search-engine with various baselines, including state-of-the-art approaches for image-retrieval, popular commercial image-search engines, and alternative similarity measures.
2) ***Applicability of our Search Interface.*** We asked users to rate their satisfaction from the system's search UI and in particular its novel features: customized search, results clustering, and explanations.
3) ***Execution times.*** We examined the performance of SimMeme as a factor of several parameters, and the ability of the system to scale.

We start by describing the experimental setup, then provide details on each of these three experiment sets.

A. Experimental Setup

Next, we describe our prototype implementation and its parameter settings, then present the datasets and baselines used throughout the experimental study.

Implementation and Settings: We implemented a prototype of SimMeme in Python 2.7, using the NetworkX library (https://networkx.github.io/) for graph processing and React (https://reactjs.org/) for the system's UI. We used SSIM [31] to calculate the visual similarity between Memes, and the noun sub-part of the lexical base WordNet [18] (82K English nouns), as the tags ontology. We used the following system parameters: The decay factor (c) of our similarity measure was set to 0.6 (this is also a typical choice for SimRank [14]), and for the probabilistic framework, a set of 50 random walks of length 15 was sampled from each Meme-node. According to our experiments, this choice of the parameters allows for fast execution times, while maintaining negligible error rate. All experiments were conducted on a 24 cores (2.1GHz) Linux server with 96GB of memory.

Datasets: As opposed to standard image retrieval where several benchmarks and ground-truth datasets exist, there are no such ones for the Meme-retrieval task. Therefore, we constructed a tagged-Meme repository[6] as follows. (Due to the page limitation, we provide here only a brief explanation on how this dataset was build.) First, we gathered 10K Memes via web crawling, then obtained tags via crowdsourcing: For each Meme, we asked 5 CrowdFlower workers (https://www.crowdflower.com/) to describe it via $3-8$ English nouns, then selected only tags provided by at least 3 different workers. The weights on the Meme-tag edges were set according to the proportion of workers that chose a specific tag to describe a Meme. To test the system scalability, we used the standard Flickr dataset [32], a commonly used dataset in image-retrieval, comprising of over 2.3M images, each associated with $5-8$ users tags.

Baseline algorithms: We compared our system to multiple baselines from three main categories:

1. General purpose image-retrieval algorithms. We examined several CBIR, TagIR and Hybrid algorithms: (1) *CNN-CBIR* [4], a *content-based* approach that uses a convolutional neural network to generate high-level descriptions of the images. (2) *Vis-W2V* [5], a *text-based* algorithm that employs a word embedding technique based on Word2Vec [33]. As for *hybrid based* algorithms, we tested: (3) *ImageNet distance* [7], an image-distance function that considers both the visual features and a (WordNet based) taxonomy of tags; and (4) *DeepWalk* [11], a deep learning based algorithm that learns a unified vector representation for both tags and Memes (See Section V for more details). We considered two implementations of DeepWalk, one that uses only the Memes-tags graph (as in the original paper [11]) and another that considers our full aligned HIN graph, ontology included. As the latter shows better performance, we present in this section only its results.

2. SimMeme with alternative similarity measures. To assess the utility of our proposed similarity measure, we formed three baselines by replacing it with existing measures commonly used for HINs, or a different aggregation of the obtained Lin and SimRank scores. (5) *PathSim* [34], an alternative random-walk based similarity measure. This measure is label-aware, and all random walks are sampled according to a predefined *meta-path scheme*. We used meta-paths (up to length 10) of the form *Meme-Meme* or *Meme-Tag*-Meme* (See [34]). (6) and (7) *Multiplication and Average*, a simple product (resp. average) of the obtained Lin and SimRank scores, as opposed to our formula that interweaves the two throughout the recursive computation.

3. Commercial search engines. Last, we further considered two popular commercial search engines: (5) *Google-Image* (https://images.google.com/) and (6) *Bing-Image* (https://www.bing.com/images). Since they do not reveal implementation details, we used their online versions, which use larger datasets and contain more image meta-data (e.g., the text surrounding Memes posted on web pages). Hence, they serve as representative examples for TBIR approaches. To allow for a fair comparison between all baselines, we extended our dataset to include all Memes retrieved by either Google or Bing (up to rank 30 of the results-set order) that were not already present. For the formulation of image-based queries, we used their UIs that allow to search by both image and keywords.

Additional baselines were considered yet omitted from presentation due to their inferior performance. These include: embedding-based algorithms ([35], [36], alternative similarity measures ([12], [13]), and in particular, additional weighted variant of SimRank, SimRank++ [19]. Furthermore, we considered several alternative semantic measures ([22], [23]).

B. Meme-Retrieval Quality Evaluation

Experimental setup. To assess the results quality of SimMeme and the competitors, we conducted two experiment sets: First, we recruited 20 student volunteers[7] and asked them each to compose one keyword-based and one image-based queries (total of 40 queries), then asked the participants to asses the quality of results obtained by each baseline. Second, we randomly selected 30 keyword-based and 30 image-based queries, and performed a crowd-based evaluation to measure the relevance of the results to the corresponding queries, using CrowdFlower workers. To generate random keyword-based queries, we sampled tags uniformly at random, and to generate random image-based queries we randomly selected an existing Meme as the query node. In total, we report the results quality of 100 Meme retrieval queries.

[6]As we explain in Section VI, we are currently working on devising a benchmark for Meme retrieval, making this dataset public available.

[7]15 out of the 20 students are not CS students.

For a fair comparison with the competitors, the queries did not employ any of the novel features that are available only in SimMeme, e.g. custom weights. The queries were then mapped to the required input representation of each baseline: a vector representation for CNN-CBIR, Vis-W2V and DeepWalk, a query node for ImageNet and Pathsim, and a set of search keywords and possibly an image for Google and Bing. For each query we considered the top 15 results obtained by each of the competitor algorithm. To measure the relevance of the result sets to the corresponding queries, each result Meme was presented to the student volunteer who wrote the query, and to 5 CrowdFlower workers, along with the search query. The users (volunteers or workers) were asked to rate each result Meme relevance on a scale of $1-3$, where 3 (resp. 1) indicates relevance (irrelevance). We then took the average (volunteers or workers) rating as the Meme's relevance score.

Last, to obtain an aggregated score for each results set, we use two common information retrieval evaluation measures: Normalized Discounted Cumulative Gain (NDCG) with a cutoff at 15, and Precision at 15 (P@15). When computing precision, we assumed that a Meme was relevant if its average score was ≥ 2 (similar trends were demonstrated for an average scores of 3). Computing the NDCG requires comparing the obtained ranking of each baseline with a perfect ranking algorithm. Here, the perfect ranking used was the one that ranks only relevant results (i.e. score of 3) in every position.

Result Analysis. Table II depicts the results obtained of all baselines, according to the workers and volunteers. Note that some of the baselines (i.e., CNN-CBIR, ImageNet, Vis-W2V) do not support keyword-based queries, therefore we considered their results only for image-based queries.

The results demonstrate the advantage of our method, achieving the highest average P@15 and NDGC scores, both by the students volunteers and by the crowd workers. We next provide a detailed comparison to all baseline algorithms.

General purpose image-retrieval. First, observe that Vis-W2V, a TagIR approach, returns more adequate results than CNN-CBIR, a CBIR approach. This result stems from the well-known "semantic-gap" problem affecting CBIR algorithms, and in particular, when retrieving semantically-rich images such as Memes, which often share the same background image. However, the hybrid-based baseline (e.g., DeepWalk, ImageNet) outperform the latter competitors. The results show that ImageNet is less suited for Meme-retrieval than DeepWalk and SimMeme, as it only considers visually similar Memes, ranked according to their semantic relations. As common for Memes, even visually dissimilar Memes may be semantically or textually related to a user query (see Figure 1). Among all baselines, DeepWalk presented the best performance, and was only surpassed by SimMeme(the $p < 0.05$ in a sign-test between DeepWalk and SimMeme).

We note that, besides the superior accuracy of SimMeme, another important advantage of our approach over DeepWalk is the flexibility that users have in their search queries. To demonstrate this, we reevaluated all image-based queries, this time adding a requirement to retrieve only visually similar Memes. In SimMeme, this is simply done by setting the visual factor to its maximum value, where in DeepWalk this is not possible unless we re-train the algorithm (originally, the users did not specify the weights and hence SimMeme used its default parameters). Next, we evaluated the queries again and asked the crowd workers to rank the relevance of the new results, while taking into account the additional request of retrieving only visually similar Memes. The average P@15 and NDGC scores achieved by DeepWalk dropped to 0.66 and 0.62, compared to 0.89 and 0.82 for SimMeme. This stems from the fact that the feature vectors used in DeepWalk are learned at preprocessing time, thus the user request, containing customized parameters, is disregarded.

SimMeme with alternative similarity measures. When we replaced our proposed measure by PathSim, Multiplication and the Average baselines, the NDCG and P@K scores decreased. We note that these baselines were use as alternative implementations to our measure, and hence the pruning phase was still used for initial filtering. These results show that our refined similarity measure better integrates visual and semantic features, in the context of Memes search.

Commercial search engines. Interestingly, SimMeme also outperformed Google and Bing in the Meme retrieval task. While their image repositories are larger than ours and they consider additional textual meta data about the Memes, they apparently less suited for Meme search. It worth mentioning that nowadays, most users use those platforms for the Meme-retrieval task, as no dedicated system which allows flexible, custom Meme search is available.

Last, recall that our prototype engine employs a fairly simple semantic similarity measure - Lin. To assess the adequacy of this choice, we replaced Lin by the node-similarity scores derived from the richer DeepWalk measure. Interestingly, this did improve the quality of results, indicating that Lin is indeed an adequate choice that is able to successfully capture semantic relations among the tags.

C. UI Evaluation

To evaluate the applicability of the proposed search interface features, we further asked the student volunteers (who wrote the queries) to fill a short survey for rating their satisfaction from the following features: (i) the customized search; (ii) the clustered results and their headers, and (iii) the result explanations. The participants were asked to rate each feature, on a scale from 1 to 5 (most satisfactory), regarding three different aspects: understandability, relevance and usability.

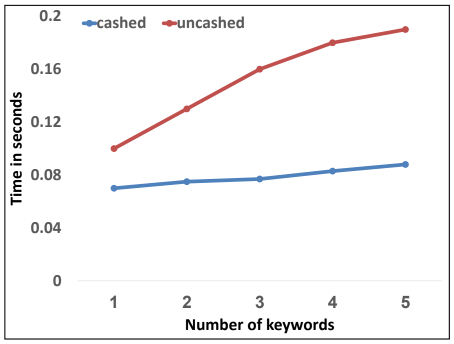
The results of this survey are summarized in Table III, where each cell contains the averaged score of all participants. All rating scores were between 3 to 5, where the customized search feature received the highest score on all accounts. In particular, all participants considered this feature a "vital" for Meme search (i.e. 5 on the usability scale), as it allows users to easily specify their intent. The explanation feature received a high score of 4.6 for its usability, yet a lower score of 4.0 for its intelligibility. Particularly, 3 of the participants mentioned that a natural language explanation would be more understandable. As the implementation of such feature involves natural language processing which is beyond the scope of this work, we leave this for future work.

D. Performance Evaluation

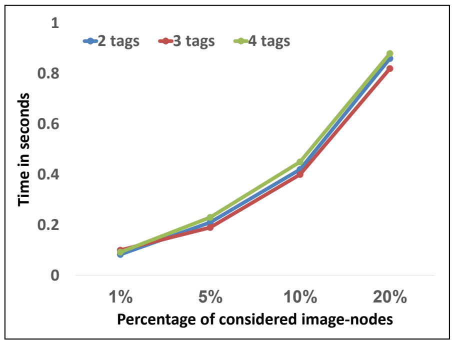
To demonstrate the ability of SimMeme to scale, we conducted the following experiments on the larger Flickr

Baseline	Approach	Avg. P@15 (volunteers)	Avg. NDCG (volunteers)	Avg. P@15 (workers)	Avg. NDCG (workers)
SiMeme	Hybrid	0.93	0.84	0.92	0.83
DeepWalk	Hybrid	0.90	0.81	0.88	0.80
ImageNet	Hybrid	0.73	0.66	0.74	0.65
Vis-W2V	TagIR	0.70	0.72	0.74	0.68
CNN-CBIR	CBIR	0.65	0.64	0.67	0.64
Google	TBIR	0.77	0.75	0.76	0.76
Bing	TBIR	0.64	0.62	0.63	0.63
PathSim	Hybrid	0.81	0.78	0.83	0.76
Average	Hybrid	0.76	0.68	0.77	0.66
Multiplication	Hybrid	0.70	0.68	0.71	0.67

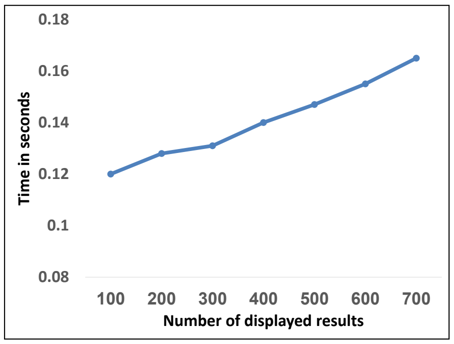
TABLE II: Users (volunteers or workers) relevance assessment for Meme retrieval queries.



(a) Semantic pruning.



(b) Similarity computations.



(c) Results presentation.

Fig. 4: Average running times of 1K randomly selected image-based queries on the Flicker dataset.

Feature	Understandability	Relevance	Usability
Clusters and headers.	4.2	4.3	4.5
Advanced search parameters.	4.5	4.1	5
Results explanations.	4.0	4.3	4.6

TABLE III: Users averaged ranking for SimMeme UI features.

dataset. Here again, we aligned between the images' tags and WordNet entities, obtaining an HIN with over 2.3M image nodes (there are no Meme-node in this dataset). We examine the queries execution times w.r.t. to the following parameters: the size of the repository, the number of displayed results, and the number of search keywords. We next report the system preprocessing time, then analyze the query execution time.

Preprocessing: Recall that the preprocessing phase consists of three parts: sampling the random walks from each image node, computing the IC values of all tags, and processing required for constant-time semantic similarity computations (as described in Section III-B). The overall time required to sample the random walks was 2.5 minutes; computing the IC values required additional 2 minutes, and the processing of WordNet taxonomy took approximately 5 minutes. We mote that this is a one-time effort performed during system setup (or periodically, to account for added data).

Query Execution times: We report the running time of SimMeme w.r.t. its main components: pruning, similarity evaluation and results presentation. This partition provides an intuition on the ability of SimMeme to scale, and a clear image of its computational bottleneck. To complete the picture, we also report the running time of SimMeme as a whole.

Recall that we cache the sets of relevant tags for search keywords that were already computed. This caching mechanism allows avoiding the computation of these sets at runtime. In this experiment we distinguish between newly arrived keywords, i.e., keywords that are not cached yet, and repeated keywords, where the query contains keywords that were all previously computed and cached.

For this experiment we randomly picked 1K image-nodes for image-based queries, containing both keywords and an image. We measured the execution time of a query as a function of the number of its tags (i.e. the query keywords). Figure 4 depicts the average running times of those queries. Particularly, Figure 4a depicts the average running times of the pruning phase, with and without using caching. Observe that while all queries were processed in less than 0.2 seconds, a further improvement of up to 50% can be achieved by precomputing and caching common search keywords.

Next, we consider the time required to evaluate the similarity scores between the candidate image nodes and the query node, as described in Section III-B. We start by examining various sizes of the image repository, however, we have noticed that the repository size has only a marginal affect on the execution times. The reason for that stems from the fact that the similarity scores are only computed for nodes that passed the pruning. Therefore, we report the running times as a factor of the number of candidate nodes, while the repository size is fixed (containing all images in Flicker). The results are depicted in Figure 4b. We report that on average, the percentage of image-nodes passing the pruning is 5%, and hence, on average, this phase takes less than 0.3 seconds. We note that the number of keywords in the queries only had a negligible affect on running times, as the same number of random walks are considered for each image-node.

In the next phase, the Memes clusters are generated as well as their corresponding headers. In this experiment, we alter the number of the displayed results, to view its affect on the execution times. The results are depicted in Figure 4c. Not surprisingly, this phase is the fastest one (on average, it took 0.145 seconds) and the execution times grow linearly with the number of displayed results.

Last, we evaluate the overall execution time of the queries mentioned above twice: before the system cached the relevant keywords, and after. According to our experiments, the average execution time of a single user query is 0.47 and 0.75 seconds, for cached and new keyword queries, resp., and the maximal time measured is 0.9 and 1.1 seconds, to cached and new keyword queries, resp.

Concluding, the most "expensive" operation in the evaluation process of a user query is the similarity computation. However, as our experiments indicate, even this phase takes on average 0.65 seconds, and thus allows for an interactive response time. Moreover, since the similarity scores can be computed in parallel, a further speedup can be easily achieved by distributing the computation over multiple cores.

V. Related Work

Internet Memes have become a widespread cultural phenomena [1] and more recently, have even attracted attention from the research community [2], [3]. Despite a growing need for users to quickly find the right Meme for a given search query, to the extent of our knowledge, there is no dedicated system for this task. General image retrieval, on the other hand, has received tremendous attention in roughly three main lines of works: (i) Content Based Image Retrieval (CBIR) [37], [4]; (ii) Text Based or Tag based Image Retrieval (TBIR/TagIR) [5], [6], and (iii) Hybrid approaches [7]. As demonstrated, these approaches are not well suited for the Meme retrieval task: CBIR techniques are focus on analyzing low-level visual features (e.g., color histograms, shapes) that are not easily correlated with high-level semantic concepts prevalent in Memes such as irony and emotions. TBIR frameworks, which associate meta-data (e.g., geo-location) to images, employ text retrieval techniques (e.g., TF/IDF, N-grams), and are highly useful when an elaborate text associated with the image is available. However, this is not a classic case for Memes, as they are typically propagated via image-sharing platforms (e.g., Instagram, Facebook), and thus are associated merely with a handful of tags. More recently, to address this issue, the TagIR approach has been proposed [6], [8]. This approach is mainly based on the relevance scores between the query keywords and the image tags. However, since the tags are often ambiguous and incomplete, additional aspects such as the Meme's caption and background-image are needed to be considered as well. Moreover, it does not support Meme retrieval based on visual features. Our approach follows the line of hybrid solutions [7], [11], [35], [36], which can account for cases where visually similar images may be semantically dissimilar and vice versa. However, as demonstrated, they do not support a tunable assessment of the similarity, allowing a flexible search.

Among the variety of works, our approach is most similar to the following two works. ImageNet [7] (which we have examined in our experiments as a baseline) considers both visual features of the images and a semantic taxonomy of tags. To determine the similarity of two images, the authors of [7] suggested to first retrieve their k-nearest neighbors in terms of visual distance, then return the aggregated semantic distance of the neighbors' categories. In contrast, SimMeme (a) considers the entire structure of the network, (b) interweaves the visual and semantic similarity scores, instead of using them separately, and (c) is also applicable for keyword-based queries. The problem of balancing between tags and visual features was also considered in [12]. The proposed system employs a random-walk based algorithm for image retrieval. However, they use the "classic" SimRank-style random walk definition, as opposed to our *Semantic Aware Random Walks* notation that also accounts for the IC values and the conceptual commonality of tags, as induced by the taxonomy. Here again, our experiments show that our refined variant of random-walks yields better results for Meme search.

Another related research domain is *cross-modal retrieval*, that enables the retrieval of items with multiple modalities (e.g., videos, images, and news articles that describe the same concept or event). Most works in this field are using *representation learning* techniques such as *embedding* and *learning to rank*, to obtain a unified vector-representation for the different data types [38], [8]. While it is possible to use such an approach for Memes retrieval (e.g., DeepWalk [11]), according to our experiments, SimMeme provides more accurate results. Also, it is less flexible in terms of user experience - setting the importance of visual features for a search query is not possible, and its results are difficult to explain (since the similarity is calculated over a machine-generated vector space).

Our similarity measure extends SimRank [13], a popular and common similarity measure for information networks, to better handle heterogeneous, semantically-rich networks. As SimRank was originally defined for unweighted networks, SimRank++, a variant that also considers edge weights was presented in [19]. In this work the authors consider graphs that model sponsored searches, and account for the evidence supporting query similarity. Since no user feedback was available in our setting (e.g., clicks), when examining SimRank++ in our experiments we considered only the weights, which amount to a restricted variant of our measure, ignoring semantics. Moreover, scalability issues for SimRank++ were not studied, as the computations in [19] were mainly performed offline. One of the contributions of the current work is an efficient computation scheme, applicable also for this variant.

Several alternative similarity measures for HIN were presented in previous work [39], [34]. In particular, PathSim [34], a semantic aware measure, was considered, yet, as shown, made limited use of the tags' semantics. SimMeme prototype employs Lin [20], an effective IC-based similarity measure, which, as shown, can be replaced. Much effort was devoted to quantify semantic similarity, especially with the increasing interest in the Semantic Web. Examples include: IC-based measures [22], [23] and feature-based ones [21]. The former regards a domain ontology, while the latter usually considers additional external sources (e.g., textual corpus [21]).

The HIN repository used in our prototype implementation comprises of Memes, tags, visual similarity edges and the WordNet ontology. Nonetheless, it can be enriched with additional machine-generated tags[40], sentiment proximity (textual or visual) [9], and high-level semantic entities, extracted from the images [4].

VI. CONCLUSION AND FUTURE WORK

In this work we presented SimMeme, a Meme-dedicated search engine, which provides a flexible interface enabling users to specify their search criteria. SimMeme employs a novel similarity measure, which interweaves visual, textual and semantic information, and allows for a comprehensive and efficient assessment of Memes similarity. To evaluate the adequacy of the results obtained, we compared our system to baselines which include state-of-the-art approaches for standard image retrieval. Our experiments indicate that SimMeme returns more adequate results to Memes search queries than the competitors, and demonstrated the efficiency of our algorithms.

As Memes typically propagate thorough social networks, an interesting direction for future work is supporting *Personalized Search*, tailoring results to an individual's interests, by incorporating information extracted from her profile. An additional possible application of SimMeme is "Meme-prediction", inspired by answer-generation works, e.g. predicting an adequate reply to an email. Similarly, we can employ SimMeme for generating the appropriate Meme as a reply to a natural language sentence. Another goal we are currently pursuing is to devise a benchmark for Meme retrieval, which can serve future academic research. Last, it would be interesting to adapt our novel measure to other tasks requiring to consider multiple object's attributes such as *entity resolution* and *link prediction*.

REFERENCES

[1] C. Bauckhage, "Insights into internet memes." in *ICWSM*, 2011.

[2] G. Bordogna and G. Pasi, "An approach to identify ememes on the blogosphere," in *WI-IAT*. IEEE, 2012.

[3] O. Tsur and A. Rappoport, "Don't let me be# misunderstood: Linguistically motivated algorithm for predicting the popularity of textual memes." in *ICWSM*, 2015.

[4] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *European conference on computer vision*. Springer, 2014.

[5] S. Kottur, R. Vedantam, J. M. Moura, and D. Parikh, "Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes," in *CVPR*, 2016.

[6] B. Q. Truong, A. Sun, and S. S. Bhowmick, "Casis: a system for concept-aware social image search," in *WWW*. ACM, 2012.

[7] T. Deselaers and V. Ferrari, "Visual and semantic similarity in imagenet," in *CVPR*. IEEE, 2011.

[8] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, and X. Wu, "Visual-textual joint relevance learning for tag-based social image search," *IEEE Transactions on Image Processing*, 2013.

[9] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *CVPR*, 2014.

[10] D. H. Ngo and Z. Bellahsene, "Yam++:(not) yet another matcher for ontology matching task," in *BDA: Bases de Données Avancées*, 2012.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*. ACM, 2014.

[12] J. Urban and J. M. Jose, "Adaptive image retrieval using a graph model for semantic feature integration," in *MIR*. ACM, 2006.

[13] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *SIGKDD*, 2002.

[14] B. Tian and X. Xiao, "Sling: A near-optimal index structure for simrank," in *SIGMOD*. ACM, 2016.

[15] M. Ekron, T. Milo, and B. Youngmann, "Simmeme: Semantic-based meme search," in *CIKM*. ACM, 2017.

[16] T. Milo, A. Somech, and B. Youngmann, "Technical report," http://courses.cs.tau.ac.il/software1/1617a/misc/main-member-full.pdf, 2018.

[17] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *SIGKDD*. ACM, 2015.

[18] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, 1995.

[19] I. Antonellis, H. G. Molina, and C. C. Chang, "Simrank++: query rewriting through link analysis of the click graph," *PVLDB*, 2008.

[20] D. Lin, "An information-theoretic definition of similarity." in *ICML*, 1998.

[21] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *TKDE*, 2003.

[22] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *arXiv preprint cmp-lg/9709008*, 1997.

[23] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *arXiv preprint cmp-lg/9511007*, 1995.

[24] N. Seco, T. Veale, and J. Hayes, "An intrinsic information content metric for semantic similarity in wordnet," in *ECAI*, 2004.

[25] L. Zhu, J. Shen, L. Xie, and Z. Cheng, "Unsupervised visual hashing with semantic assistant for content-based image retrieval," *TKDE*, 2017.

[26] D. Fogaras and B. Rácz, "Scaling link-based similarity search," in *WWW*, 2005.

[27] D. Harel and R. E. Tarjan, "Fast algorithms for finding nearest common ancestors," *siam Journal on Computing*, 1984.

[28] Y. Shao, B. Cui, L. Chen, M. Liu, and X. Xie, "An efficient similarity search framework for simrank over large dynamic graphs," *PVLDB Endowment*, 2015.

[29] B. S. Seah, S. S. Bhowmick, and A. Sun, "Prism: concept-preserving summarization of top-k social image search results," *PVLDB*, 2015.

[30] S. Wang, F. Jing, J. He, Q. Du, and L. Zhang, "Igroup: presenting web image search results in semantic clusters," in *SIGCHI*. ACM, 2007.

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, 2004.

[32] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *TIST*, 2016.

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[34] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, 2011.

[35] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*. WWW, 2015.

[36] X. Jin, J. Luo, J. Yu, G. Wang, D. Joshi, and J. Han, "irin: image retrieval in image-rich information networks," in *WWW*. ACM, 2010.

[37] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He, "Emr: A scalable graph-based ranking model for content-based image retrieval," *TKDE*, 2015.

[38] M. Wang, W. Li, D. Liu, B. Ni, J. Shen, and S. Yan, "Facilitating image search with a scalable and compact semantic mapping," *IEEE transactions on cybernetics*, 2015.

[39] C. Wang, Y. Sun, Y. Song, J. Han, Y. Song, L. Wang, and M. Zhang, "Relsim: Relation similarity search in schema-rich heterogeneous information networks," in *SDM*, 2016.

[40] Y. Cheng, W. Mao, C. Jin, Y. Zhang, X. Huang, and T. Zhang, "Learning tag relevance by context analysis for social image retrieval," in *CCL*. Springer, 2014.