Knowledge Graph based Query Processing by matching words to entities using Wikipedia

CSE 373 Project Retrospective

Guanxuan Wu

University of Washington

guanxw@uw.edu

December 16, 2021

Abstract

The thirty-years development of Search Engines could never set apart the fundamental problem: to improve relativity for the page rankings with the given query. As the NLP is now widely used, this paper discusses a data abstraction via knowledge graph (KG) for NLP models that could be applied in relating keywords to entities with a higher probability.

1 Introduction to Knowledge Graphs

Knowledge graphs have long been used in Natural Language Processing, with previous works such as a convolution with k-means [3] are widely applied in the data industry, for instance, creating a thesis search engine [4] and preservation of Chinese cultural heritages [6].
In the previous definitions for knowledge graphs [10], those data structures are usually defined as some directed graphs with "labels" on each edge, which requires a further abstraction to make it applicable in practical algorithms. In this paper, we will define a new abstraction for the relational data that is related to such keywords. To begin with, here is a simple, abstract definition that is derived from a previous construction that was applied in deep NLP [12]:

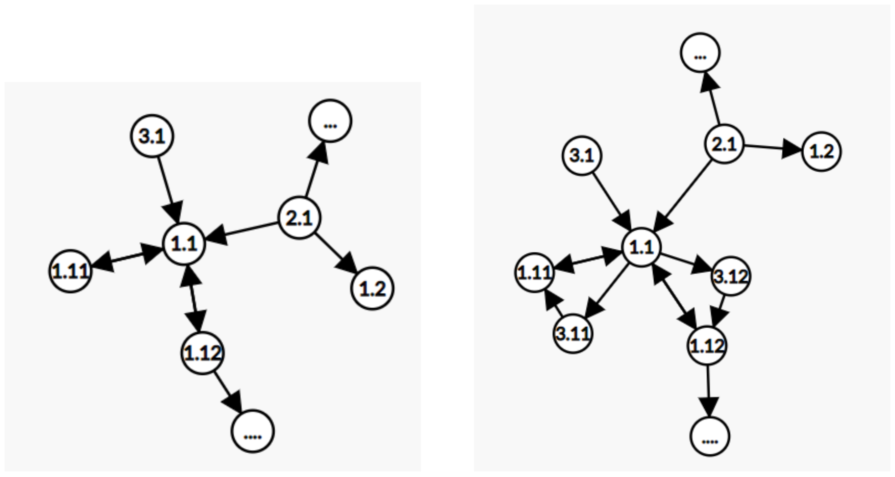
Definition 1. *A knowledge graph (KG) is a directed graph $G(V, E, w : E \to [0, 1])$ such that:*
1. Each vertex corresponds to a keyword, that is a "word" or "phrase";
2. Each edge have a weight that is defined by the number and authority values of the sources that connected two items with; as each vertex that sends at least one edge out sends a limited number of directed edges, it can be normalized with $w_i = \frac{W_i}{\sum_j W_j}$ with W being the originally calculated value.

Such skeleton definition is shared by all such applications with knowledge graphs, however, with such an obscurely defined weight function w, we are not able to conclude anything. So in the rest of this paper, we will focus on constructing and improving such a knowledge graph for our given applications.

2 Construct a Wikipedia Entry Graph

2.1 Simple Case

Here we first discuss a simple example about the query analysis for Wikipedia, which could be inspiring since Wikipedia is a structured semantic knowledge base [14]. To decrease complexity, we consider each of the entry titles as an atomic keyword (vertex) and then we can roughly divide the entries of Wikipedia into three categories:



(a) Without Interpolatory Vertices (b) With Interpolatory Vertices 3.11, 3.12

Fig. 1: An incomplete graph representation of pages "UW"(2.1), "University of Washington"(1.1), "Udub"(3.1), "Ana Mari Cauce"(1.11), "Seattle"(1.12)

Definition 2. *Entity pages, which are usually long, complete pages with a full description of some entity (such as a person, organization, event, etc.) for human and machines for reference, and having plenty of links to other pages; the weight of edges from this kind of pages to other pages could be determined by article semantics and count of links. [8] (Example: "University of Washington")*

Definition 3. *Disambiguation pages, which are short pages with a list of links corresponding to entity pages; such pages usually have entry titles as an acronym of entities, we can just set the weights to each page equally $1/n$ in our naive approach, where n is the total number of entities that the acronym could refer to. (Example: UW)*

Definition 4. *Redirection pages simply redirect to entity pages, which can be denoted as vertices that have only one edge connecting with the redirected page with weight 1. (Example: Udub).*

By this construction, for each $v_i \in V$, we have a set of w_{v_i} functions. So by this, we can also get the relativity of any two vertices: consider the maximum $s-t$ flow $f_{\max}(s,t)$ [7] of the network that constructed by letting each of the weights of edges being the capacity of such edge. Then $f_{\max}$ is an indicator for co-relativity between some pair of vertices.
Besides, there is an acquirable and strictly well-defined pseudoquasimetric space on V:

Theorem 1. *Let $d_E : E \to [0,\infty)$ be defined as $d_E((i,j)) = 1/w(i,j) - 1$, then the minimum path $d(s,t)$ for any $s,t \in V$ forms a pseudoquasimetric on V. That is, for any $s,t,v \in V$, $d(s,s) = 0$ and $d(s,t) \leq d(s,v) + d(v,k)$*

Also, by trivial eliminations of Redirection pages (i.e. combining nodes) indiscernibility axiom also holds ($d(s,t) = 0 \implies t = s$), which results in a quasimetric, have an even stronger property for the reduced topological space. Such grants advanced algorithms such as Locality Sensitive Hashing [2] available on any of such defined graphs, and thus we can do some improvements over the original definition in Section 2.3.
With these techniques, we can simply construct a knowledge graph connecting Wikipedia pages, which resolving the problems by acronym users that drew the attention of Wiki editors; however, in the application, such relationships are far from enough, so we introduce the "interpolatory vertices" that constructed via a combination of keywords.

2.2 Interpolatory Vertices by Infobox and Semantics

By the method above (see Fig. 1(a)), we can easily find that there exists some connection between "University of Washington" and "Ana Mari Cauce", however, as there are so many links in the page, the weight of such keyword will likely be underestimated. In a non-stub, informatively worthy Wikipedia page, there is usually an infobox that provides a quick mapping with different entry titles. Also, in the context some of the sentences are highly indicative of a relationship; for these cases, we create a "interpolatory vertex" between the two titles, which can be regarded as a redirection vertex that has a unique outgoing edge. For example, 3.11 in Fig. 1(b) can be regarded as the vertex created as an identifier as the combination of page "University of Washington" and text key "president", then directly connected to "Ana Mari Cauce" with probability 1; such strongly indicative semantics could also form an edge between the original page to the interpolatory vertex, which increases the weight and decreases the relative distance between the original page to the important attributes of it.

2.3 Further Adjustments

2.3.1 LSH for User Defined Parameters

In the commercial practices for any system, it is necessary to reinforce the model by inputs from users. One example here is an adjustment for the disambiguation pages.
In Definition 3, when we initialize our data within our server without knowing the states of each individual user or the frequency of each page being used, we should have a uniform weight for each item; however, user states can introduce more parameters for our model. Consider the term "UW", it can refer to University of Washington, Wisconsin, Waterloo, Warsaw, or Wyoming, which are all officially using "UW" as their acronyms, yet they are completely different institutions, and for a resident in Washington state their desired result for querying "UW" must be "University of Washington". To prevent ambiguity and also consider efficiency, we cannot just change our back-end model each time when a user from a different region logged into our system, so there must be a data preprocessing. Here we introduce the method of Locality Sensitive Hashing (LSH) to cluster keywords and thus reduce the range of disambiguation for individual users. [1][11]

Definition 5 (Locality Sensitive Hash Functions)**.** *Suppose we have a family a functions $\mathcal{H} = \{h : P \to \mathbb{Z}\}$ of maps from our points P to the set of integers $\mathbb{Z}$; we say $\mathcal{H}$ is (c, cr, p_1, p_2)-LSH if: for all $p, q \in P$:*

$$d(p,q) \leq r \implies \mathbb{P}[h(p) = h(q)] \geq p_1$$

$$d(p,q) \geq cr \implies \mathbb{P}[h(p) = h(q)] \leq p_2$$

Algorithm 1 (LSH Algorithm)**.** *Let $\mathcal{H}$ be a family of such LSH, then we have the algorithm below to find the clusters of entries*

```
Preprocessing:
 Choose k * l functions uniformly random from H, denote as h_{1,1}, ..., h_{l,k};
 Construct l hash tables;
 for all 1<=i<=l store f_i(v)=(h_{i,1}(v), ..., h_{i, k}(v)) for all v in V in the i-th table;
 for all i, sort {f_i(v)}.
Given KeyNode v_0:
 for i = 1 to l:
  Compute f_i(v_0);
  Find all vertices v where f_i(v)=f_i(v_0) by binary search on table i;
  then if d(v_0,v)<=cr, output p.
 end for.
```

Theorem 2. *Let $p_1 = p_2^\rho$, let $l \in \Theta(|V|^\rho \ln |V|)$, $k = \log_{p_2}(1/|V|)$, then Algorithm 1 boosts the probability of outputting v as $d(v_0, v) \leq r$ to $1 - 1/|V|$ within space of*

$$O(l \cdot |V| \cdot k) = O(|V|^{1+\rho} \frac{\log |V|}{\log \frac{1}{p_2}})$$

and time of

$$O(n^\rho (d + \frac{\log |V|}{\log \frac{1}{p_2}}) + |O| d)$$

let $|O|$ be the output size $|d|$ be the time of calculating distance (for Dijkstra, $d \in \Theta(|E| + |V| \log |V|)$).

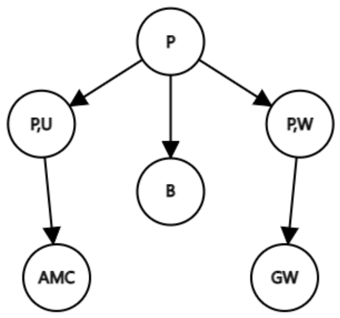


Fig. 2: P is “President”, W is “Washington”, U is “UW”, and B is “Biden”; even without any externally defined weights, we can still go with either left or right branch only if the second keyword is defined.

By this algorithm and feeding some “pivot nodes” (such as “Seattle”, “Madison” and “Warsaw”) to the algorithm, it can cluster the graph effectively, thus we can cluster the items only having the same hash value in a disambiguation page and present them to our users at a certain location.

2.3.2 Distance away from the similarly named titles

In another direction, sometimes we have a bunch of queries and keywords that are similar in spelling but completely different in references, such as “President UW”, “President Washington” and “President” There are a bunch of solutions to this problem: The first one is creating additional alias vertices (redirections), which is what Wikipedia currently uses for “President Washington” (and usually just redirect the isolated word “president” to POTUS, which is Biden); another way is by creating a special class of vertices that stores “attributes” (Job or personal titles, classes, categories, etc) and get the distance of the next keyword of such vertices and the choices of entities, as illustrated in Fig. 2.

3 Ethics and Next Steps

3.1 Affordance Analysis

However, in real life, Wikipedia is not the only authority on the Internet. Usually, we determine the sources' authority and set some kind of weight vector before training a model. Now consider otherwise:
Suppose we try to aggregate ideas across sources using the following rule: if there are multiple sources for an idea, choose the idea that appears the most often amongst the sources because it represents consensus. This obviously reduces our costs of preprocessing data, yet it could disastrously harm the reliability of our model. Firstly, there is plenty of fake news, conspiracy theory, and hate speech on the Internet, and the self-cleaning capabilities are never enough to put all of them away. Many individuals are living in their information cocoons [9], propagating and spreading ideas that contradict the facts. Also, contents are not necessarily created by human beings. Some “bots” do nothing but copy some low-effort contents everywhere, sometimes as lipsum to cover the sites only created for advertisements. Other cases are intentionally spreading their beliefs over other voices. Both possibilities could make the trained model with this kind of unqualified data lose its reliability for any usage; there were AI chatbots known for extensively using racist slurs and cursing languages.
To solve such a problem, the amelioration of our algorithm, especially increasing the training weights of the authoritative, reliable sources (from highest to lower, peer-reviewed journals, government documents, Wikipedia, mainstream media, well-moderated social networks), is vital. Either way, escaping from building the experimental features cannot accumulate experience for later works, and reducing social risks is usually just a technical issue that responsible developers should care about.

3.2 Next Steps

The algorithm defined in this paper requires an evaluation of shortest paths over an extensive graph, and introducing a parallel algorithm [5] will significantly improve performance. Also, the algorithm has a vast scale of data to process, so to reduce the parameters used, some neural network compression [13] helps it to proceed on lower-spec systems. As work in this paper was done individually with limited time to submit, there is also a lack of experimental data, which should be supplemented later.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments". In: *Journal of Computer and System Sciences* 58.1 (1999), pp. 137–147. ISSN: 0022-0000. DOI: `https://doi.org/10.1006/jcss.1997.1545`. URL: `https://www.sciencedirect.com/science/article/pii/S0022000097915452`.

[2] Alexandr Andoni and Ilya P. Razenshteyn. "Optimal Data-Dependent Hashing for Approximate Near Neighbors". In: *CoRR* abs/1501.01062 (2015). arXiv: `1501.01062`. URL: `http://arxiv.org/abs/1501.01062`.

[3] K. M. Annervaz, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. "Learning beyond datasets: Knowledge Graph Augmented Neural Networks for Natural language Processing". In: *CoRR* abs/1802.05930 (2018). arXiv: `1802.05930`. URL: `http://arxiv.org/abs/1802.05930`.

[4] Issar Arab. "AUI Thesis Search Engine - WordNet and NLP document based indexing". In: 2018.

[5] Xiaojun Dong et al. "Efficient Stepping Algorithms and Implementations for Parallel Shortest Paths". In: *CoRR* abs/2105.06145 (2021). arXiv: `2105.06145`. URL: `https://arxiv.org/abs/2105.06145`.

[6] Jinhua Dou et al. "Knowledge graph based on domain ontology and natural language processing technology for Chinese intangible cultural heritage". In: *Journal of Visual Languages & Computing* 48 (2018), pp. 19–28. ISSN: 1045-926X. DOI: `https://doi.org/10.1016/j.jvlc.2018.06.005`. URL: `https://www.sciencedirect.com/science/article/pii/S1045926X18300041`.

[7] L. R. Ford and D. R. Fulkerson. "Maximal Flow Through a Network". In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404. DOI: `10.4153/CJM-1956-045-5`.

[8] Evgeniy Gabrilovich and Shaul Markovitch. "Wikipedia-based Semantic Interpretation for Natural Language Processing". In: *CoRR* abs/1401.5697 (2014). arXiv: `1401.5697`. URL: `http://arxiv.org/abs/1401.5697`.

[9] Cédric Gossart. "Can Digital Technologies Threaten Democracy by Creating Information Cocoons?" In: Apr. 2014, pp. 145–154. ISBN: 1466660384. DOI: `10.4018/978-1-4666-6038-0.ch010`.

[10] Aidan Hogan et al. "Knowledge Graphs". In: *ACM Computing Surveys* 54.4 (July 2021), pp. 1–37. ISSN: 1557-7341. DOI: `10.1145/3447772`. URL: `http://dx.doi.org/10.1145/3447772`.

[11] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality". In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Dallas, Texas, USA: Association for Computing Machinery, 1998, pp. 604–613. ISBN: 0897919629. DOI: `10.1145/276698.276876`. URL: `https://doi.org/10.1145/276698.276876`.

[12] Ishani Mondal, Yufang Hou, and Charles Jochim. *End-to-End NLP Knowledge Graph Construction*. 2021. arXiv: `2106.01167 [cs.CL]`.

[13] Yuzhang Shang et al. "Lipschitz Continuity Guided Knowledge Distillation". In: *CoRR* abs/2108.12905 (2021). arXiv: `2108.12905`. URL: `https://arxiv.org/abs/2108.12905`.

[14] T. Yano and Moonyoung Kang. "Taking advantage of Wikipedia in Natural Language Processing". In: 2008.