Microsoft Concept Graph: Mining Semantic Concepts for Short Text Understanding (Post-print)

Authors: Ji, Lei, Wang, Yujing, Shi, Botian, Zhang, Dawei, Wang, Zhongyuan, Yan, Jun, Ji, Lei

Date: 2022-11-27T00:00:00+00:00

Abstract

Knowledge is important for text-related applications. In this paper, we introduce Microsoft Concept Graph, a knowledge graph engine that provides concept tagging APIs to facilitate the understanding of human languages. Microsoft Concept Graph is built upon Probase, a universal probabilistic taxonomy consisting of instances and concepts mined from the Web. We start by introducing the construction of the knowledge graph through iterative semantic extraction and taxonomy construction procedures, which extract 2.7 million concepts from 1.68 billion Web pages. We then use conceptualization models to represent text in the concept space to empower text-related applications, such as topic search, query recommendation, Web table understanding and advertisements' relevance. Since the release in 2016, Microsoft Concept Graph has received more than 100,000 page views, 2 million API calls and 3,000 registered downloads from 50,000 visitors over 64 countries.

Full Text

Preamble

Microsoft Concept Graph: Mining Semantic Concepts for Short Text Understanding

Lei Ji[1],[2]†, Yujing Wang[1], Botian Shi[3], Dawei Zhang[4], Zhongyuan Wang[5] & Jun Yan[6]

[1]Microsoft Research Asia, Haidian District, Beijing 100080, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, Beijing 100049, China
[3]Beijing Institute of Technology, Haidian District, Beijing 100081, China

[4]MIX Labs, Haidian District, Beijing 100080, China
[5]Meituan NLP Center, Chaoyang District, Beijing 100020, China
[6]AI Lab of Yiducloud Inc., Huayuan North Road, Haidian District, Beijing 100089, China

Keywords: Knowledge extraction; Conceptualization; Text understanding

Citation: L. Ji, Y. Wang, B. Shi, D. Zhang, Z. Wang & J. Yan. Microsoft concept graph: Mining semantic concepts for short text understanding. *Data Intelligence* 1(2019), 262-294. doi: 10.1162/dint_a_{00013}

Received: November 20, 2018; **Revised:** February 12, 2019; **Accepted:** February 19, 2019

Abstract

Knowledge is crucial for text-related applications. In this paper, we introduce Microsoft Concept Graph, a knowledge graph engine that provides concept tagging APIs to facilitate the understanding of human language. Microsoft Concept Graph is built upon Probase, a universal probabilistic taxonomy consisting of instances and concepts mined from the Web. We begin by describing the construction of the knowledge graph through iterative semantic extraction and taxonomy construction procedures, which extract 2.7 million concepts from 1.68 billion Web pages. We then employ conceptualization models to represent text in the concept space, empowering various text-related applications such as topic search, query recommendation, Web table understanding, and advertisement relevance. Since its release in 2016, Microsoft Concept Graph has received more than 100,000 page views, 2 million API calls, and 3,000 registered downloads from 50,000 visitors across 64 countries.

† Corresponding author: Lei Ji (Email: leiji@microsoft.com; ORCID: 0000-0002-7569-3265)

---

1. Introduction

Concepts are indispensable for both humans and machines to understand the semantic meanings underlying raw text. Humans comprehend an instance—especially an unfamiliar one—through its basic-level concept, a process psychologists and linguists define as Basic-level Categorization (BLC). For example, while people may not understand "Gor Mahia," the concept "football club" from Wikipedia enables them to grasp its semantic meaning easily. Psychologist Gregory Murphy began his highly acclaimed book [1] with the statement "Concepts are the glue that holds our mental world together." A *Nature* book review [2] calls this an understatement, arguing that "Without concepts, there would be no mental world in the first place."

To enable machines to understand instances as humans do, we need a knowledge graph comprising instances, concepts, and their relationships. However, we observe two major limitations in existing knowledge graphs that motivate us to build Probase [3], a brand-new taxonomy for general-purpose human language understanding.

First, previous taxonomies have limited concept space. Many existing taxonomies are constructed by human experts and are difficult to scale. For instance, the Cyc project [4] contains only about 120,000 concepts after 25 years of evolution. Other projects like Freebase [5] rely on crowdsourcing to expand their concept space, yet still lack general coverage across many domains, creating barriers for general-purpose text understanding. Automatically constructed knowledge graphs such as YAGO [6] and NELL [7] also suffer from limited concept coverage. Table 1 compares the number of concepts in Probase and other popular open-domain taxonomies, demonstrating that Probase has a substantially larger concept space.

Second, previous taxonomies treat knowledge as black and white. Traditional knowledge bases aim to provide standard, well-defined, and consistent reusable knowledge, treating knowledge in binary terms. However, this approach has clear limitations, especially when the concept space is extremely large, because different knowledge has varying confidence intervals or probabilities, and the optimal probability threshold depends on the specific application. Unlike previous taxonomies, our philosophy annotates knowledge facts with probabilities and lets the application determine the best way to utilize them. We believe this design is more flexible and can benefit a broader range of text-based applications.

Based on the Probase knowledge taxonomy, we propose a novel conceptualization model to learn text representation in the Probase concept space. The conceptualization model (also known as the Concept Tagging Model) aims to map text into semantic concept categories with associated probabilities. It provides computers with commonsense computing capabilities and makes machines "aware" of the human mental world, enabling them to better understand human communication in text. Using this conceptualization model, the Probase taxonomy can facilitate various text-based applications, including topic search, query recommendation, advertisement relevance, and Web table understanding.

Figure 1 [Figure 1: see original paper] illustrates the overall framework, which consists of three main layers: (1) knowledge construction, (2) conceptualization, and (3) application.

Knowledge Construction Layer. This layer addresses the construction of the Probase knowledge network. First, isA facts are mined automatically from the Web through a semantic iterative extraction procedure. Second, the taxonomy is constructed using a rule-based algorithm. Finally, we calculate probability scores for each instance/concept relationship in the taxonomy.

Conceptualization Layer. Based on the constructed semantic knowledge net-

work, we design a conceptualization model to represent raw text in the Probase concept space. The model comprises three major components: (1) single instance conceptualization, (2) context-aware single instance conceptualization, and (3) short text conceptualization.

Application Layer. The conceptualization model enables us to generate "Bag-of-Concepts"representations of raw text, which can enhance various applications, including but not limited to topic search, query recommendation, ad relevance calculation, and Web table understanding. We also build a Probase browser and a tagging model demo in this layer, providing users with quick insights into specific queries.

The remainder of this paper is organized as follows: Section 2 discusses related work, Section 3 presents the construction of the Probase semantic network, Section 4 introduces the conceptualization model built upon this network, and Section 5 demonstrates example applications. Section 6 provides datasets and statistics, and Section 7 concludes the paper.

---

2. Related Work

Knowledge graphs have attracted significant interest across many research fields. Existing knowledge graphs are built either manually or automatically.

WordNet [8] differs from traditional thesauruses by grouping words based on meaning rather than morphology. Terms are organized into synsets, each representing a distinct concept, with synsets interlinked through conceptual semantics and lexical relations. WordNet contains over 117,000 synsets for 146,000 terms and is widely used for term disambiguation due to its definition of various senses for each term.

DBpedia [10] extracts Wikipedia Infoboxes into knowledge facts that can be semantically queried using SPARQL. This collaboratively edited knowledge base has released 670 million RDF triples, providing an ontology with 280 classes, 3.5 million entities, and 9,000 attributes.

YAGO [6] (Yet Another Great Ontology) is an open-source large-scale semantic knowledge base that fuses knowledge from WordNet, GeoNames, and Wikipedia. YAGO combines WordNet' s taxonomy with Wikipedia categories to assign entities to over 200,000 classes, comprising more than 120 million facts about 3 million entities. Manual evaluation shows YAGO' s accuracy is approximately 95%, with temporal and spatial information attached to entities and relations through declarative extraction rules.

Freebase [5] is a large knowledge base containing substantial human-labeled data contributed by community members. It extracts data from multiple sources including Wikipedia, NNDB, Fashion Model Directory, and MusicBrainz, containing over 125 million facts, 4,000 types, and 7,000 properties. Since 2016, all

Freebase data has been transferred to Wikidata.

ConceptNet [11] is a semantic network built through crowdsourcing to create a large commonsense knowledge base, focusing on commonsense relations including isA, partOf, usedFor, and capableOf. It contains over 21 million edges and 8 million nodes.

NELL [7] is a continuously running system that iteratively extracts facts from hundreds of millions of Web pages, with patterns being boosted during the process. NELL has accumulated over 50 million candidate beliefs and extracted 2,810,379 asserted instances across 1,186 different categories and relations.

WikiTaxonomy [9] automatically generates a taxonomy from Wikipedia categories, producing 105,418 isA links from a network of 127,325 categories and 267,707 links. The system achieves 87.9% balanced F-measure according to manual taxonomy evaluation.

KnowItAll [12] aims to automate the extraction of large collections of facts from the Web in an unsupervised, domain-independent, and scalable manner. Its three major components—Pattern Learning (PL), Subclass Extraction (SE), and List Extraction (LE)—achieve significant improvements in recall while maintaining precision and enhancing extraction rates.

Existing knowledge graphs suffer from low coverage and lack of well-organized taxonomies. Moreover, none focuses on extracting the super-concepts and sub-concepts of an instance. To automatically generate such taxonomies, we propose Probase, which constructs a semantic network of isA facts automatically.

---

3. Semantic Network Construction

This section details the entire data construction procedure for the Probase semantic network. Section 3.1 describes the iterative data extraction process, Section 3.2 introduces the taxonomy construction step, and Section 3.3 presents the probability calculation algorithm.

3.1 Iterative Extraction

The Probase semantic network [3] is built upon isA facts extracted from the Web. isA facts can be formulated as pairs consisting of a super-concept and a sub-concept. For example, "cat isA animal" forms a pair where "cat" is the sub-concept and "animal" is the super-concept. We propose a semantic iteration-based method to mine isA pairs from the Web.

3.1.1 Syntactic vs. Semantic Iteration Prior state-of-the-art information extraction methods [7, 8, 12] rely on syntactic integrative (bootstrapping) approaches. These methods start with seed patterns to discover high-confidence

example pairs, derive new patterns from extracted pairs, and continue iteratively until a stop criterion is met. However, we observe that syntactic iteration methods face fundamental barriers in deep knowledge acquisition because high-quality syntactic patterns are rare. Therefore, we propose a semantic interactive approach that extracts new pairs with high confidence based on knowledge accumulated from existing pairs.

3.1.2 The Semantic Iteration Algorithm We first extract candidate pairs using Hearst patterns [13] (Table 2). For instance, from the sentence "···animals such as cat ···" , we extract the pair <animal, cat>, meaning cat isA animal. Pattern matching sometimes involves ambiguity. In the sentence "···animals other than dogs such as cats ···" , we can extract two candidate pairs: <animals, cats> and <dogs, cats>. The algorithm must determine the correct super-concept between "animals" and "dogs" . Another example is "···companies such as IBM, Nokia, Proctor and Gamble ···" , where we have multiple sub-concept choices: <companies, Proctor> and <companies, Proctor and Gamble>. The algorithm should automatically determine their correctness.

Let Xs denote the candidate set of super-concepts for sentence s, and Ys denote the candidate set of sub-concepts for sentence s. Let Γ be the set of isA pairs already extracted as ground truth. The algorithm' s remaining task is to detect correct super-concepts and sub-concepts from Xs and Ys based on knowledge accumulated in Γ. For each sentence s with multiple super-concept choices, we must select one correct super-concept, based on the observation that only one correct answer exists when ambiguity arises in super-concept matching. After determining the correct super-concept, we filter out correct sub-concepts from candidates in Ys. Unlike the super-concept case, we may expect multiple valid sub-concepts, with results being a subset of Ys.

3.1.3 Super-Concept Detection We compute likelihood p(xk|Ys) for each candidate xk. Assuming independence among sub-concepts given any super-concept xk, we have:

$$p(x_k|Y_s) \propto p(x_k) \prod_{y_i \in Y_s} p(y_i|x_k)$$

where p(xk) is the percentage of pairs in Γ containing xk as super-concept, and p(yi|xk) is the percentage of pairs in Γ having yi as sub-concept when xk is super-concept. For pairs not existing in Γ, we set p(yi|xk) = , where is a small positive number. Without loss of generality, assume x1 and x2 are the top two candidates with highest probability scores, and p(x1|Ys) > p(x2|Ys). We compute the ratio:

$$r(x_1, x_2) = \frac{p(x_1|Y_s)}{p(x_2|Y_s)}$$

x1 is chosen as the correct super-concept if r(x1, x2) exceeds a predefined threshold. Otherwise, the sentence is skipped in the current iteration and may be recovered in later rounds when Γ becomes more informative. Intuitively, p(animals|cats) should be much larger than p(dogs|cats), enabling correct selection of "animals" as the super-concept.

3.1.4 Sub-Concept Detection Our sub-concept detection implementation uses features extracted from original sentences, motivated by two observations:

Observation 1: The closer a candidate sub-concept is to the pattern keyword, the more likely it is a valid sub-concept.

Observation 2: If we are certain that a candidate sub-concept at position k (by distance from the pattern keyword) is valid, then candidate sub-concepts from positions 1 to k−1 are also likely correct.

For example, in the sentence "···representatives in North America, Europe, the Middle East, Australia, Mexico, Brazil, Japan, China, and other countries ···" , "and other" is the pattern keyword and China is the closest candidate sub-concept. Obviously, China is correct. Furthermore, if we know Australia is a valid sub-concept of "countries" , we can be more confident that Mexico, Brazil, and Japan are also correct sub-concepts of the same category, while North America, Europe, and the Middle East are less likely to be instances of that category.

Specifically, we find the largest k such that likelihood p(yk|x) exceeds a predefined threshold, then treat y1, y2, ···, yk as valid sub-concepts of super-concept x. Note that sub-concept yi sometimes involves ambiguity. In the sentence "··· companies such as IBM, Nokia, Proctor and Gamble ···" at position 3, the noun phrase could be "Proctor" or "Proctor and Gamble" . Suppose candidates at position k are c1, c2, ···We calculate each candidate' s conditional probability given super-concept x and all previous sub-concepts:

$$p(c_i|x, y_1, ..., y_{k-1}) \propto p(c_i|x) \prod_{j=1}^{k-1} p(y_j|c_i, x)$$

As before, y1, y2, ···, yk–1 are assumed independent given super-concept x. p(ci|x) is the percentage of existing pairs in Γ where ci is a sub-concept when x is super-concept. p(yj|ci, x) is the likelihood that yj is a valid sub-concept given x as super-concept and ci as another valid sub-concept in the same sentence. If c1 and c2 are the top two ranked concepts, we select c1 as the final concept if r(c1, c2) exceeds a certain ratio, where:

$$r(c_1, c_2) = \frac{p(c_1|x, y_1, ..., y_{k-1})}{p(c_2|x, y_1, ..., y_{k-1})}$$

Intuitively, p(Proctor and Gamble|companies) should be much larger than p(Proctor|companies) after Γ accumulates sufficient knowledge, enabling the algorithm to automatically select the correct candidate.

3.2 Taxonomy Construction

The taxonomy construction procedure consists of three steps: (1) local taxonomy construction, (2) horizontal merging, and (3) vertical merging. We first build a local taxonomy for each sentence, then perform horizontal and vertical merging sequentially on the collection to construct a global taxonomy.

Local Taxonomy Construction. Figure 2 [Figure 2: see original paper] illustrates building a local taxonomy from each sentence. For example, from sentence "A such as B, C, D" , we extract three pairs <A, B>, <A, C>, and <A, D> where A is super-concept and B, C, D are sub-concepts. We build a local taxonomy with A as root node and B, C, D as child nodes.

Horizontal Merging. After building local taxonomies for each sentence, we perform horizontal merging. Suppose we have two local taxonomies T1 and T2 rooted by A1 and A2 respectively, where A1 and A2 correspond to the same surface form. If we are confident that A1 and A2 express the same sense, we can merge the two trees horizontally (Figure 3 [Figure 3: see original paper]). We design a similarity function to calculate the probability that A1 and A2 are semantically equal. Intuitively, greater overlap between children of A1 and A2 increases confidence that they share the same sense. Therefore, we adopt absolute overlap for similarity calculation: sim(A1, A2) = |children(A1) children(A2)|, using a constant threshold d to determine whether two local taxonomies can be horizontally merged.

Vertical Merging. Given two local taxonomies rooted by Ai and B1, where B is a child of Ai, if we are confident that B shares the same sense as B1, we can merge the two taxonomies vertically. As shown in Figure 4 [Figure 4: see original paper], after merging, Ai becomes the root, the original subtree B merges with the taxonomy rooted by B1, and other subtrees C and D remain in their positions. To determine if B and B1 share the same sense, we calculate the absolute overlap between R1' s children and B' s siblings (highlighted nodes in Figure 4).

A more complicated case is shown in Figure 5 [Figure 5: see original paper]. Both T1 and T2 have child nodes of R1 and R2 with considerable overlap with B' s siblings in Ti, so both can be vertically merged with Ti. However, child nodes of R1 and R2 lack sufficient overlap, indicating R1 and R2 may express different senses. In this case, we split the two senses in the merged taxonomy, merging T1 and T2 as two distinct subtrees in taxonomy Ti.

3.3 Probability Calculation

Probability is a crucial feature of our taxonomy design. Unlike previous approaches that treat knowledge as black and white, our solution embraces noisy data with probability scores and lets applications make optimal use of it. We provide two probability scores: plausibility and typicality.

Plausibility is the joint probability of an isA pair. For each isA claim E, we use p(E) to denote its probability of being true. We adopt a simple noisy-or model for calculation. Assuming E is derived from sentences {s1, s2, ···, sn}, a claim is false only if every piece of evidence from {s1, s2, ···, sn} is false. With independent evidence pieces:

$$p(E) = 1 - \prod_{i=1}^{n}(1 - p(s_i))$$

where p(si) is the probability of evidence si being true. We characterize each si by features Fi including: (1) PageRank score of the Web page containing si; (2) the Hearst pattern used to extract <x, y> from si; (3) number of sentences with x as super-concept; (4) number of sentences with y as sub-concept; (5) number of sub-concepts extracted from si; (6) position of y in the sub-concepts list from si. Assuming feature independence, we apply Naïve Bayes [14] to derive p(si) from Fi. We exploit WordNet to build a training set: for pair <x, y> appearing in WordNet, if a path exists between x and y (i.e., x is an ancestor of y), we treat it as positive; otherwise negative.

Typicality is the conditional probability between a super-concept and its instance (sub-concept). Intuitively, robin is more typical of "bird" than ostrich, and Microsoft is more typical of "company" than Xyz Inc. Therefore, we need a probability score representing instance typicality to its super-concept. The typicality measure of instance i to super-concept x is:

$$T(i|x) = \frac{n(x,i) \cdot p(x,i)}{\sum_{j \in I_x} n(x,j) \cdot p(x,j)}$$

where x is super-concept, i is an instance of x, n(x, i) is the number of appearances supporting i as an instance of x, p(x, i) is the plausibility of pair <x, i>, and Ix is the set of all instances of super-concept x. For example, with x = Company, i = Microsoft, j = Xyz Inc, n(x, i) should be much larger than n(x, j), giving Microsoft a much higher typicality score than Xyz Inc for Company. Similarly, we define typicality representing the probability of concept x to instance i:

$$T(x|i) = \frac{n(x,i) \cdot p(x,i)}{\sum_{c \in C_i} n(c,i) \cdot p(c,i)}$$

4. Conceptualization

This section introduces the conceptualization model that leverages the Probase semantic knowledge network to facilitate text understanding. The conceptualization model (also known as the Concept Tagging Model) aims to map text into semantic concept categories with probabilities. It provides computers with semantic commonsense and makes machines "aware" of the human mental world, enabling better understanding of human communication in text.

We consider three sub-tasks: (1) **Single Instance Conceptualization**, which returns BLC for a single instance (e.g., "Microsoft" maps to Software Company and Fortune 500 company with prior probabilities); (2) **Context-Aware Single Instance Conceptualization**, which produces appropriate concepts based on context (e.g., "Apple" maps to Fruit or Company without context, but to Fruit with higher probability given context "pie"); (3) **Short Text Conceptualization**, which returns types and concepts for short text sequences (e.g., in "He is playing game on Apple iPhone and eating an apple," the first "Apple" is Company while the second is Fruit).

4.1 Single Instance Conceptualization

Assume e is an instance and c is a super-concept. We obtain BLC results based on typicality scores P(e|c) and P(c|e), where P(e|c) denotes instance e' s typicality to concept c, and P(c|e) denotes concept c' s typicality to instance e. We propose several representative metrics for BLC [15].

MI is the mutual information between e and c:

$$MI(e,c) = \sum_{e,c} P(e,c) \log \frac{P(e,c)}{P(e)P(c)}$$

PMI denotes pointwise mutual information, widely used to measure association between discrete variables:

$$PMI(e,c) = \log \frac{P(e,c)}{P(e)P(c)}$$

NPMI is a normalized PMI version, proposed to make PMI less sensitive to occurrence frequency and more interpretable:

$$NPMI(e,c) = \frac{PMI(e,c)}{-\log P(e,c)}$$

Both PMI and NPMI suffer from a trade-off between general and specific concepts. General concepts may be correct but cannot distinguish sub-category

instances, while specific concepts may be more representative but have low coverage. Therefore, we propose PMIk and Graph Traversal measures to address this.

PMIk compromises to avoid overly general or specific concepts:

$$Rep(e, c) = \frac{P(e, c)^2}{P(e)P(c)}$$

Taking the logarithm yields:

$$\log Rep(e, c) = 2 \log P(e, c) - \log P(e) - \log P(c) = PMI_2(e, c)$$

This corresponds to PMI2, a normalized form in the PMIk family [16].

Graph Traversal calculates relatedness between two nodes in a large network. Scores from general graph traversal can be formulated as:

$$Time(e, c) = \sum_{k=1}^{\infty} P_k(e, c)$$

where Pk(e, c) is the probability of starting from e to c and back to e in 2k steps. For k = 2 (4-step random walk):

$$Time(e, c) = 2 \cdot P_2(e, c) = 4 \cdot P(e, c) \cdot (1 - P(e, c))$$

Thus, the simple, easy-to-compute Rep(e, c) method is equivalent to graph traversal under the 4-step random walk constraint.

4.2 Context-Aware Single Instance Conceptualization

One instance may map to distinct concepts depending on context. For "apple ipad," we want to annotate "apple" as company/brand and "ipad" as device/product. The major challenge is distinguishing and detecting correct concepts in different contexts. We propose a framework for context-aware single instance conceptualization [17] comprising offline knowledge graph construction and online concept inference.

4.2.1 Offline Knowledge Graph Construction The Probase semantic network contains millions of concepts. We first cluster all concepts into 5,000 clusters using a K-Medoids algorithm [18]. We use concept clusters instead of individual concepts for noise reduction and computational efficiency. In the following, "concept" denotes concept cluster. We build a large offline knowledge graph comprising three sub-graphs: (1) instance-to-concept, (2) concept-to-concept, and (3) non-instance term-to-concept. Figure 6 [Figure 6: see original paper] shows a subgraph around the term "watch."

Instance-to-concept graph. We directly use the Probase semantic network, where P(c|e) serves as the typicality probability of concept (cluster) c to instance e:

$$P(c|e) = \sum_{c^* \in c} P^*(c^*|e)$$

where c* is a concept in cluster c and P(c|e) is c*' s typicality to e.

Concept-to-concept graph. We assign transition probability P(c2|c1) to edges between concepts c1 and c2, derived by aggregating co-occurrences between all unambiguous instances of the two concepts:

$$P(c_2|c_1) = \frac{\sum_{e_1 \in c_1, e_2 \in c_2} n(e_1, e_2)}{\sum_{e_1 \in c_1, e_2 \in \mathcal{E}} n(e_1, e_2)}$$

where $\mathcal{E}$ is all instances and the denominator normalizes the probability. In practice, we only use the top 25 related concepts for each concept.

Non-instance term-to-concept graph. Some terms cannot match any instances or concepts (e.g., verbs, adjectives). To better understand short text, we mine lexical relationships between non-instance terms and concepts. We use the Stanford Parser [19] for POS tagging and dependency relationships, revealing each token' s type (e.g., adjective, verb). Our goal is to obtain two distributions:

P(z|t) represents the prior probability that term t is of type z. For example, "watch" is a verb with probability P(verb|watch) = 0.8374. We compute:

$$P(z|t) = \frac{n(t, z)}{n(t)}$$

where n(t, z) is the frequency of term t annotated as type z, and n(t) is t' s total frequency.

P(c|t, z) denotes the probability of concept c given term t of specific type z. For example, P(movie|watch, verb) depicts how likely the verb "watch"is lexically associated with concept movie. We detect co-occurrence relationships between instances, verbs, and adjectives in Web documents parsed by the Stanford Parser, requiring dependency-based rather than mere sentence-level co-occurrence. We first obtain P(e|t, z), representing how likely term t of type z co-occurs with instance e:

$$P(e|t, z) = \frac{n_z(e, t)}{n_z(t)}$$

where nz(e, t) is the frequency of term t and instance e having a dependency relationship when t is type z. Then, using instances as bridges, we calculate relatedness between non-instance terms (adjectives/verbs) and concepts:

$$P(c|t,z) = \sum_{e \in c} P(e|t,z) \cdot P(c|e)$$

In Equation (18), e c means e is an instance of concept c, assuming concept c is conditionally independent of term t and type z given instance e.

4.2.2 Online Concept Inference We use the offline-built heterogeneous semantic graph to annotate query concepts. First, we segment the query into candidate terms using Probase as lexicon and identify all term occurrences. Second, we retrieve the subgraph from the heterogeneous semantic graph focusing on query terms. Finally, we perform multiple random walks on the subgraph to find concepts with highest weights after convergence.

4.3 Short Text Conceptualization

Short text is difficult to understand for three reasons: (1) Unlike long sentences, short text lacks syntactic features and cannot directly apply POS tagging; (2) Short text has insufficient statistical signals; (3) Short text is usually ambiguous due to lack of context. While many works focus on statistical approaches like topic models [20] to extract latent topics as implicit semantics, we argue that semantic knowledge is needed for better short text understanding. We aim to utilize the semantic knowledge network to enhance short text understanding [21].

Unlike the knowledge graph for context-aware single instance conceptualization (Section 4.2), we add a novel sub-graph: typed-term co-occurrence graph—an undirected graph where nodes are typed-terms and edge weights represent lexical co-occurrence. However, the enormous number of typed-terms increases storage costs and affects computational efficiency. Therefore, we compress the original typed-term co-occurrence network by retrieving Probase concepts for each instance, replacing typed-terms with related concept clusters and aggregating edge weights accordingly. Figure 7 [Figure 7: see original paper] shows this compression.

Given short text, each term is associated with a type and corresponding concepts. Types include instance, verb, adjective, and concept. For each instance-type term, we also learn its concepts. The online inference procedure involves three steps: text segmentation, type detection, and concept labeling. Figure 8 [Figure 8: see original paper] illustrates the example "book disneyland hotel california." The algorithm first segments it as "book disneyland hotel california,"then detects each segment' s type, and finally labels concepts for each instance. The output "book[v] disneylande hotel[c] californiae" indicates: book is a verb, disneyland is a park instance, hotel is a concept, and california is a state instance.

4.3.1 Text Segmentation Text segmentation divides short text into meaningful components. We define two heuristic rules: (1) Except for stop words, each word belongs to exactly one term; (2) Terms must be coherent (mutually reinforcing). Intuitively, {april in paris lyrics} is better than {april in paris lyrics} for "april in paris lyrics" because "lyrics" is more semantically related to songs than months or cities. Similarly, {vacation april in paris} is better than {vacation april in paris} for "vacation april in paris" because "vacation," "april," and "paris" are highly coherent.

The algorithm proceeds as follows: First, construct a term graph (TG) with candidate terms and relationships. Add edges between non-mutually-exclusive candidate terms, weighting them to reflect mutual reinforcement strength. Finally, transform best segmentation finding into a clique-finding task in TG, maximizing average edge weights while achieving 100% word coverage.

4.3.2 Type Detection Type detection annotates each term as verb, adjective, instance, or concept. For example, "watch" appears in instance-list, concept-list, and verb-list, yielding possible typed-terms watch[c], watch[e], and watch[v]. The algorithm determines the best typed-term from candidates. In "watch free movie," the best typed-terms are watch[v], free[adj], and movie[c].

Traditional approaches use POS tagging algorithms considering only lexical features (e.g., Markov Model [22]), which are insufficient for short text type determination. Our solution calculates probability using both lexical and semantic coherence features, formulating type detection as a graph model with two variants: Chain model and Pairwise model.

Chain model adopts first-order bilexical grammar, considering topical coherence between adjacent typed-terms. It builds a chain-like graph with typed-term nodes, adding edges between adjacent terms and weighting them by affinity scores (Figure 9 Figure 9: see original paper). However, it only considers consecutive term relations, leading to mistakes.

Pairwise model adds edges between typed-terms from all term pairs, not just adjacent ones. In Figure 9(b), edges exist between non-adjacent "watch" and "movie." The goal is finding the typed-term sequence whose maximum spanning tree (MST) has the largest total weight. In Figure 9(b), as long as edges (watch[v], movie[c]) and (free[adj], movie[c]) have the largest total weight, {watch[v], free[adj], movie[c]} is successfully recognized as the best type detection sequence for "watch free movie."We employ the Pairwise model as it achieves higher experimental accuracy than the Chain model.

4.3.3 Concept Labeling Concept labeling re-ranks candidate concepts by context for each instance. The main challenge is handling ambiguous instances. Our intuition: a concept is appropriate for an instance only if it is both a common semantic concept of that instance and supported by surrounding context. In "hotel California eagles," both animal and music band are popular concepts

for "eagles," but finding concept song in context increases confidence that music band is correct.

After type detection, we have a set of instances for concept labeling. For each instance, we collect concept candidates and perform disambiguation using a Weighted-Vote approach combining Self-Vote and Context-Vote. Self-Vote is the original affinity weight (normalized co-occurrence) between concept cluster c and target instance. Context-Vote leverages affinity weights between target instance and other context concepts. For "hotel california eagles," eagles' original concept vector is (<animal, 0.2379>, <band, 0.1277>, <bird, 0.1101>, <celebrity, 0.0463>, ···), while context "hotel california" yields (<singer, 0.0237>, <band, 0.0181>, <celebrity, 0.0137>, <album, 0.0132>, ···). After Weighted-Vote disambiguation, eagles' final conceptualization is (<band, 0.4562>, <celebrity, 0.1583>, <animal, 0.1317>, ···).

---

5. Application

Probase Browser displays the Probase taxonomy backbone and provides an interface to search for super-concepts, sub-concepts, and similar concepts for a given query. Figure 10 [Figure 10: see original paper] shows a browser snapshot.

Tagging Service provides text tagging capability with concept vectors, enabling semantic similarity calculation and various text processing applications. Figure 11 [Figure 11: see original paper] shows a single instance conceptualization demo for "python." Figure 12 [Figure 12: see original paper] shows context-aware results for "apple" with contexts "pie" and "ipad," demonstrating correct concept mapping under different contexts. Figure 13 [Figure 13: see original paper] illustrates short text conceptualization for "apple engineer is eating the apple," showing the algorithm' s ability to distinguish different "apple" senses.

Topic Search [23] aims to understand underlying topics in queries for better search relevance. Figure 14 [Figure 14: see original paper] shows a query for "database conference in Asian cities" correctly ranking VLDB 2002, 2006, and 2010 (held in Hong Kong, Seoul, and Singapore). Traditional Web search treats queries as keyword sequences without semantic understanding, making correct answers difficult. Our framework improves Web search using Probase knowledge and conceptualization models by classifying queries into patterns based on concepts, entities, and keywords, then interpreting queries with Probase semantic concepts. Preliminary results show the framework understands various topic-like queries and achieves much higher user satisfaction.

Understanding Web Tables [24] uses Probase to interpret Web tables, unlocking information hidden in Web pages. We built a pipeline to detect Web tables, understand their contents, and apply derived knowledge to semantic search. From 300 million Web documents, we extracted 1.95 billion raw HTML

tables. Many lack useful relational information (e.g., used for page layout) or have overly complex structures. Rule-based filtering yielded 65.5 million tables (3.4%) with potentially useful information. We use Probase taxonomy to associate table content with semantic concepts, building a semantic search engine over tables. Figure 15 [Figure 15: see original paper] shows that querying "American politicians birthday" returns aggregated Web tables with birthday and related knowledge for various American politicians.

Channel-based Query Recommendation [25] anticipates user search needs when browsing different channels by recommending hot, highly related queries. Figure 16 [Figure 16: see original paper] shows News, Sports, and Entertainment channels with recommended queries for exploring hot topics. A major challenge is representing short text queries and channels. Traditional bag-of-words representation suffers from surface form mismatches, especially with short text. We leverage Probase taxonomy to obtain "Bag of Concepts" representations, avoiding surface mismatching and handling synonyms and polysemy. Using Probase' s large taxonomy, we learn a concept model for each category and conceptualize short text into relevant concepts. A concept-based similarity mechanism classifies short text into the most similar category. Experiments show our framework maps queries to channels with high precision (average precision = 90.3%).

Ads Relevance [3] in sponsored search maps queries to related ad bidding keywords. Since both queries and bidding keywords are short, traditional bag-of-words approaches perform poorly. We leverage Probase concept taxonomy to enhance relevance calculation. For each short text, we identify instances, map them to basic-level concepts with score Rep(e, c), merge concepts into a concept vector representing text semantics, and calculate relevance through cosine similarity between query and bidding keyword concept vectors. Experiments on real ad click logs from Microsoft Bing divided candidate query-bidding pairs into 10 buckets by relevance score. Figure 17 [Figure 17: see original paper] shows CTR has strong linear correlation with relevance scores from our model.

---

6. Data and Statistics

6.1 Data

Microsoft Concept Graph is available at https://concept.research.microsoft.com/, representing a subgraph of our semantic network. The core taxonomy contains over 5.4 million concepts, with distribution shown in Figure 18 [Figure 18: see original paper] (Y-axis: number of instances per concept on logarithmic scale; X-axis: 5.4 million concepts ordered by size). Our concept space far exceeds other knowledge bases (Freebase: $ $2,000 concepts; Cyc: ~120,000 concepts).

6.2 Concept Coverage

Using Microsoft Bing query logs, we estimate concept coverage. A query is considered covered if at least one concept is found. We calculate coverage percentage for Probase and compare against four other taxonomies using the top 50 million Bing queries. Figure 19 [Figure 19: see original paper] shows Probase has the largest coverage, YAGO ranks second, and Freebase has low coverage despite its large instance space, demonstrating its skewed instance distribution (most instances in several top categories lacking general coverage).

6.3 Precision of isA Pairs

To estimate Probase isA pair correctness, we created a benchmark dataset of 40 concepts across various domains. For each concept, we randomly selected 50 instances (sub-concepts) for human evaluation. Figure 20 [Figure 20: see original paper] shows precision for all 40 concepts, with overall precision of 92.8% averaged across concepts.

Figures 21 [Figure 21: see original paper] and 22 [Figure 22: see original paper] show average precision and discovered concepts/isA pairs after each iteration. Precision degrades while discovered concepts and pairs increase with each round, requiring a trade-off between precision and fact count (we set iteration number to 11).

6.4 Conceptualization Experiment

6.4.1 Single Instance Conceptualization **Dataset preparation.** Human labelers manually annotated concept correctness with four categories (Table 3). Each (instance, concept) pair was annotated by three labelers, with majority vote determining the final label (a fourth annotator broke ties). We collected 5,049 labeled records.

Measurement. We use Precision@k and nDCG. For Precision@k, Good/Excellent = 1, Bad/Fair = 0:

$$Precision@k = \frac{\sum_{i=1}^{k} rel_i}{k}$$

For nDCG, Bad = 0, Fair = 1, Good = 2, Excellent = 3:

$$nDCG@k = \frac{DCG@k}{IDCG@k}, \quad DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where reli is the relevance score at rank i, and IDCG uses ideal sorted order.

Experimental result. Figure 23 [Figure 23: see original paper] evaluates top 20 results using Precision and nDCG with and without smoothing, comparing

our scoring functions against baselines MI, NPMI, and PMI3 (defined as PMI3(e, c) = log P(e, c)^3 / (P(e)P(c))). Our scoring function outperforms baselines on both precision and nDCG.

6.4.2 Short Text Conceptualization **Dataset preparation.** We built two datasets: queries and tweets. For queries, we manually selected 11 ambiguous terms: "apple," "fox" (instance ambiguity); "watch," "book," "pink," "blue," "population," "birthday" (type ambiguity); and "April in Paris," "hotel California" (segmentation ambiguity). We randomly selected 1,100 queries containing these terms plus 400 unrestricted queries (1,500 total). For tweets, we randomly sampled 1,500 tweets via Twitter' s API, removing @username, hashtags, and URLs. Human labelers annotated correctness with at least three labelers per record, using majority vote.

Experimental result. We compared our method with baselines: [27] (instance disambiguation based on similar instances), [28] (based on related instances), and [26] (based on both in tweets). Table 4 shows our conceptualization is both effective and robust.

---

7. Conclusion

This paper presents the end-to-end framework for building and utilizing Microsoft Concept Graph, comprising three layers: semantic network construction, conceptualization, and applications. In the semantic network construction layer, we focus on knowledge extraction and taxonomy construction to build Probase, an open-domain probabilistic taxonomy. Like human mental processes, we represent text in concept space using conceptualization models, empowering applications including topic search, query recommendation, ad relevance calculation, and Web table understanding. Since its 2016 release, the system has received wide public attention: over 100,000 page views, 2 million API calls, and 3,000 registered downloads from 50,000 visitors across 64 countries.

Author Contributions

All authors contributed equally. J. Yan (jun.yan@yiducloud.cn), Z. Wang (wzhy@outlook.com), and D. Zhang (zhangdawei@outlook.com) previously led the Microsoft Concept Graph system; L. Ji (leiji@microsoft.com) currently leads development of V2.0 components. Y. Wang (Yujing.Wang@microsoft.com) and L. Ji summarized the methodology. Y. Wang and B. Shi (botianshi@bit.edu.cn) summarized applications and experiments. All authors made meaningful contributions to revising and proofreading.

References

[1] G. Murphy. *The big book of concepts*. Cambridge, MA: The MIT Press, 2004. isbn: 9780262632997.
[2] P. Bloom. Glue for the mental world. *Nature* 421 (2003), 212–213. doi: 10.1038/421212a.
[3] W. Wu, H. Li, H. Wang, & K. Zhu. Probase: A probabilistic taxonomy for text understanding. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 217–228. doi: 10.1145/2213836.2213891.
[4] D.B. Lenat, & R.V. Guha. *Building large knowledge-based systems: Representation and inference in the Cyc project*. Reading, MA: Addison-Wesley, 1990. isbn: 9780201517521.
[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, & J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008, pp. 1247–1250. doi: 10.1145/1376616.1376746.
[6] F.M. Suchanek, G. Kasneci, & G. Weikum. YAGO: A core of semantic knowledge. In: *Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 697–706. doi: 10.1145/1242572.1242667.
[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, & T.M. Mitchell. Toward an architecture for never-ending language learning. In: *Proceedings of the 24th Conference on Artificial Intelligence*, 2010, pp. 1306–1313. Available at: https://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1879/2201.
[8] G.A. Miller. WordNet: A lexical database for English. *Communications of the ACM* 38(11) (1995), 39–41. doi: 10.1145/219717.219748.
[9] S.P. Ponzetto, & M. Strube. Deriving a large-scale taxonomy from Wikipedia. In: *Proceedings of the 22nd National Conference on Artificial Intelligence*, 2007, pp. 1440–1445. Available at: http://www.aaai.org/Papers/AAAI/2007/AAAI07-228.pdf.
[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, & Z. Ives. DBpedia: A nucleus for a Web of open data. In: *Proceedings of the 6th International Semantic Web Conference/Asian Semantic Web Conference*, 2007, pp. 722–735. doi: 10.1007/978-3-540-76298-0_{52}.
[11] R. Speer, & C. Havasi. Representing general relational knowledge in ConceptNet 5. In: *Proceedings of the 8th International Conference on Language Resources and Evaluation*, 2012, pp. 3679–3686. Available at: http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072_{Paper}.pdf.
[12] O. Etzioni, M. Cafarella, D. Downey, A.P. Shaked, S. Soderland, D.S. Weld, & A. Yates. Web-scale information extraction in knowitall: Preliminary results. In: *Proceedings of the 13th International Conference on World Wide Web*, 2004, pp. 100–110. doi: 10.1145/988672.988687.
[13] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In: *Proceedings of the 14th Conference on Computational Linguistics*, 1992, pp. 539–545. doi: 10.3115/992133.992154.
[14] A. McCallum, & K. Nigam. A comparison of event models for naive bayes

text classification. In: *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998, pp. 41–48.
[15] Z. Wang, H.X. Wang, J. Wen, & Y. Xiao. An inference approach to basic level of categorization. In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM)*, 2015, pp. 653–662. doi: 10.1145/2806416.2806533.
[16] B. Daille. Approche mixte pour l' extraction de terminologie: Statistique lexicale et filtres linguistiques. PhD dissertation, Université Paris VII, 1994.
[17] Z. Wang, K. Zhao, H. Wang, X. Meng, & J. Wen. Query understanding through knowledge-based conceptualization. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 3264–3270.
[18] P. Li, H. Wang, K. Zhu, Z. Wang, & X. Wu. Computing term similarity by large probabilistic isA knowledge. In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, 2013, pp. 1401–1410. doi: 10.1145/2505515.2505567.
[19] D. Marneffe, B. MacCartney, & C.D. Manning. Generating typed dependency parses from phrase structure parses. In: *Proceedings of the 5th International Conference on Language Resources and Evaluations*, 2006, pp. 449–454.
[20] D. Blei, A.Y. Ng, & M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research* 3(Jan) (2003), 993–1022. Available at: http://jmlr.csail.mit.edu/papers/v3/blei03a.html.
[21] W. Hua, Z. Wang, H. Wang, K. Zheng, & X. Zhou. Short text understanding through lexical-semantic analysis. In: *IEEE 31st International Conference on Data Engineering*, 2015, pp. 495–506. doi: 10.1109/ICDE.2015.7113309.
[22] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language* 6(3) (1992), 225–242. doi: 10.1016/0885-2308(92)90019-Z.
[23] Y. Wang, H. Li, H. Wang, & K.Q. Zhu. Toward topic search on the Web. Technical report, Microsoft Research, 2010. Available at: https://www.microsoft.com/en-us/research/publication/toward-topic-search-on-the-web/.
[24] J. Wang, H. Wang, Z. Wang, & K.Q. Zhu. Understanding tables on the Web. In: *Proceedings of the 31st International Conference on Conceptual Modeling*, 2012, pp. 1–14. doi: 10.1007/978-3-642-34002-4_{11}.
[25] F. Wang, Z. Wang, Z. Li, & J. Wen. Concept-based short text classification and ranking. In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, 2014, pp. 1069–1078. doi: 10.1145/2661829.2662067.
[26] P. Ferragina, & U. Scaiella. TAGME: On-the-fly annotation of short text fragments (by wikipedia entities). In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010, pp. 1625–1628. doi: 10.1145/1871437.1871689.
[27] Y. Song, H. Wang, Z. Wang, H. Li, & W. Chen. Short text conceptualization using a probabilistic knowledge-base. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011, pp. 2330–2336.

doi: 10.5591/978-1-57735-516-8/IJCAI11-388.
[28] D. Kim, H. Wang, & A. Oh. Context-dependent conceptualization. In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013, pp. 2654–2661.

Author Biography

Lei Ji is a researcher at Microsoft Research Asia. She received Bachelor' s and Master' s degrees from Beijing Institute of Technology and is currently a PhD student in Computer Science at the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include knowledge graphs and cross-modality visual and language learning.

Yujing Wang is a researcher at Microsoft Research Asia. She received Bachelor' s and Master' s degrees from Peking University. Her research interests include AutoML, text understanding, and knowledge graphs.

Botian Shi is a PhD student in Computer Science at Beijing Institute of Technology, having conducted this work during an internship at Microsoft Research Asia. His research interests include natural language processing and multimodal knowledge-based image/video understanding.

Dawei Zhang is Founder of MIX Labs. He received his Master' s degree from Peking University. His research interests include machine learning, natural language processing, and knowledge graphs.

Zhongyuan Wang is a senior researcher and Senior Director at Meituan-Dianping. He received Bachelor' s, Master' s, and PhD degrees in Computer Science from Renmin University of China in 2007, 2010, and 2015, respectively. His research interests include knowledge bases, Web data mining, online advertising, machine learning, and natural language processing.

Jun Yan is Chief AI Scientist at Yiducloud. He received his PhD from the School of Mathematical Science at Peking University. His research interests include medical AI research, knowledge graph mining, and online advertising.

Note: Figure translations are in progress. See original paper for figures.