Pub2vec: A Recommender System for Similar Publications via Citation Network Embeddings

Brian K. Ryu*
Department of Chemical Engineering, Stanford University
bryu@stanford.edu

Abstract

Machine learning methods for recommender systems have proved their efficacy and have become popular for a wide range of applications ranging from product recommendations to advertisements and social networks. However, a potentially beneficial application of recommender systems that has yet to be realized is one for scientific publications. In this project, I present the *pub2vec* algorithm trained on a citation network to identify and recommend similar publications. A citation network is a social network that contains citation relations of academic publications. Although the network is simply a directed graph where nodes represent publications and edges represent citation relations, the network implicitly embodies information on the contents via links between citing and cited papers. The algorithm generates low-dimensional representations of a citation network, which are input to nearest neighbor algorithms to identify similar or closely related works. My results show that the publications recommended by *pub2vec* are highly relevant to queried publications. This work will potentially be able to aid researchers in conducting literature searches.

1 Introduction

Recommender systems are type of filtering systems that predicts relevance or preference of items to a specific user and provide adequate recommendations.[1, 2] Commonly used for E-commerce, recommender systems have been successful in aiding customers to find products or displaying personalized online advertisements. In social networks, recommender systems have been utilized to present personalized news feeds or connection recommendations.[3] Overall, recommender systems have proven their efficacy in providing recommendations to users in a wide variety of areas. An application of recommender systems, however, that has yet to be realized is one for scientific publications; an algorithm that can effeciently provide recommendations for "similar publications" can be useful for researchers in conducting literature surveys.

A potential avenue for constructing a recommender system for scientific literature may be via citation networks. A citation network is a social network that contains citation relations of academic publications. The network is represented as a directed acyclic graph (DAG) in which each node depicts a publication while directed edges represent citation relations. The analysis of citation networks has been an important topic in scientometrics, which studies the body of scientific literature. However, an application of citation network analysis for a recommender system for similar publication has not been developed to date. Conventional search engines or bibliographies such as Google Scholar or DBLP allows users to search academic publications based on keywords. Yet, such venues do not provide a means to identify works related to a specific publication. An algorithm to recommend related or similar publications can be beneficial for researchers during literature searches. In particular, a machine learning-based recommender system may be able to tackle the challenge of identifying similar publications.

The main challenge in incorporating machine learning approaches to graphs is the high dimensionality of graphs. A graph, $G = (V, E)$, consists of a set of nodes, V, and edges E. Edge relations are commonly represented as an adjacency matrix, whichrequires a large space complexity of $O(|V|^2)$. As a result, performing machine learning tasks for large networks such as citation networks or social (e.g. Facebook, or Instagram) networks comprising over millions of nodes may become prohibitively expensive. Furthermore, the absence of spatial locality or fixed node ordering renders networks unruly data representations for machine learning. While other representations of graphs such as sparse matrix representations or adjacency lists exist, the inherent high-dimensionality of graphs remains a conundrum.

*GitHub Code Repository: https://github.com/bkryu/CS229_Project

In this project, I present the *pub2vec* algorithm to (1) generate low-dimensional representations of a citation network and (2) identify similar or closely related works. The algorithm extends and generalizes *node2vec* by aggregating features learned from multiple *node2vec* runs with different sets of parameters. With features gathered from random walks that explore breadth and depth, I generate purely citation relation-based paper recommendations. Qualitative and quantitative examination of recommended publications show that *pub2vec* is able to rapidly suggest highly relevant publications to the user upon queries.

2 Related work

Several approaches in providing recommender systems exist in literature. The underlying theme of building recommender systems is identifying what is relevant to the user. For E-commerce, recommender systems commonly utilize pre-tagged characteristics in order to quickly recommend additional items.[1] Another approach is collaborative filtering where recommendations are made based on the assumption that users who have liked similar products in the past will like similar products in the future as well.[4] Once a "neighborhood" of the current user is established, nearest neighbor algorithms are then used to narrow down to recommended items. Context-sensitive recommendations such as those used by Uber and Lyft compiles data from a variety of sources: GPS, current time, operational status, and etc. to create routes and pickups for drivers.[5] Despite the myriad types of recommender systems, no machine learning-based recommender system for literature searches exist to the best of the author's knowledge.

Though not for recommender systems, many studies have tackled high-dimensionality via embedding methods that reduce graph representations to lower dimensions, which is essential for fast creation of recommendations. A successful classical method is spectral clustering, where eigenvalues and eigenvectors of the Laplacian matrix of the graph are used to represent the graph.[6] An example of a machine learning approach is the Large-scale Information Network Embedding (LINE) algorithm.[7] LINE embeds a network into a matrix by sampling immediate and secondary neighboring edges from source nodes. Another approach is *DeepWalk*, which simulates random walks between nodes to learn the neighborhood of each node.[8] *Node2vec* generalizes the approach of *DeepWalk* by allowing tunable random walk parameters to selectively focus on community detection or structural equivalence.[9] Such approaches have led to effective algorithms for node classification or link prediction.

3 Dataset and Features

The full DBLP citation network database, acquired on May 5th, 2019 by Arnetminer and available at `https://aminer.org/citation` was used to analyze the network of computer science literature.[10] The DBLP bibliography includes all major journal articles as well as conference proceedings, which are both highly important in computer science literature. The citation network is a directed graph containing 4,107,340 scientific publications (nodes) and 36,624,464 citation relationships (edges). In addition to basic citation relations, the dataset contains "field of study" information that was collected by the mining software and describes the relevance of each publication to a scientific field. Each node in the citation network contains a list of fields, each coupled to a weight, that specifies how the paper is related various fields (e.g. [("Web mining, 0.65"), ("Deep learning", 0.21") ...]). This field of study information is later stored as a vector and used to construct a metric to evaluate the performance of the recommender system.

4 Methods

The overall procedure for identifying closely related publications via *pub2vec* is summarized in Figure 1. First, the *pub2vec* algorithm is run on a directed, acyclic citation network (Figure 1(a)). Through repeated sets of random walks respectively biased for depth-first search (DFS) and breadth-first search (BFS), the algorithm learns neighborhoods that include DFS-neighbors and BFS-neighbors. Then, neighborhood features from random walks are aggregated to create a 2D embedding of the learned features where each column is a d-dimensional vector representing d features of a single node (Figure 1(b)). Such an embedding holistically summarizes both the community as well as the structural role of each node. Finally, nearest neighbors in the extracted features space are identified and recommended to the user (Figure 1(c)).

4.1 The *pub2vec* Algorithm

The pseudocode for *pub2vec* is summarized in Algorithm 1. The goal of *pub2vec* is to obtain the mapping function from node to feature vectors, $f : V \to \mathbb{R}^d$. *LearnPubFeatures* repeatedly calls *LearnFeatures* with different sets of random walk parameters and aggregates features learned each time. The functions *LearnFeatures* and *node2vecWalk* are directly taken from the original *node2vec* work and here I only discuss key ideas.[9] *LearnFeatures* first initializes a table of transition probabilities for 2nd order Markov chains. This preprocessing speeds up the random walk simulation to $O(1)$ time complexity per node. Then, *node2vecWalk* is repeatedly called to generate random walks to sample neighborhoods. For every source node $u \in V$, $N_S(u) \subset V$ is the neighborhood function of node u using some sampling strategy S. The sampling strategy S includes the set of parameters used to perform random walks. The objective function we use to

Algorithm 1: The *pub2vec* algorithm.

LearnPubFeatures(Graph $G = (V, E, W)$, Dimensions δ, Walks per node r, Walk length l, Context size k, Return-list P, In-out-list Q)
Initialize f_{p2v} as empty matrix.
for $i = 1$**to** $length(P)$ **do**
  f =LearnFeatures$(G, \delta, r, l, P[i], Q[i])$
  Append f to f_{p2v}
return f_{p2v}

LearnFeatures(Graph $G = (V, E, W)$, Dimensions δ, Walks per node r, Walk length l, Context size k, Return p, In-out q)
π = PreprocessModifiedWeights(G, p, q)
$G' = (V, E, \pi)$
Initialize *walks* to Empty
for $iter = 1$**to** r **do**
  for ***all*** *nodes* $u \in V$ **do**
    $walk$ = node2vecWalk(G', u, l)
    Append $walk$ to $walks$
  f = StochasticGradientDescent$(k, \delta, walks)$
return f

node2vecWalk(Graph $G' = (V, E, \pi)$, Start node u, Length l)
Initialize $walk$ to $[u]$
for $walk\_iter = 1$**to** l **do**
  $curr = walk[-1]$
  V_{curr} =GetNeighbors$(curr, G')$
  s =AliasSample(V_{curr}, π)
  Append s to $walk$
return $walk$

obtain representations is:

$$\sum_{u \in V} \log Pr(N_S(u)|f(u)) \quad \text{where} \quad Pr(n_i(u)|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))} \tag{1}$$

for $n_i \in N_S$, which is the log-probability of observing a network neighborhood given the learned feature representation. The optimal feature representation f that maximizes this objective function via stochastic gradient descent (SGD). This framework was developed in *node2vec* for community detection, link prediction, and node classification tasks.[9] In essence, *pub2vec* repeatedly runs *node2vec* with varying parameters to obtain δ features each time. Then, the features are aggregated to obtain a comprehensive feature vector of size d for each node.

For this project, I vary the parameter q in Algorithm 1, which controls the ratio of preference in breadth to depth in the random walk. A high q favors BFS-like random walks, while a low q favors DFS-like random walks. I have selected

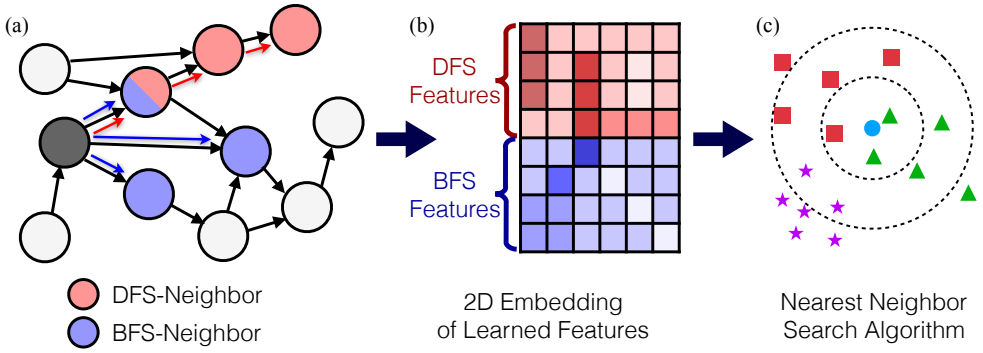

Figure 1: Summary of recommending similar publications. (a) A directed, acyclic citation network is embedded in a (b) lower-dimensional 2D embedding. The embedding is then (c) analyzed using nearest neighbor algorithms.

$q = 0.5, 1.0$, and 2.0 using $\delta = 32$ feautures learnt per iteration. The range of q's allows the algorithm to balance the importance of community and structural role in the feature vector. The resultant feature vector for each publication has 96 features.

4.2 Scalability: Time and Space Complexity Analysis

The procedure in Figure 1 contains two computational steps: (1) Obtaining a 2D feature embedding via *pub2vec*, and (2) generating recommendations via nearest neighbor search in embedded feature space. The computationally heavier first step involves preprocessinging of transition probabilities and simulating random walks for each set of random walk parameters (*node2vec* calls). Constructing the transition matrix requires $O(n|V|)$ computations where n is the average degree of nodes, and simulating random walks requires $O(|V|)$ compute time. Hence, for m *node2vec* calls, a total of $O(mn|V|)$ time is required to obtain the comprehensive embedding matrix. Storage of the transition matrix in memory requires $O(|E|^2)$ floating points numbers. With the current data set with 32.6 million edges, the memory demand was over 200 GB of memory for the probability matrix. Therefore, the representation learning for this project was done on the memory-optimized n1-standard-64 compute node on Google Cloud Platform with 240 GB RAM. Once all random walks are completed, the generated embedding matrix can be saved and only requires $O(d|V|)$ storage. With the current dataset of approximately 4.1 million nodes and 96 features, the embedding matrix requires less than 4 GB of storage.

After the embedding is created, the second step performs a nearest neighbor search to identify nodes that are closest in the feature space. This step requires $O(d|V|)$ compute time and space, similar to the previous step. In practice, however, this operation executes significantly faster than the random walks. The ability to generate recommendations fast from the embedding matrix is the primary advantage of the current implementation.

5 Experiments and Results

The performance of recommender systems can be best evaluated via A/B testing or user feedback, which extends beyond the scope of this project. Instead, I provide qualitative evaluations based on visual examinations of query-recommendation pairs, and quantitative evaluations using the "field of study" feature included in the dataset.

5.1 Qualitative Evaluations

Using the embedding as described in Section 4.1, a feature matrix, $f \in \mathbb{R}^{96\times|V|}$ was constructed. To present qualitative results, I query a series of publications with from a range of computer science sub-fields and citation counts. The query and recommended publication titles are summarized in Table 1. For each recommended publication, relevance to the queried publication was manually assessed an shown at the right-most column. Qualitative inspection shows that the relevance of returned recommendations depends on the number of citations of the input query. When input publications with few citations were queried (≤ 100), returned publications were mostly irrelevant. When input publications were highly cited (≥ 800), the learned features were able to successfully identify similar works. The relatively poor quality of

Input Publication	**# Cited**	**Top-3 Similar Publications Recommended by *pub2vec***	**Relevant?**
Training Classifiers with Natural Language Explanations[13]	22	Feasibility study of stochastic streaming with 4K UHD video traces[11]	×
		Modeling single event crosstalk speedup in nanometer tech.[12]	×
		Cross lifecycle var. anal.: Utilizing requirements and testing artifacts[14]	×
Learning from untrusted data[16]	97	Fuzzy Planar Graphs[15]	◯
		A 65nm std. cell set and flow dedicated to auto. async. circuits design[17]	×
		A new table of permutation codes[18]	×
Parsing with Compositional Vector Grammars[21]	808	Better Word Representations with RNN for Morphology[19]	◯
		Semi-Supervised Recursive Autoencoders for Pred. Sentiment Dist.[20]	◯
		Dyn. Pool. and Unfolding Recursive Autoencoders for Paraph. Detect.[22]	◯
node2vec: Scalable feature learning for networks [9]	2,284	LINE: Large-scale Information Network Embedding[7]	◯
		A High-Perf. Semi-Supervised Learning Method for Text Chunking[23]	◯
		Dep. Tree-based Sentiment Class. using CRFs with Hidden Variables[24]	◯
k-means++: the advantages of careful seeding[27]	4,648	Clustering of the self-organizing map[25]	◯
		Integ. constraints and metric learning in semi-supervised clustering[26]	◯
		Data clustering: 50 years beyond K-means[28]	◯
A formal basis for the heuristic determination of min. cost paths (A*)[31]	9,026	Collision detect. and avoidance in computer controlled manipulators[29]	◯
		A mobius automation: an app. of artificial intelligence techniques[30]	◯
		Heuristics: intelligent search strategies for computer problem solving[32]	◯
Going deeper with convolutions [35](GoogLeNet)	17,646	Very Deep Conv. Net. for Large-Scale Image Recognition[33]	◯
		Caffe: Convolutional Architecture for Fast Feature Embedding[34]	◯
		ImageNet Classification with Deep Conv. Neural Nets.[36]	◯

Table 1: Publications queried, the number of times queried publication is cited (middle), the top-3 recommended publications returned by the *pub2vec* algorithm, and their relevance to queried title.

recommendations for scarcely cited publications is likely due to the lack of "information" arising from citation relations. With only few other works citing a publication, there is insufficient training data for the *pub2vec* algorithm to learn appropriate features of size $d = 96$ for the node. Furthermore, there exist a significantly larger number of publications with few citations than many citations, and these nodes structurally lie near the periphery of a graph. Thus, there exists a significant number of nodes with similar structural features to a peripheral node, and a significant portion of embedded parameters pertain to the structural role of a node. A potential means to ameliorate the algorithm's weakness against scarcely cited publications is to assign more features on communities at embedding time. This change can be done by Adjusting the random walk parameters to include high-q value random walks that favor BFS searches. Nevertheless, the algorithm performs remarkably well in identifying closely related publications based on features learned from random walks given that a citation network was the only information given to *pub2vec* to create recommendations.

5.2 Quantitative Evaluations

To further obtain a qualitative evaluation of recommended publications, I compare the distances between the queried and recommended publications using (1) the embedded feature space, and (2) the aforementioned "field of study" features. The distance between two nodes $u, v \in V$ in embedded feature space is computed as $d_f = ||f_u - f_v||_2$ where f_u is the embedded feature vector of u. Similarly, the distance between the same two nodes in the field of study features is computed as $d_g = ||g_u - g_v||_2$ where g_u is the field of study vector of node u, as discussed in Section 3. Here I compute the distribution of n d_f and d_g for top-8 similar publications, as well as 30 randomly selected publications for each of the input publications shown in Table 1. The results are shown in Figure 2:

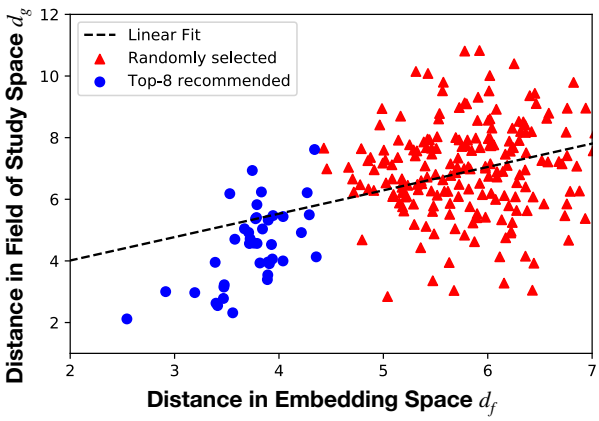

Figure 2: Publication-to-publication distance in field of study space d_g as a function of distance in embedding space d_f. Blue circles indicate publications that were recommended by *pub2vec*. Red triangles represent publications that were randomly selected. The linear fit resulted in a slope 0.374 and intercept 4.853, with an r^2 value 0.146.

Ideally, one would expect a strong positive correlation between d_f and d_g, which means that the distance between two nodes in embedding space is positively correlated to the distance in field of study. The agreement shown in Figure 2 is noisy but shows a reasonable positive correlation, especially for the publications recommended by *pub2vec*. The large amount of noise can be attributed to two sources of inaccuracies. First, the field of study information gathered by mining software may not have accurately reflected the contents in the actual publication. Therefore, to continue to use field of study vectors as evaluation metrics in future work, assessing the accuracy of field of study information is necessary. Second, the recommendations provided by *pub2vec* may simply be irrelevant. Based on the qualitative examination of results from Section 5.1, it is unlikely that recommendations are actually unrelated to queried publications. Ultimately, proper evaluation through user feedback and testing is necessary for practical use of the current recommender system.

6 Conclusions and Future Work

In this project, I have presented the *pub2vec* algorithm trained on the DBLP computer science citation network for recommending similar publications. The algorithm extends the approach of *node2vec* to holistically learn an embedding matrix that describe the community and structural features of nodes.[9] A significant advantage of the current approach is the separation of training (embedding) and query steps; once the embedding is generated, producing recommendations for a particular nodes can be done fast. My results show that for fairly well cited publications (≥ 800), the algorithm is able to generate pertinent publications to users. A quantitative evaluation using field of study labels for each node show that the embedded features adequately describe each publication. However, this metric merely serves as a provisional metric and performing proper testing of the recommender system is the remaining challenge. Hence, future work should acquire feedback from users by receiving feedback or analyzing user usage. Then, other random walk parameters such as the walk length or feature vector size should be optimize to produce ideal recommendations.

7 Contributions

All work for this project was done by Brian K. Ryu. This project was done solely for CS 229 and no other class (I have neither taken nor am enrolled in CS 224W).

References

[1] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166, ACM, 1999.

[2] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–59, 1997.

[3] J. He and W. W. Chu, "A social network-based recommender system (snrs)," in *Data mining for social network data*, pp. 47–74, Springer, 2010.

[4] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[5] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An energy-efficient mobile recommender system," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 899–908, ACM, 2010.

[6] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, International World Wide Web Conferences Steering Committee, 2015.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, ACM, 2014.

[9] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, ACM, 2016.

[10] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, ACM, 2008.

[11] J. Kim and E.-S. Ryu, "Feasibility study of stochastic streaming with 4k uhd video traces," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1350–1355, IEEE, 2015.

[12] S. Sayil and L. Yuan, "Modeling single event crosstalk speedup in nanometer technologies," *Microelectronics Journal*, vol. 46, no. 5, pp. 343–350, 2015.

[13] B. Hancock, M. Bringmann, P. Varma, P. Liang, S. Wang, and C. Ré, "Training classifiers with natural language explanations," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2018, p. 1884, NIH Public Access, 2018.

[14] M. Steinberger, I. Reinhartz-Berger, and A. Tomer, "Cross lifecycle variability analysis: Utilizing requirements and testing artifacts," *Journal of Systems and Software*, vol. 143, pp. 208–230, 2018.

[15] S. Samanta and M. Pal, "Fuzzy planar graphs," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 1936–1942, 2015.

[16] M. Charikar, J. Steinhardt, and G. Valiant, "Learning from untrusted data," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 47–60, ACM, 2017.

[17] M. Moreira, B. Oliveira, J. Pontes, and N. Calazans, "A 65nm standard cell set and flow dedicated to automated asynchronous circuits design," in *2011 IEEE International SOC Conference*, pp. 99–104, IEEE, 2011.

[18] D. H. Smith and R. Montemanni, "A new table of permutation codes," *Designs, Codes and Cryptography*, vol. 63, no. 2, pp. 241–253, 2012.

[19] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 104–113, 2013.

[20] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 151–161, Association for Computational Linguistics, 2011.

[21] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465, 2013.

[22] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in neural information processing systems*, pp. 801–809, 2011.

[23] R. K. Ando and T. Zhang, "A high-performance semi-supervised learning method for text chunking," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 1–9, Association for Computational Linguistics, 2005.

[24] T. Nakagawa, K. Inui, and S. Kurohashi, "Dependency tree-based sentiment classification using crfs with hidden variables," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 786–794, Association for Computational Linguistics, 2010.

[25] J. Vesanto, E. Alhoniemi, *et al.*, "Clustering of the self-organizing map," *IEEE Transactions on neural networks*, vol. 11, no. 3, pp. 586–600, 2000.

[26] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the twenty-first international conference on Machine learning*, p. 11, ACM, 2004.

[27] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[28] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[29] S. M. Udupa, *Collision detection and avoidance in computer controlled manipulators*. PhD thesis, California Institute of Technology, 1977.

[30] N. Nilson, "A mobile automation: An application of artificial intelligence techniques," in *Proceedings of the Fith International Joint Conference on Artificial Intelligence*, p. 509, 1969.

[31] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[32] J. Pearl, "Heuristics: intelligent search strategies for computer problem solving," 1984.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[34] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.