Link Prediction for Personal Knowledge System

Georgiy Malaniya, Ivan Gerasimov, Aleksei Zelentsov

March 29, 2023

Project Description

Background: Obsidian is a popular personal knowledge management tool, but finding relevant links between notes can be challenging as the amount of information gRs.
Goal: Our project aims to develop a link prediction system for Obsidian using recommender systems approach to predict new links between notes and improve the user's PKM experience. We will compare our system with existing link prediction methods and baselines further.

Obsidian

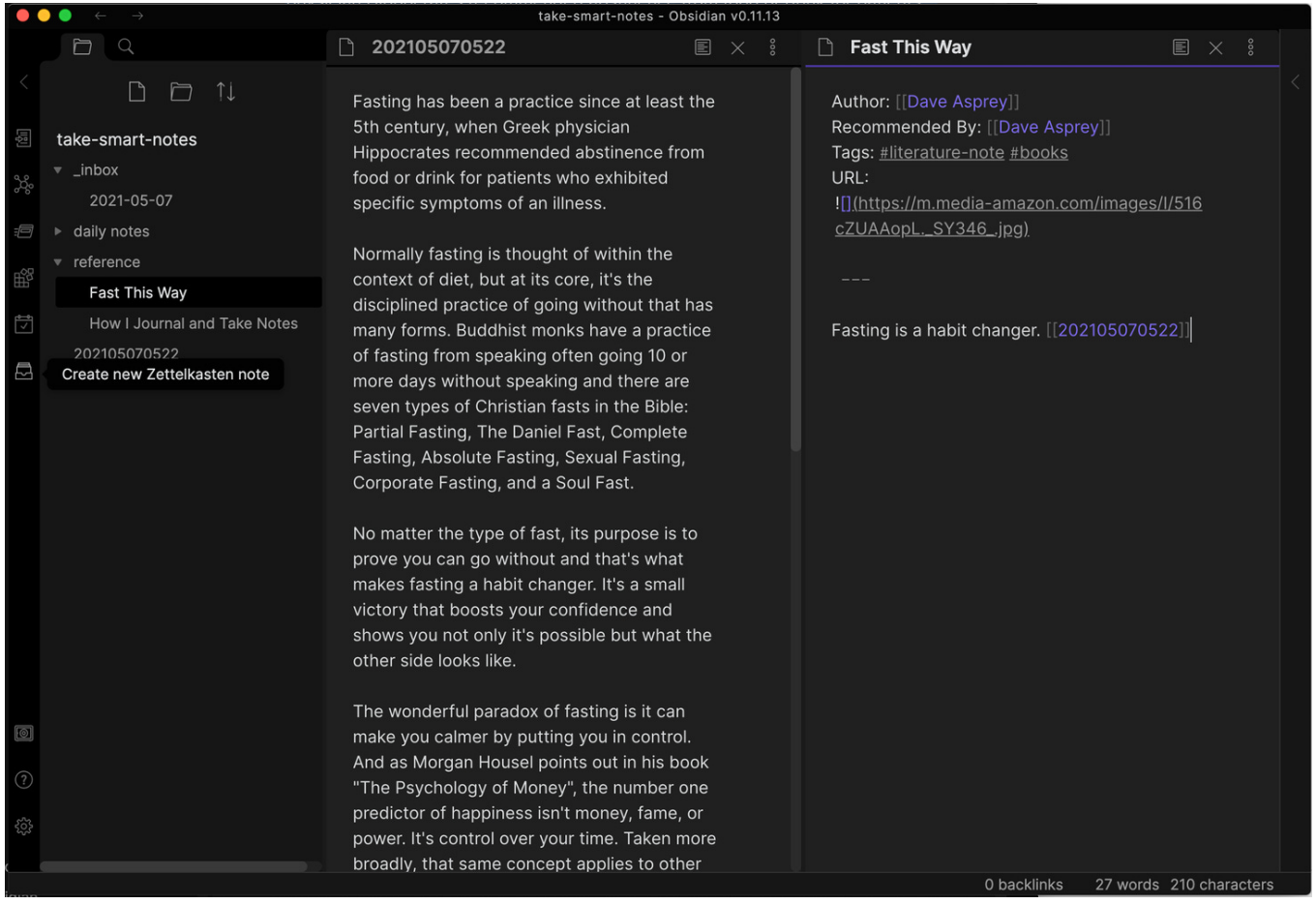


Figure: Obsidian interface

Obsidian

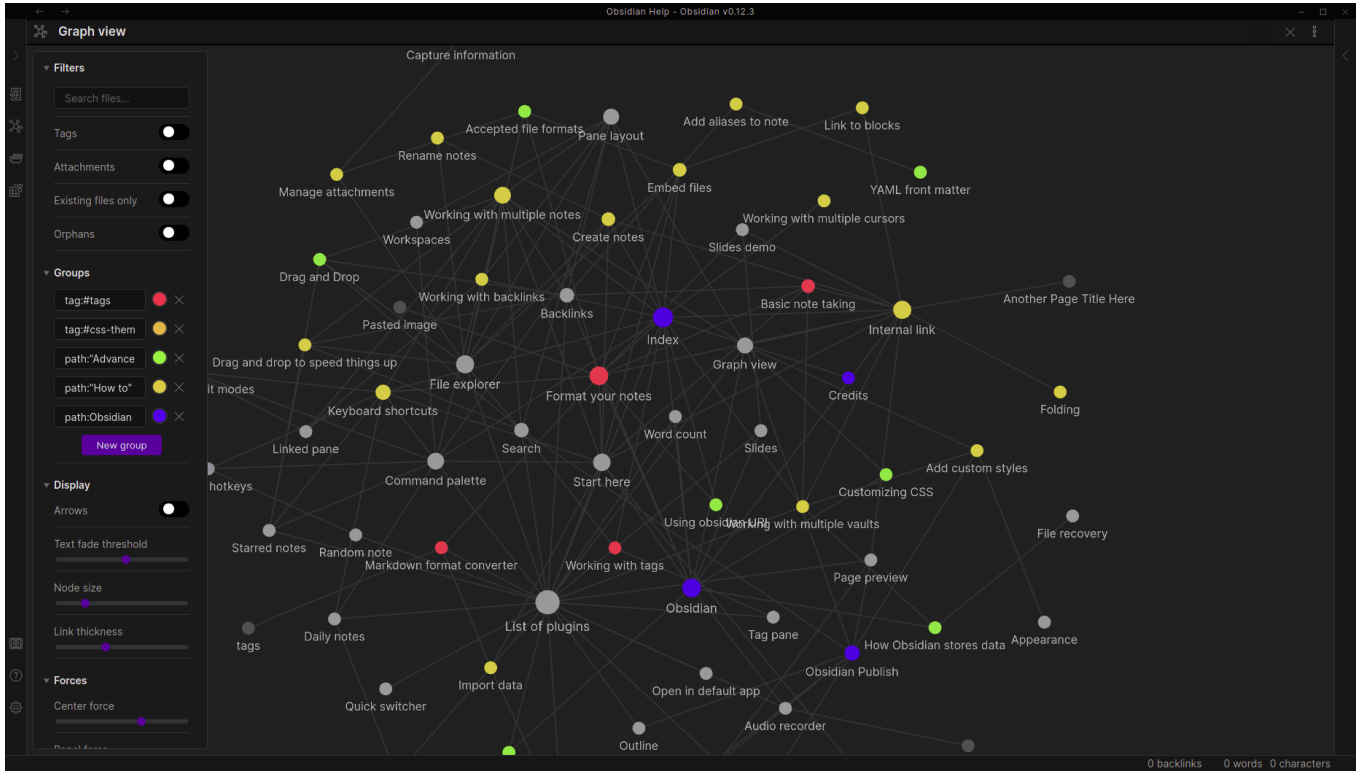


Figure: Obsidian backlinks graph

Main Hypothesis

We hypothesize that using both word embeddings and collaborative matrix factorization techniques can improve the accuracy and efficiency of link prediction for Obsidian.
By leveraging the **semantic information** in the content of notes as well as the user's behavior in creating and linking notes, we expect our system to outperform existing methods that rely on collaborative matrix only.

Model

- **BERT-LightFM**: Uses BERT for generating word embeddings, followed by an LightFM for link prediction.

Model

For the **BERT-LightFM** model, we use the LightFM library to train a recommender system that predicts the likelihood of a user interacting with an item. The optimization objective is to maximize the weighted approximate rank pairwise (WARP) loss, which is defined as:

$$\mathcal{L} = -\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} w_{u,i} \log \sigma(\hat{y}_{u,i}) + \alpha ||\Theta||^2 + \beta ||\Gamma||^2 \qquad (1)$$

where:

- $\mathcal{U}$ and $\mathcal{I}$ are the set of users and items respectively
- $w_{u,i}$ is the weight for the interaction between user u and item i
- $\sigma(x)$ is the logistic function
- $\hat{y}_{u,i}$ is the predicted interaction score between user u and item i using the embeddings of the user and item
- Θ and Γ are regularization matrices for the user and item embeddings, respectively
- α and β are hyperparameters that control the strength of the regularization

Optimization Objective

We evaluate the performance of our model using **recall** metrics. Specifically, we use **top-k recall** at 5, 10, and 20.

Scenario

- **Warm-start** scenario: the system has some initial information about the user's preferences.
- **Cold-start** scenario: there is no such information available.
- **Standard CF** scenario: assumes that the system has a reasonable amount of data available for training.

We chose to focus on the warm-start scenario for our evaluation because it is a more realistic scenario for a real-world personal knowledge system and also more suitable for comparison with existing graph-analysis baselines (such as Adamic Adar).

Data Preprocessing

1. Obtained data from a public Obsidian vault on GitHub.
2. Extracted relevant files from the vault using the Obsidian tools API.
3. Visualized the data as a graph using the NetworkX library.
4. Cleaned the data and removed looped backlinks and nodes without connections.

Train/Test Splitting and Holdout Construction

1. We selected a subset of nodes from the graph.
2. For each of the selected nodes, we deleted their connections from the graph.
3. The remaining connections were used to create the training set of links.
4. The deleted connections were stored in a separate evaluation set of links, which were used to evaluate the performance of the link prediction algorithms.

Metrics

We used two recall metric to evaluate our models We used the following formula to calculate recall on one node:

$$\text{recall} = \frac{\text{number of correctly predicted backlinks}}{\text{total number of deleted backlinks}}$$

And then averaging this metric for every node in test set

Adamic-Adar Algorithm

Description: Adamic-Adar measures the similarity of two nodes based on their common neighbors, with a weighting factor based on the inverse logarithm of the degree of each neighbor.
Formula: Given two nodes u and v and their set of common neighbors $N(u) \cap N(v)$, the Adamic-Adar score is calculated as follows:

$$AA(u, v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log(|N(w)|)}$$

where $|N(w)|$ is the degree of node w.

Adamic-Adar as a Baseline

- Adamic-Adar is a widely used baseline for link prediction tasks.
- It is a simple, yet effective method that captures the notion that nodes with fewer common neighbors are more important than those with many.
- As a baseline, it provides a lower bound for the performance of more sophisticated methods.

Results

We tested our model on two datasets: "Brain" and "Obsidian hub"

Results

Results on "Brain" dataset

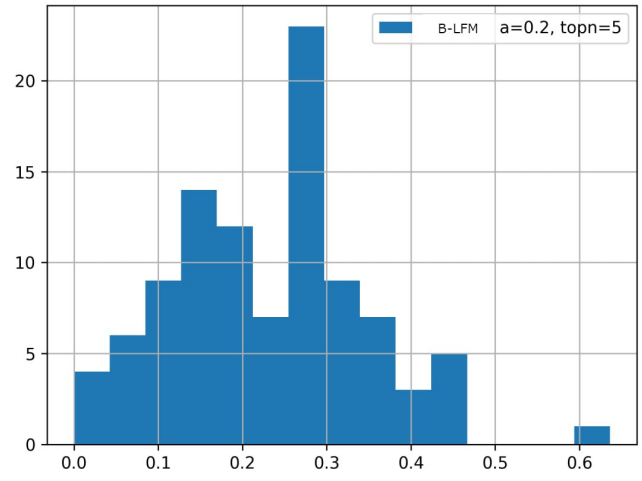
	$\alpha = 0.1$			$\alpha = 0.2$		
	top-n=5	top-n=10	top-n=20	top-n=5	top-n=10	top-n=20
AA	0.163	0.269	0.407	0.163	0.273	0.411
BERT-LFM	0.226	0.347	0.421	0.205	0.309	0.410

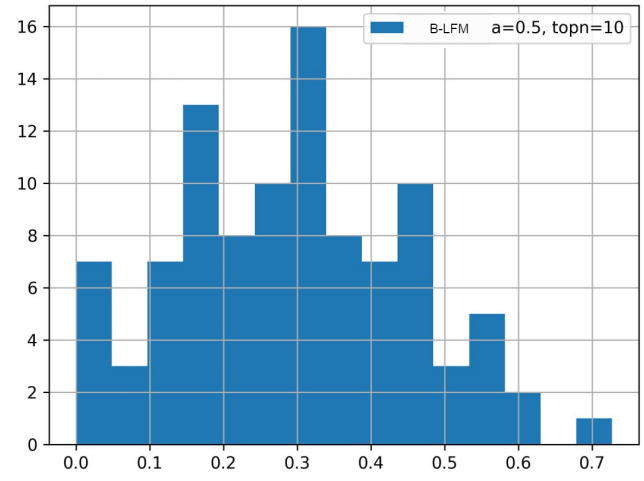
Results

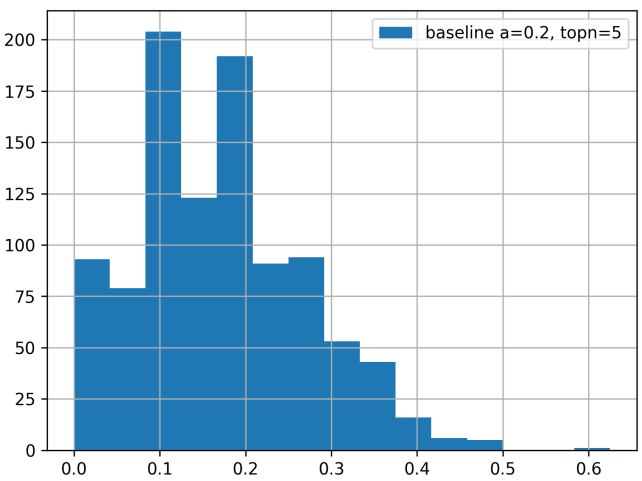
Results on "Obsidian hub" dataset

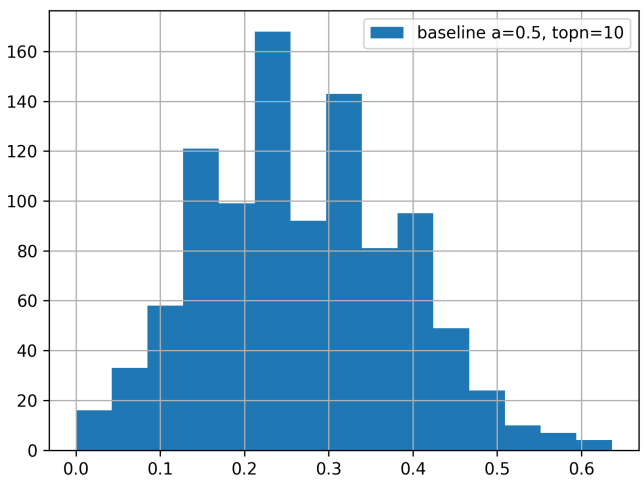
	$\alpha = 0.1$			$\alpha = 0.2$		
	top-n=5	top-n=10	top-n=20	top-n=5	top-n=10	top-n=20
AA	0.007	0.016	0.029	0.019	0.018	0.025
BERT-LFM	0.081	0.107	0.129	0.106	0.157	0.198

Histograms


B-LFM a=0.2, topn=5
0
5
10
15
20
0.0
0.1
0.2
0.3
0.4
0.5
0.6


B-LFM a=0.5, topn=10
0
2
4
6
8
10
12
14
16
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


baseline a=0.2, topn=5
0
25
50
75
100
125
150
175
200
0.0
0.1
0.2
0.3
0.4
0.5
0.6


baseline a=0.5, topn=10
0
20
40
60
80
100
120
140
160
0.0
0.1
0.2
0.3
0.4
0.5
0.6

GitHub

RecSysTeam/LinkPredict