WARSAW UNIVERSITY OF TECHNOLOGY

FACULTY OF MATHEMATICS AND INFORMATION SCIENCE

Ph.D. Thesis

Barbara Rychalska, M.Sc.

A Multimodal and Interpretable Recommender System for Massive Datasets

Supervisor

prof. dr hab. inż. Przemysław Biecek

WARSAW 2023

Abstract

The area of machine learning for recommender systems has faced unprecedented growth in recent years. Research is fuelled by practical business demand for these models. The pace of progress is measured by dynamically growing benchmarking effort, often in terms of competitions held at esteemed scientific conferences, such as the annual KDD Cup at SIGKDD Conference on Knowledge Discovery and Data Mining, RecSys Challenge held at ACM Conference Series on Recommender Systems (RecSys), or Stanford's OGB Large-Scale Challenge. Recommender systems research relies heavily on topics such as big data processing, scalable machine learning, large graph processing, information retrieval, and multimodal aspects of data. The practical application of recommender systems in production also calls for development of interpretability and robustness algorithms.

The purpose of this PhD thesis is to research the abilities and limitations of current recommender algorithms, and introduce new methods in the broad area of recommenders. These new methods should improve upon the performance results achieved by previous research and contribute to the growth of this science branch.

The thesis introduces a number of research objectives, which have all been achieved. The final result is a full recommender system, which reaches new state-of-the-art results on multiple datasets and performance metrics. The recommender system is applied in a business setting in two commercial platforms, which are currently available on the market and have reached a business success.

Streszczenie

Metody uczenia maszynowego dla systemów rekomendacyjnych rozwijają się obecnie niezwykle szybko. Rozwój metod naukowych podsycany jest przez rosnącą praktyczną potrzebę wykorzystywania systemów rekomendacyjnych w biznesie. Postęp w dziedzinie mierzony przez dynamicznie rozwijane benchmarki, często tworzone w ramach konkursów na prestiżowych międzynarodowych konferencjach, takich jak na przykład konkurs KDD Cup na konferencji SIGKDD Conference on Knowledge Discovery and Data Mining, RecSys Challenge organizowany przez ACM Conference Series on Recommender Systems (RecSys), czy OGB Large-Scale Challenge uniwersytetu Stanford. Badania nad systemami rekomendacyjnymi wiążą się ściśle z dziedzinami takimi przetwarzanie dużych danych (*big data*), skalowalne uczenie maszynowe, przetwarzanie danych grafowych, wyszukiwanie informacji (*information retrieval*), oraz przetwarzanie danych multimodalnych. Zastosowanie systemów rekomendacji w praktycznych scenariuszach biznesowych powoduje, że ważnym wyzwaniem jest tu również interpretowalność modeli oraz badanie ich stabilności wobec przykładów wrogich (*adversarial examples*).

Celem niniejszej pracy doktorskiej jest zbadanie zdolności i ograniczeń istniejących systemów rekomendacji, oraz stworzenie nowych metod rekomendacyjnych. Zaproponowane rozwiązania powinny poprawić jakość znanych metod, co może przyczynić się do rozwoju tej dziedziny naukowej.

Wszystkie postawione cele pracy zostały zrealizowane, czego efektem końcowym było stworzenie multimodalnego i odpornego systemu rekomendacyjnego, który osiąga sumarycznie lepsze wyniki na wielu benchmarkach. Badania opisane w pracy zostały wdrożone w ramach dwóch platform komercyjnych, które są obecnie dostępne na rynku i osiągają sukces biznesowy.

Contents

List of Figures

List of Tables

1 Introduction

This work focuses on machine learning recommender systems based on user interaction histories and additional multimodal data. A particular focus is placed on the aspects of performance, scalability, interpretability, and robustness of recommender models.

Recommender systems build the core of many digital organizations which aim to offer their products to a large pool of customers. The core task of recommendation consists in choosing relevant products for a customer, based on multiple criteria such as the customer's individual shopping history and similarity of products [Jannach et al., 2010]. The problem of recommendation arises in companies which sell any type of goods or services - be it e-commerce, banking, gaming, telecommunication or various other companies [Jin, 2017, Smith and Linden, 2017, Lu et al., 2012, Pawlicka et al., 2021]. While historically done semi-manually with hand-crafted rules, recommendation now is increasingly done with the help of machine learning models. It is estimated that the whole market of automated recommendation was worth 2.12 billion USD in 2020, and it is expected to reach 15 billion USD by 2026 [Mordor Intelligence Research, 2021].

The area of recommendation has drawn significant attention from researchers since the beginning of the Internet. Recommendation is closely linked to another early Internet invention: search engines. Both recommendation and search aim to find relevant content. The difference is that searching follows by user query, while recommendation uses various indirect clues but without conscious, explicit queries by the user. Both of these areas are linked to the growth of large datasources and networks, which needed to be navigated in some way to allow users find relevant information in reasonable time.

One of the first attempt at a personalized behavioral recommendation was the Xerox PARC Tapestry [Goldberg et al., 1992], a system which helped users find relevant documents across various mailing lists. The system introduced the idea that explicit annotations and rankings by users could help others find relevant content. Another early system, the GroupLens [Resnick et al., 1994] was focused on filtering massive amounts of news articles and introduced the basic idea of automatically finding similar users in the database for making predictions. These first approaches were conceptually simple, but more and more sophisticated approaches quickly followed, starting with methods such as the principal component analysis (PCA) on user-item interaction matrices, and leading to mostly neural network-based models designed today. Many approaches are based on learning node representations in graphs. Graphs are mathematical structures composed of nodes which are connected by edges. Nodes can represent various entities such as people or objects, which are connected by edges if an interaction happened between them. It turns out that graphs are the adequate structures in representing various types of interactions, such as purchases or friendships in social networks.

Fuelled by famous challenges and contests, such as the famous \$1 million Netflix Prize in 2006, and various competitions in the KDD Cup series, the recommendation domain was approached by both academia and commercially-funded groups [Jannach et al., 2010]. Large tech companies such as Amazon and Netflix attribute much of their commercial succes to high quality recommendation[1][2], so the area is expected to develop dynamically in the future.

In spite of significant efforts, and huge progress achieved in recommender systems to date, there are still many challenges left to be solved by the research community:

- **Large size of complex models leads to slow training and inference times.** Many of the proposed recommender algorithms are not fast enough due to complicated neural architectures and they usually cannot scale to massive datasets [Ferrari Dacrema et al., 2019, Bai et al., 2021a, Tripathy et al., 2020]. Datasets currently in operation reach hundreds of millions of users, and billions of recorded transactions. For example, Twitter (one of the most popular worldwide social media platforms) has over 330 million monthly users and approximately 500 million tweets every day [Aslam, 2021], on which recommendations need to be made in near-real time.

- **Problematic evaluation of models, unclear rate of advancement.** A substantial body of work suggests that current direction of ever-more-complex neural architectures might not be bringing adequate quality improvements, and might lead to ”phantom progress”, i.e. a progress which fails to materialize under careful testing procedures [Ferrari Dacrema et al., 2019, Ludewig et al., 2019].

- **Hardships in incorporation of multimodal data.** The most popular and established models rarely allow to incorporate multimodal data, that is auxiliary data sources in the form of text, images, or categorical assignments. Most models are designed to ingest just one modality. Models which combine speed and multimodal capabilities are especially scarce. However, multimodal data is crucial to recommendation, as each product is usually characterized by various modalities such as textual data (product name, description),

[1]`https://www.amazon.science/the-history-of-amazons-recommendation-algorithm`

[2]`https://fortune.com/2012/07/30/amazons-recommendation-secret/`

images (product photos), categorical data (price, amount in stock, etc.), and interaction-based data (who bought/clicked/viewed this product before).

- **More research on interpretability and robustness methods is needed.** The high impact of recommender quality on financial results of companies demands further work on the reliability of the systems. New methods on recommender interpretability need to be introduced in order to better explain their internal workings to decision makers, convincing them that the models can be trusted and are robust to errors in data or outlier inputs. Such goals can be achieved with the help of Explainable Artificial Intelligence (XAI) methods, which is a very dynamically growing science branch at this moment. Particularly pressing are the problems connected with the existence of adversarial examples [Goodfellow et al., 2015]. Adversarial examples are inputs to models that are perturbed either on purpose or accidentally, in such a way that the perturbation should not change anything in model prediction from human perspective (due to being too insignificant or almost imperceptible). However, adversarial examples are proven to significantly damage the performance of models. The problem is still awaiting a working solution, which could secure the models working in production systems [Deldjoo et al., 2021, Di Noia et al., 2020].

These challenges, coupled with the rising importance of recommender systems in business, call for increasing research efforts directed at creating faster, more scalable, more versatile and more robust algorithms. These are the exact purposes of my thesis. In my work I propose new algorithms for the most pressing problems spanning various branches of recommender systems. Novel algorithms presented here contribute to solving the problems of low speed, high complexity, and lack of multimodal models. Accompanying interpretability and robustness-enhacing methods have been proposed, which strengthen the applicability of the proposed recommendation algorithms.

As a practical business application, my work has led to the creation of a comprehensive recommender system, which is being commercialized under the name of the Monad platform, developed at the company Synerise. Monad is currently being introduced to large enterprises, such as international banks and e-commerce stores.

1.1 Research Objectives

This thesis is built around the following research objectives:

1. Improving the scalability, performance, and properties of embedding algorithms which exploit graph structure of the data. To this end, I perform an analysis of graph neural networks, which are the state-of-the-art graph embedding algorithms today. I identify the source of their inefficiency, and prove that it is possible to simplify the graph neural network into an algorithm which avoids the scalability problem while keeping high quality of produced embeddings.

2. Formulation and evaluation of a recommendation algorithm which can ingest additional multimodal data. At the same time it needs to keep to the requirement of low network complexity, without computationally expensive neural network structures.

3. Formulation and evaluation of new algorithms of content-based recommendation, in particular enhancing the ability to search for similar items in an accurate and fast way, avoiding the known pitfalls of existing uni-modal and cross-modal approaches.

4. Formulation and evaluation of new approaches to recommender system robustness enhancement. Discovering yet unknown vulnerabilities in models, and possible ways of fixing the vulnerabilities.

Together, the proposed approaches create a full, comprehensive recommender system.

1.2 Research Hypotheses

Based on the analysis of the research objectives, I formulate a general hypothesis: current recommender algorithms can be further improved in order to achieve better quality, scalability and multimodal integration. This can be achieved with novel ideas such as unsupervised and self-supervised algorithms inspired by compression methods, density estimation methods and graph neural networks. Based on this, I introduce five main hypotheses:

1. Certain forms of graph neural networks (GNNs), usually found in recommender systems, can be simplified by removing weight optimization and non-linearities to obtain a purely unsupervised algorithm based on weighted averaging of node embeddings. Such simplified algorithm can hold high quality of performance. At the same time, it can be much faster to train than a full GNN. To validate this hypothesis I introduce the Cleora algorithm [Rychalska et al., 2021a] and conduct extensive comparison tests with other popular

graph embedding algorithms. Quality of Cleora embeddings is measured in popular metrics such as the Mean Reciprocal Rank (MRR) and Hit Rate (HR@K). The algorithm speed is measured in terms of wall clock time of the embedding computation. The results are presented in Section 4.1.

2. Density estimation-based modeling of user item spaces allows to increase the quality in top-k recommendation. To verify this hypothesis, I propose a top-k recommender approach based on EMDE (Efficient Manifold Density Estimator) [Dąbrowski et al., 2021]. Quality is measured by Recall@K and Normalized Discounted Cumulative Gain (NDCG). The results are presented in Section 4.2.

3. Centroid-based image retrieval combined with a neural model allows to increase quality and speed of unimodal image-based content recommendations. To verify this hypothesis, I propose the centroid-based image retrieval approach [Wieczorek et al., 2021]. Quality is measured by Mean Average Precision and Accuracy@K metrics. Measurements of model latency are conducted by comparing the speed of inference with instance-based evaluation and centroid-based evaluation. The results are presented in Section 4.3.

4. Transformation of embeddings from non-matching feature spaces to a unified feature space allows to both increase the quality and speed of cross-modal content recommendations. To verify this hypothesis, I propose the T-EMDE algorithm Rychalska et al. [2021b] - a fast algorithm which expresses multimodal representations in a joint representation space. Quality is measured by MRR and Recall@K metrics, while latency is measured by wall clock times of model training and inference phase. The results are presented in Section 4.4.

5. Increasing the resilience of models against some adversarial examples allows to strengthen the model against related adversarial examples. To validate this hypothesis, I create the WildNLP framework [Rychalska et al., 2019] which works on the textual modality. The WildNLP is tested on a wide array of existing textual models and metrics. The results are presented in Section 4.5.

1.3 Original Elements of the Thesis

In terms of this thesis, I have conducted a number of original works.

1. I proposed the method of robustness enhancement with WildNLP. The method relies on enhancing model robustness against one adversarial example by retraining models on inputs modified with on a conceptually similar adversarial example.

2. I proposed two methods of analysing the reliability of textual models: 1) a novel approach of antonym swapping 2) analysis of the statistical properties of the representations in intermediate neural network layers.

3. I proved that a density estimation method for multimodal data (EMDE) can serve as an effective recommender when adequately combined with a neural network.

4. I proved that graph convolutional neural networks, used for computation of node embeddings, can be simplified to weighted matrix multiplication, which retains embedding quality, and offers significant gain in speed and scalability.

5. I proposed the T-EMDE algorithm, which is a differentiable density estimation algorithm inspired by EMDE. EMDE, being non-differentiable, cannot be used in many settings and relies on static region creation within the embedding space. T-EMDE, on the other hand, uses the concept of movable centroids in the embedding space, which can be aligned during training.

6. I proved that the centroid-based retrieval is an effective mechanism in uni-modal content recommendation.

7. I created a set of benchmarks for testing scalability of graph embedding approaches.

8. I performed a large portion of work connected with data analysis and data preparation, literature search comprising all research hypotheses and implementation of the methods.

1.4 Publications

I have written 28 research articles, preprints and technical reports overall, with 223 total citations including 88 citations of articles comprising this PhD thesis[3]. I have an h-index of 8.

[3]based on Google Scholar `https://scholar.google.pl/`

Table 1: List of publications which support this thesis. Number of citations computed based on Google Scholar.

Section	Publication	MEiN Points	Citations
2.1	Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, Jacek Dąbrowski and Przemysław Biecek. ”Cleora: a Simple, Strong and Scalable Graph Embedding Scheme” accepted for publication in the proceedings of the International Conference on Neural Information Processing (ICONIP 2021)	140	12
2.2	Jacek Dąbrowski*, Barbara Rychalska*, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Babel, Andrzej Michałowski, and Adam Jakubowski. ”An efficient manifold density estimator for all recommendation systems” accepted for publication in the proceedings of the International Conference on Neural Information Processing (ICONIP 2021); *-equal contribution	140	10
2.3	Mikołaj Wieczorek*, Barbara Rychalska*, and Jacek Dąbrowski. ”On the Unreasonable Effectiveness of Centroids in Image Retrieval” accepted for publication in the proceedings of the International Conference on Neural Information Processing (ICONIP 2021); *-equal contribution	140	39
2.4	Barbara Rychalska*, Mikołaj Wieczorek* and Jacek Dąbrowski. ”T-EMDE: Sketching-based Global Similarity for Cross-Modal Retrieval” preprint `https://arxiv.org/pdf/2105.04242.pdf`; *-equal contribution	0	0

2.5	Barbara Rychalska*, Dominika Basaj*, Alicja Gosiewska, and Przemysław Biecek. ”Models in the Wild: On Corruption Robustness of Neural NLP Systems” In Tom Gedeon, Kok Wai Wong, and Minho Lee, editors, Neural Information Processing, pages 235–247, Cham, 2019. Springer International Publishing. ISBN 978-3-030-36718; *-equal contribution	140	18
2.5	Barbara Rychalska*, Dominika Basaj*, Anna Wróblewska, and Przemyslaw Biecek. ”Does It Care What You Asked? Understanding Importance of Verbs in Deep Learning QA System” published in Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP; *-equal contribution	0	9
2.6	Jacek Dąbrowski and Barbara Rychalska. ”Synerise Monad - Real-Time Multimodal Behavioral Modeling” published in Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM ’22).	140	0

Table 1 displays all articles which comprise this PhD thesis. Most of the articles have already been accepted to international conferences or workshops. Six of these articles describe algorithms regarding various parts of a recommender system. The last, seventh article - ”Synerise Monad - Real-Time Multimodal Behavioral Modeling” [Dąbrowski and Rychalska, 2022] - describes a commercial platform composed of the algorithms and ideas introduced in other papers.

Table 2 displays some examples of my work not included within this thesis.

Table 2: List of publications not taken into account within this thesis. The list contains only the most important articles. Number of citations was computed based on Google Scholar.

Publication	MEiN Points	Citations
Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak, and Piotr Andruszkiewicz ”Samsung Poland NLP team at SemEval-2016 task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methodsto measure semantic similarity” Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 602–608, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1091. `https://aclanthology.org/S16-1091`	0	69
Barbara Rychalska and Jacek Dabrowski. ”Synerise at SIGIR Rakuten Data Challenge 2020: Efficient manifold density estimator for cross-modal retrieval” The 43th International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR) eCom Workshop Challenge, 2020.	0	3
Dominika Basaj, Witold Oleszkiewicz, Igor Sieradzki, Michał Górszczak, Barbara Rychalska, Tomasz Trzciński and Bartosz Zieliński. ”Explaining Self-Supervised Image Representations with Visual Probing” Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI-21)	200	13

1.5 Industrial Significance

My PhD was conducted in terms of the Industrial Doctorates (”Doktoraty Wdrożeniowe”) Programme. The works presented in this thesis are already used in production by the company Synerise S.A., which was my PhD host. The models and methods contribute to the core business of the company, which is a multipurpose and multimodal recommender platform. The speed and scalability aspects of the introduced models proved to be of core importance and allowed to process data from even the biggest customers, such as e-commerces, banks, and telecoms. The successes of these algorithms in prestigious international research competitions such as SIGIR Rakuten Challenge 2020 [Rychalska and Dabrowski, 2020] (1st place), Twitter Recsys Challenge 2021 (2nd place), KDD Cup 2021 [Daniluk et al., 2021a] (3rd place), Booking.com WSDM Challenge 2021 [Daniluk et al., 2021b] (2nd place), contributed greatly to gaining a competitive edge on the market and obtaining international recognition. Our algorithms performed against competitors such as Google DeepMind, NVIDIA, Baidu, Intel, Rakuten Research, and many other leading industry and academic research institutions.

The algorithms described in this thesis are currently used in two commercial products. One of them is the Synerise platform[4], a real-time interaction management and automation platform for large enterprises which generate behavioral event data. The Synerise platform is used by companies such as Żabka, mBank, Carrefour, CCC, Rossmann, and many other large data-rich businesses in European Union, Great Britain, Middle East and South America. The processed data volumes reach tens of millions users and tens of thousands of products for each individual company served by the plafrorm. The graphs processed by the algorithms are up to billions of nodes in size.

The second product in which the presented algorithms are used is the Monad platform, which is not yet released to the market, but is being tested in a few pilot projects with selected companies. Monad assists Data Science teams with creating behavioral data models, such as recommender systems, churn prediction models, or propensity prediction models. Monad currently processes hundred-million node interaction graphs at e-commerce and banking companies, among other examples [Dąbrowski and Rychalska, 2022].

Due to a successful combination of research and business objectives, Synerise received a number of international business awards, particularly for fast growth and the innovative aspect of its products. Recently the company was recognized by The Financial Times as the 17th top growing company in European technology sector in their ”FT1000 Europe's Fastest Growing Companies 2022” ranking[5]. Synerise was also recognized among the top three of all companies in Europe in the field of artificial intelligence and big data.

Open-sourced versions of described algorithms are used by market leaders. An official confirmation of use has been made by Zomato - a large restaurant aggregator operating in 10,000 cities across 24 countries. Having searched the literature for algorithms which would satisfy tight latency and quality constraints, they have picked our algorithms for their production system, as

[4]`https://synerise.com/`

[5]`https://www.ft.com/ft1000-2022`

described in their research blog post [6]. They note that our algorithms have been more successful than popular state-of-the-art industry algorithms such as GraphSAGE [Hamilton et al., 2017].

1.6 Acknowledgements

The works described in this thesis have been partially supported by my Preludium grant no 2018/31/N/ST6/02273 entitled ”MAD-NLP: Multiaspectual Diagnostics of NLP Systems”, awarded by the National Science Centre (NCN) in 2018. Moreover, I have worked at the NCBiR grant no POIR.01.01.01-00-0695/19 in Synerise. Furthermore, I have pursued my research interests on a 2-month research stay at Nanyang Technical University Singapore in Professor Shafiq Joty’s research lab. The stay was financed in terms of RENOIR Project by the European Union’s Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement No 691152. I was also a laureate of the Iwanowska Programme by NAWA, with a research stay at Oxford University in Professor Yarin Gal’s research lab ”Oxford Applied and Theoretical Machine Learning Group” (OATML). The stay was unfortunately cancelled due to the COVID-19 pandemic. For a number of years I was a laureate of the Rector’s Scholarship for the best PhD students at the Warsaw University of Technology. During my work on the PhD I was a member of the research group MI^2 DataLab led by prof. dr hab. inż. Przemysław Biecek at the Warsaw University of Technology.

1.7 Structure of the Thesis

In Section 2 I define the necessary background knowledge on neural networks. Section 3 presents an introduction to recommender systems and metrics used for measuring their performance. Then in Section 4 I present my research results.

- Research Hypothesis 1 is discussed in Section 4.1,
- Research Hypothesis 2 is discussed in Section 4.2,
- Research Hypothesis 3 is discussed in Section 4.3
- Research Hypothesis 4 is discussed in Section 4.4,
- Research Hypothesis 5 is discussed in Section 4.5

[6] https://www.zomato.com/blog/connecting-the-dots-strengthening-recommendations-for-our-customers-part-two

Finally, in Section 5 I summarize my results and the conclusions from this thesis.

2 Introduction to Neural Networks

Before transitioning to the topic of recommendation, it is necessary to describe in detail the models forming the backbone of many successful recommender approaches: the neural networks. These explanations will be vital for the understanding the most recent recommendation models, described in following sections, as well as models created in terms of this thesis.

2.1 The Perceptron

Research on neural networks started with models loosely inspired by natural neurons found in the human brain. One of the first models is the well-known Perceptron model [McCulloch and Pitts, 1943, Rosenblatt, 1958]. Here we focus on the more advanced version from Rosenblatt [1958] displayed in Figure 1. Rosenblatt's artificial approximation of the natural neuron consumes a series of inputs and returns a single value. This is similar to natural neurons found in the brain, which are interconnected within large networks and receive multiple signals from other neural cells. Each input x_i is multiplied with an associated, trainable weight w_i, followed by a summation of all $x_i \cdot w_i$ products. This way, a form of a weighted sum of inputs is computed. Then, additional bias value b is added to the summed result. The bias b has a special function, as it is not directly linked to any of the inputs – simulates an input whose value is always 1. The intuition behind the bias is that it creates a trainable threshold which is compared to the aggregated value of the real inputs multiplied by weights. Natural neurons have a similar mechanism – they propagate a signal only when certain conditions are met, usually dependent on a measurable threshold.

The output of the Perceptron is a binary value, either 1 or 0:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The learning rule relies on an iterative update of weights w and the bias b such that the classification result should reach the correct output value (either 0 or 1) for each training example, as given in the training set. If the classification result is already correct, no update takes place. However, if the classification result is incorrect, an update to w and b is performed with the following rule, with $TrueClassification$ and $PredictedClassification$ assuming values either 0 or 1:

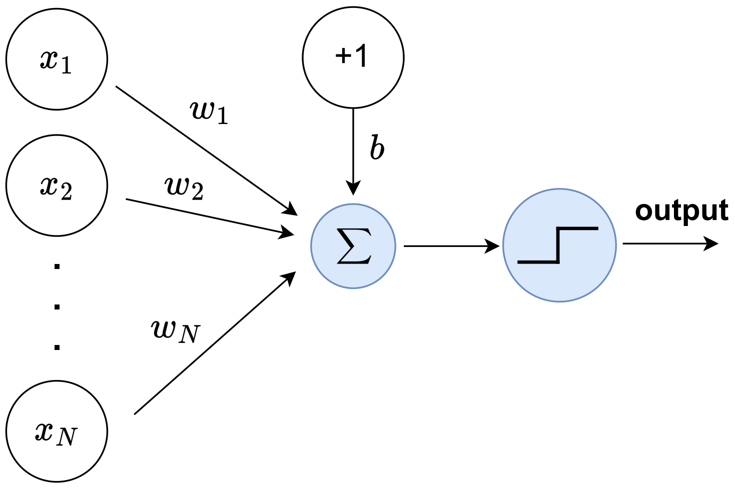


Figure 1: The classic Perceptron model from Rosenblatt [1958].

$$w = w + (TrueClassification - PredictedClassification) \cdot x \,, \tag{2}$$

$$b = b + (TrueClassification - PredictedClassification) \,. \tag{3}$$

This is done in order to move the classification boundary such that the current example gets classified correctly (over 0 if the desired result is 1, or below zero if the desired result is 0). This learning rule requires that the learned problem be linearly separable, otherwise learning will not terminate.

Composed of just one neuron and with the simplistic learning rule, the classic Perceptron model is now long obsolete. However, the modern neural networks still retain the trainable weights and biases, and use a learning rule to iteratively update them during the course of training. Their learning method - the backpropagation - is however very different, with many more variations and upgrades.

Modern neural networks have developed into a wide array of approaches, based on the structure of inputs they consume and the tasks they are expected to perform. Among the most importand classes of neural networks are the feedforward networks, recurrent neural networks, autoencoders, graph neural networks, and attention-based networks.

2.2 Feedforward Neural Networks

Proceeding to the models which are still in use today, it is necessary to start with the simple feedforward neural network (FFN) [Schmidhuber, 2015]. Feedforward neural networks usually form a series of layers, with each layer containing many basic neural units - artificial neurons. The neurons are linked into a network with connections whose strength is modified during training. Modern neural networks usually have many layers of neurons, with the first layer known as the input layer, the middle layers known as hidden layers, and the last layer called the output layer. See Figure 2 for an example. In a feedforward network, the information moves in only one direction — forward — from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

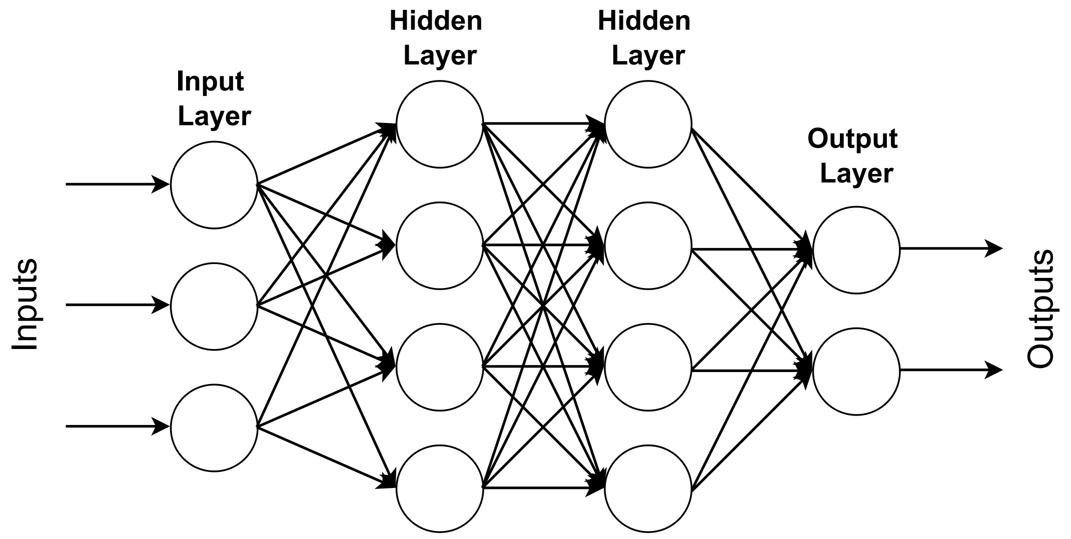


Figure 2: A multilayer feed-forward network.

The process of training feed forward networks (and other neural models mentioned in this thesis) is composed of two stages: the forward pass and the backward pass.

2.2.1 Forward Pass

The process of neuron update with a series of input values is displayed in Figure 3. Each neuron receives a set of inputs via connected edges with attached weight values. Weights represent the strength of connection. As set of input values a_i enter the neuron, each gets multiplied by a connection-specific weight value w_i and summed. Then, a bias b term is added. The resulting output is fed to a non-linear activation function, such as the sigmoid. The application of a non-linear activation function is important as it allows the network to represent very complex functions at the output. Otherwise, the update formula would create simple linear combinations of input features.

Let a_{out} denote the output value of a neuron and σ denote a suitable activation function, for

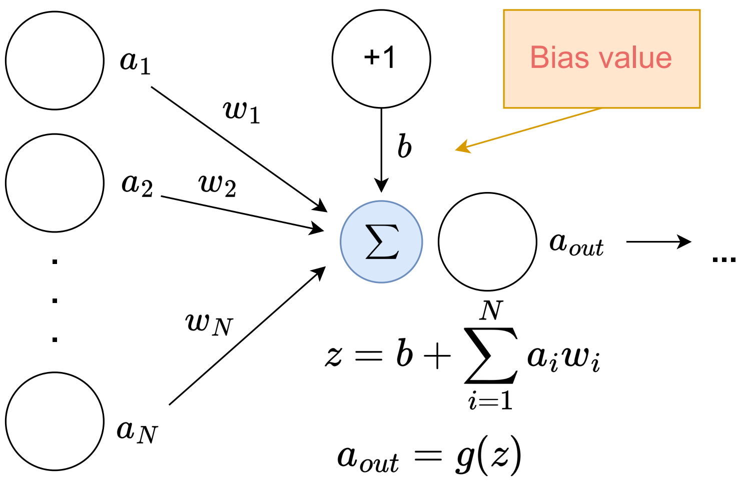


Figure 3: The update rule in a neural network.

example the sigmoid function. Then, the update rule of a single neuron is given as follows:

$$z = b + \sum_{i=1}^{N} h_i w_i \,, \tag{4}$$

$$a_{out} = \sigma(z) \,. \tag{5}$$

2.2.2 Backward Pass

After the full forward pass (computed for all neurons), the last layer of the network usually contains values which correspond to desired targets (e.g. a target class in classification). Based on the last layer's values and a desired target values, a loss function is used to calculate the degree of error in the network. There exist many loss functions which can be used with neural networks, such as mean squared error or cross entropy loss, depending on the neural network task (such as classification or regression) - see Section 2.2.3.

After calculation of the loss, it is then fed to the backpropagation algorithm, which aims to minimize the loss function by adjusting network's weights and biases. In order to do this, the gradient is computed - which is a measure of the change required from each trainable parameter in order to lower the loss function value.

With a given trainable weight w, the backpropagation algorithm is given by the chain rule:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w} \,. \tag{6}$$

This formula allows to compute the contribution of each trainable weight w to the obtained network error. After computation of gradients, the weights are updated with their gradient values (modified by a user-defined learning rate). After a number of iterations of paired forward and backward passes, the network loss is expected to reach a plateau.

The descibed forward-backward update formula is universally applied also to other, more complex types of neural networks, which are described in following sections.

2.2.3 Loss Functions

Neural Networks use many loss functions, depending on the task performed by the network. In this section, I define popular loss functions for the tasks of regression and classification.

Definition 1. ***Mean Squared Error (MSE)*** *is a simple loss function computed for regression tasks. With y denoting true output value, $\hat{y}$ denoting predicted output value returned by the network, and N being the number of examples, MSE is computed as follows:*

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \,. \tag{7}$$

Definition 2. ***Cross Entropy Loss*** *is a simple loss function for binary classification tasks, which can be extended to multiple class scenario. It punishes the network for predicting large probabilities for the incorrect class. With y denoting true output class (0 or 1) in the binary classification case, $\hat{y}$ denoting predicted probability of class 1, and N being the number of examples, cross entropy loss is computed as follows:*

$$L = \frac{1}{N}\sum_{i=1}^{N} -(y_i \cdot log(\hat{y_i}) + (1{-}y_i) \cdot log(1{-}\hat{y_i}))\,. \tag{8}$$

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are models devoted to representing sequential data, in which the ordering of items is of primary importance. RNNs have gained prominence in Natural Language Processing (due to their ability to accurately model sequences of words) [Goldberg, 2017] and in recommendation in modeling the order of purchased items in a shopping session [Donkers et al., 2017], among other fields.

In RNNs the connections between nodes form a sequence along the temporal dimension, which can also be viewed as a temporal loop. RNNs exploit an internal state (memory) to process variable length sequences of inputs. The internal state gets updated with information at each passing time step.

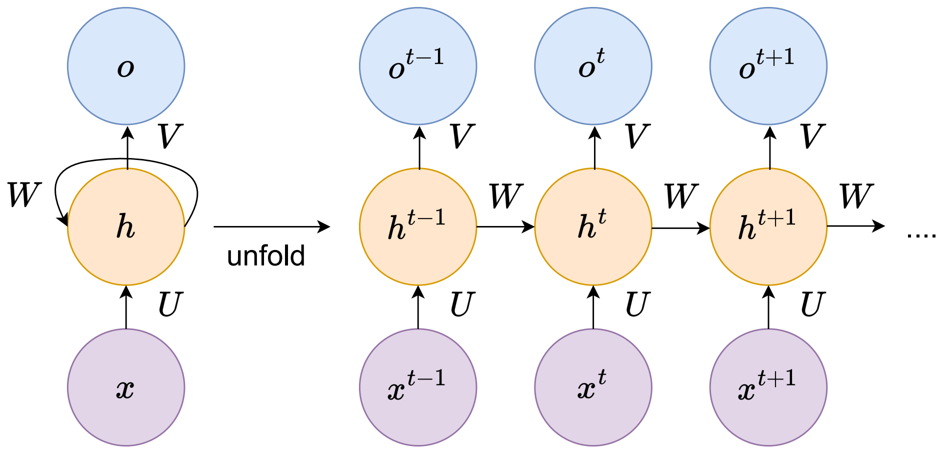


Figure 4: Recurrent Neural Network with unfolding through time.

A simple recurrent model can be expressed as follows (see also Figure 4):

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h)\,, \tag{9}$$

$$o_t = \sigma(V_h h_t + b_y)\,, \tag{10}$$

where x_t is the input vector, h_t is the hidden state at time step t, h_{t-1} is the hidden state at previous time step $t-1$, o_t is the output at time step t, W_h, U_h, V_h are trainable weight matrices, b_h, b_y are network bias terms, and σ - non-linear activation function of the network.

One of the most prominent examples of RNNs is the Long-Short Term Memory (LSTM) model and Gated Recurrent Unit (GRU). They employ elaborate gating mechanisms in the recurrent memory module to fix gradient flow problems in the backpropagation stage.

2.4 Autoencoders

An autoencoder is an artificial neural network which learns to express input data in terms of an arbitrary encoding. This encoding is most often small in terms of dimensionality and can be thought of as a form of compression. Autoencoder architecture allows for learning in a fully unsupervised setting, i.e. on data which do not have any labels or target values assigned. Due to their ability to perform data compression, they may be used for dimensionality reduction or embedding learning.

Figure 5 shows a basic form of an autoencoder. It usually has a number of layers with feed-forward connections and is composed of two main parts: the encoder and the decoder. Between

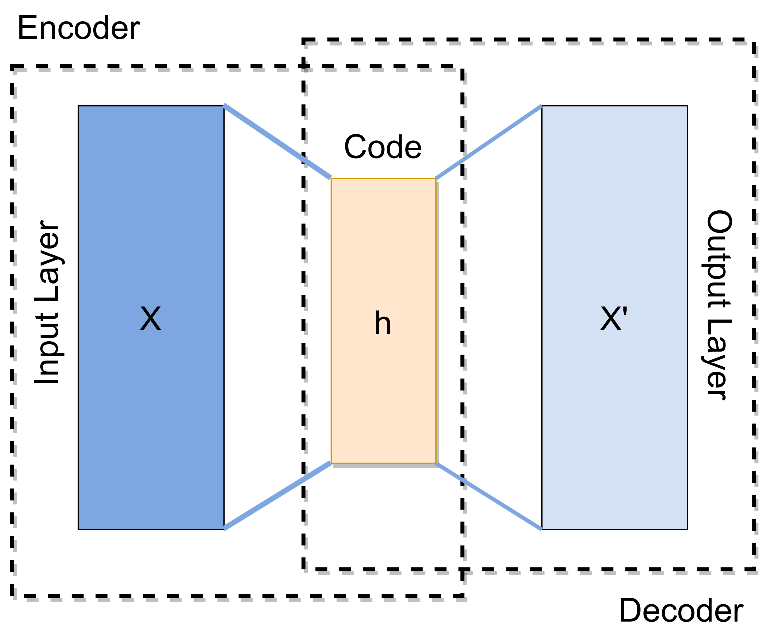


Figure 5: Autoencoder Neural Network. A middle hidden layer is used to squash the input representation, and then expand it back at output.

them, a small bottleneck is introduced in a form of a hidden vector. This vector represents the compressed input vector (its *latent encoding*). Formally, the forward run of a simple 1-layer autoencoder can be expressed with following equations:

$$h = \sigma(Wx + b)\,, \tag{11}$$

$$x' = \sigma'(W'h + b')\,. \tag{12}$$

Here, x denotes the input vector, x' - the vector which is reconstructed by the network, h - the hidden compressed vector, W and W' - weight matrices optimized by the network, b, b' - bias terms of the network, and σ, σ' - non-linear activation functions of the network. The training objective is to minimise a reconstruction error between x and x', using a desired loss function (for example one of the loss functions defined in Section 2.2.3).

2.4.1 Variational Autoencoders

Variational Autoencoders (VAEs) share the basic idea of representation squashing with classic Autoencoders, however they operate in the probabilistic domain. In a VAE it is assumed that input data is sampled from a parametrized distribution which is called a prior, just as in Bayesian

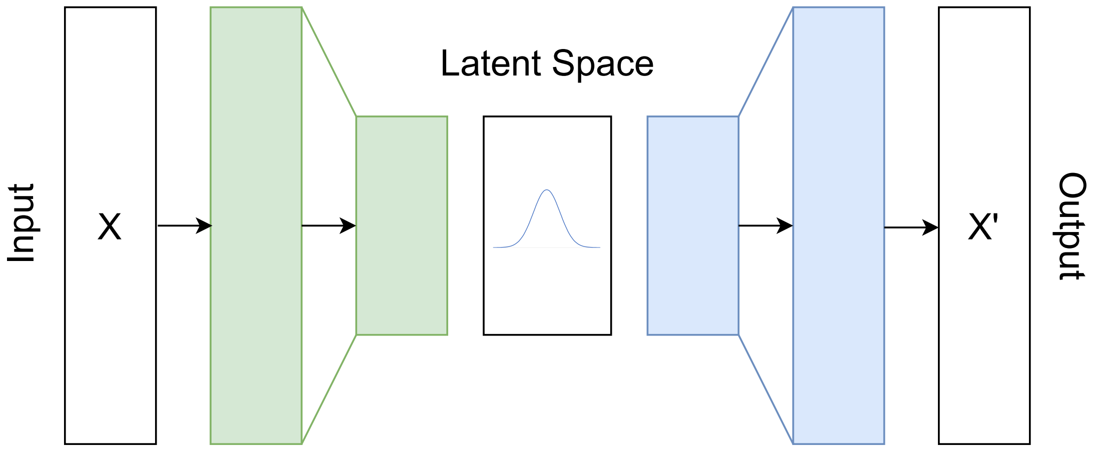


Figure 6: Variational Autoencoder Neural Network (VAE).

inference. The reconstruction error is composed of two parts: the regular reconstruction loss (such as in regular Autoencoders) and Kullback–Leibler divergence between the parametric posterior and the true posterior. This term is used to punish the model for generating outputs not from the desired distribution.

The advantage of Variational Autoencoders over classic Autoencoders is that in VAEs the learned representations are more evenly distributed over the representation space [Makhzani et al., 2015]. With classic Autoencoders, the only restriction imposed on the model is that the latent representations should allow to reproduce the input. In practice, learned representations tend to cluster together unevenly in the representation space. Variational Autoencoders, on the other hand, are trained to produce smooth representations, without discontinuities such as gaps between clusters of data.

Let x denote the input vector and h be the latent encoding of x. We assume that x comes form an unknown distribution p_θ defined by a set of parameters θ. Then we can formulate the following dependencies:

$$p_\theta = \int_h p_\theta(x, h)\, dh \ = \int_h p_\theta(x|h) p_\theta(h)\, dh \ . \tag{13}$$

Here we arrive at the conditional distribution $p_\theta(x|h)$ which will be represented by the decoder. The encoder will represent the distribution $p_\theta(h|x)$, which is approximated by a Gaussian distribution $q_x(h)$. Under such formulation, the neural network can be applied to adequately optimize the unknown parameters of distributions involved.

2.5 Graph Neural Networks

Graph Neural Networks (GNNs) is a class of neural network for processing graph data structures. Graph structures represent the relations (edges) between a collection of entities (nodes). Formally, a graph G can be defined as $G = (V, E)$, where V is the set of nodes, and E are the edges between them. If there are directional dependencies between nodes then edges are directed. If not, edges are undirected.

Graphs can be found in many areas of life, such as biology and chemistry (molecules, DNA structure), social networks, road networks, or behavioral purchase data where consumers can be linked to the items they buy.

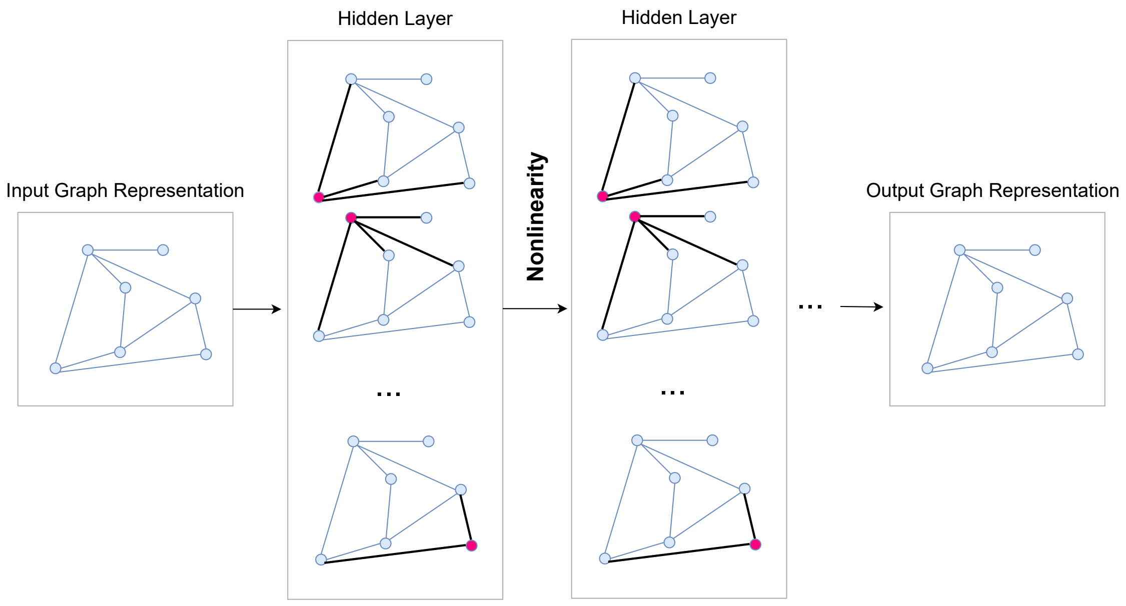


Figure 7: Illustration of a Graph Neural Network. In each hidden layer, the neighborhoods of all nodes are considered.

GNNs have been created to address the issue of processing such complicated and interconnected data. Previously introduced neural architectures could process more regular data, such as fixed-size images or sequences of words. It is worth noting that these simple structures can also be understood as graphs. For example, in the case of text, each word can be modeled as a node, and two neighboring words can be linked with an edge. However, most real-life graphs are much more complex and irregular. They often have arbitrary numbers of nodes, and some nodes can be hubs (highly connected with many neighbors) while others may have extremely few edges.

GNNs often take the form of message passing neural network paradigm. The term 'message passing' refers to the fact that in GNNs, a node representation is created by aggregating representation of its neighboring nodes. This way, nodes can propagate messages throughout the graph by communicating them to their neighbors (see Figure 7). Message passing can be performed in a variety of ways depending on the model. For example, Graph Convolutional Networks (GCNs) [Kipf and Welling, 2017] use the following update rule:

$$H^{[i+1]} = \sigma(W^{[i]} H^{[i]} A^*) \,. \tag{14}$$

Here, $H^{[i]}$ and $H^{[i+1]}$ denote matrices containing hidden node representations from layer i and $i+1$, respectively. $W^[i]$ is the trainable weight matrix from layer i and A^* is the normalized adjacency matrix of the graph. Given a graph with n nodes, its adjacency matrix has dimensionality $n \times n$ and indicates whether pairs of vertices are adjacent or not in the graph (usually with 1 representing an edge and 0 representing lack of an edge). The insertion of A^* in the forward pass equation enables the model to learn the feature representations based on node connectivity. Thus, each node considers only the hidden vectors of its neighbors during computation of $H^{[i+1]}$.

2.6 Attention-Based Neural Networks

Attention is a mechanism which allows the network to learn intricate patterns of correspondence between input and target sequence/set elements. One of the first works which introduced this popular mechanism was [Bahdanau et al., 2015]. The most famous formulation of attention is that of the Transformer self-attention [Vaswani et al., 2017]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \,, \tag{15}$$

where Q is the query vector, K is the key vector, V is the value vector, and d_k is the dimension of the K vectors. Q, K, and V vectors are obtained by running the same inputs x through specialized neural network layers. This way, different abstractions of inputs get created and are then multiplied against each other to represent various interactions between sequence elements. Attention allows to detect sophisticated patterns in sequences, as each sequence element can consider all other elements and signal that the network should devote more focus to even very small but important parts of the data.

Unfortunately, the multiplication QK^T results in quadratic complexity of the formula, which proves problematic for production systems and has spawned multiple attempts at reducing the computational complexity [Beltagy et al., 2020, Wang et al., 2020b, Tay et al., 2021]. Various forms of attention are widely applied in the field of recommender systems and content retrieval systems [de Souza Pereira Moreira et al., 2021, Diao et al., 2021, Hu et al., 2019].

3 Recommender Systems

3.1 Background

The dynamic growth of Internet-based services has generated huge amounts of digital information. Globally, the number of Internet users increased from 413 million in 2000 to over 4.9 billion in 2021[7]. With the growth of social media and various forms of Internet-based commerce and advertising, the growth of digital data is bound to be dynamic in the coming years.

Internet hosts various forms of information, such as texts, images, videos, and audio. Data is often stored on massive social networks, such as Facebook and Twitter, or large e-commerce sites, such as Amazon or eBay. Amazon has more than 200 million visitors each month and in 2017 shipped over 5 billion products all around the globe[8][9]. With such massive information sources, it becomes challenging for users to look for relevant information. To deal with this issue, two important information processing technologies have been developed [Belkin and Croft, 1992]: information retrieval and recommendation. In information retrieval (IR), or search system, users explicitly present their goals by sending a search query, as done on Google. On the other hand, in an e-commerce platform such as Amazon, the service is expected to guess the goals of users from their behavior history and proactively present them the product which might be interesting. This goal is known as recommendation.

3.2 Multiple Aspects of Recommender Systems

Recommender systems span a broad range of areas, often providing various functionalities related to product retrieval in a single production system [Jannach et al., 2010]. Thus the word ”recommendation” can be understood in many ways, which are complementary. One can sketch two chief recommendation branches: **content-based recommendation** [Jannach et al., 2010] and **collaborative recommendation**. They cover two important use-cases: recommendation based on past user interactions and recommendation based on search of similar products.

In content-based recommendation, users are able to search for products similar to other products based on intrinsic product characteristics, such as the appearance, description or product features. This is the case for example when a recommender is expected to find a similar piece

[7]https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/

[8]https://www.nchannel.com/blog/amazon-statistics/

[9]https://www.theverge.com/2018/1/2/16841786/amazon-prime-2017-users-ship-five-billion

of clothing based just on a photo of a clothing seen by the user previously. Content-based recommendation can also materialize in the form of cross-modal retrieval, where items are recommended based on multiple modalities. For example, the input can be a textual description of an item previously bought by a user and the task could be to find the closest matching product from inventory. Content-based recommendation can also help in cases where some products do not yet have any user interactions (e.g. due to beings freshly introduced to inventory). Then we can recommend them on the basis of the product similarity alone.

In collaborative recommendation the users themselves produce training data - by clicking, adding items to basket, and buying them. The recommender algorithm then assesses the items based on common buying patterns. Two products are deemed similar if they exhibit similar interaction patterns from users. The types of interactions can be user-defined, e.g. a certain company might be interested in separate modeling of buy, click or view interactions. The outputs can be usually of two types: 1) a probability that the user will interact with a given item, or 2) an ordered list of recommended items [Jannach et al., 2010].

Collaborative recommendation is often divided into two groups based on the structure of user interaction history:

- **Session-based Recommendation.** Session-based recommendation is one of the subtypes of recommendation based on the format of user inputs. In session-based recommendation user interactions with the item inventory takes the form of an ordered session. The session usually represents a shopping or browsing session. Each user usually has multiple sessions.

- **Top-K Recommendation.** In contrast to session-based recommendation, in top-k recommendation each user is characterized with an unordered set of historical interactions. The set characterizes all interactions gathered together, usually without an assigned date. Top-k recommendation can arise for example in streaming services where the customers watch movies over a long timespan, and each user is characterized with a set of movies they liked throughout their complete history. It also materializes in cases where a shopping basket is scanned and saved at the end of shopping, and the order of scanning is random.

The quality of all kinds of multimodal recommenders (e.g. dealing with item pictures, textual descriptions and interaction history) is often dependent on the quality of numeric representations of image, text, and interactions. These numeric representations are often produced by

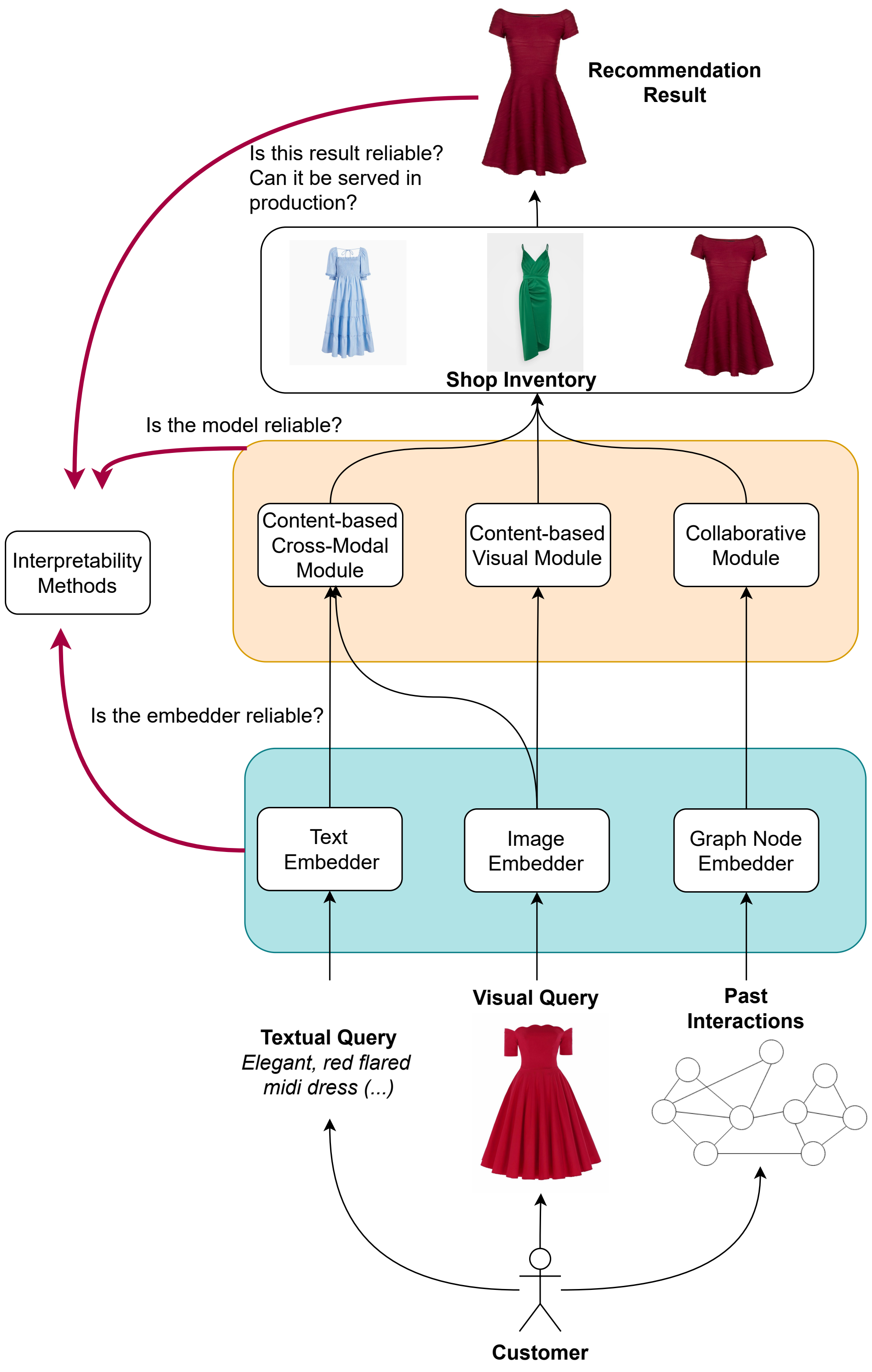


Figure 8: A multi-purpose recommendation system. The system spans multiple stages: the various modality embeddings (for text, graph nodes, visuals), both collaborative and content-based recommendation and interpretability/robustness control methods to guarantee reliability.

specially designed unsupervised methods, which extract knowledge from images, text and interaction data and transform it into numeric form. Following the work of Mikolov et al. [2013], the idea of *embedding vectors* has taken firm roots in many areas of machine learning, including recommender systems. Embedding vectors are vectors whose relative similarities correlate with semantic similarity of items they designate. The idea of vector representation is nowadays extremely popular in representation of text [Pennington et al., 2014], image [Chen et al., 2020b] and interactions [Akyildiz et al., 2020], among many others modalities of data. Thus, their study and enhancement remains crucial for recommender performance.

Apart from recommendation algorithms themselves, many accompanying algorithms also play a vital role in ascertaining a successful operation of a recommender in a production setting. An important class of algorithms are interpretability methods, designed to ascertain that our recommendations are reliable, i.e. they are high quality and not affected by some specific artifacts in training datasets, which are carried over to production. It is desirable for recommendation models to be interpretable, i. e. we should be able to discover why they made a certain decision (e.g. which inputs had the decisive impact on the decision). Also, many recent articles describe the problem of *adversarial examples* [Goodfellow et al., 2015, Athalye et al., 2018] - inputs to a model designed in such a way as to deceive the model into a wrong output decision. The analysis of robustness of models to this kind of attack is an important practical feature.

A full view of a recommender system with all the abovementioned aspects is shown in Figure 8. It features a recommender which operates on three input modalities: past interactions, visual data of items, and textual descriptions. Each modality is processed by a dedicated embedding model. Resulting embeddings are fed to a recommender model (comprised of collaborative and content-based methods). Each model returns a ranking list of recommended items. The outputs of the recommender and models which comprise the recommender are analyzed by intepretability and robustness detection methods.

3.3 Recommender Evaluation Metrics

Recommender algorithms operate on various kids of inputs, yet the outputs are usually of one form: a ranked lists of recommended items, with most recommended items at the top. Thus, the models are usually evaluated with a set of ranking-based metrics. These metrics are usually computed on top K items retrieved in the ranking, with K being somewhere between 1 to 100, depending on the need (e.g. number of recommended items that can be displayed on a website).

Definition 3. ***HitRate@K (HR@K)*** *is a simple binary metric that looks at whether any of the top-K recommended items are relevant for the given user.*

Definition 4. ***Recall@K*** *calculates the proportion of recommended items that are relevant out of the total number of relevant items. With relevant items in the top K list denoted as R_K and all relevant item set denoted as A, we have:*

$$Recall@K = \frac{|R_K|}{|A|}\,. \tag{16}$$

Definition 5. ***Precision@K*** *calculates the proportion of recommended items that are relevant out of all of recommended items in the top K returned items. With relevant items in the top K list denoted as R_K we have:*

$$Precision@K = \frac{|R_K|}{K}\,. \tag{17}$$

Definition 6. ***Average Precision (AP@K)*** *is the sum of Precision@K for different values of K divided by the total number of relevant items in the top K results.*

Definition 7. ***Mean Average Precision (mAP@K)*** *is the mean of Average Precision@K for the whole dataset.*

Definition 8. ***Cumulative Gain*** *is calculated as the sum of all the relevance scores in a recommendation set. In contrast to previously introduced metrics, Cumulative Gain-related metrics allow to not only compute whether relevant items are in the recommendation rankings, but also assess their explicit scores. With the relevance score at rank i denoted as rel_i, we have:*

$$DG = \sum_{i=1}^{n} rel_i\,. \tag{18}$$

Definition 9. ***Discounted Cumulative Gain (DCG@K)*** *is calculated as the Cumulative Gain with added correction which values more the position of relevance scores. With the relevance score at rank i denoted as rel_i, we have:*

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{log_2(i+1)}\,. \tag{19}$$

Definition 10. ***Ideal Discounted Cumulative Gain (IDCG@K)*** *is a helper metric which calculates what the score DCG@K would be if the items in the ranking were sorted by the true, that is ideal (unknown for the recommender model) relevance. With REL_p defined as a list of relevant items up to position p and relevance score at rank i denoted as rel_i, we have:*

$$IDCG = \sum_{i=1}^{|REL_p|} \frac{rel_i}{log_2(i+1)} \,. \tag{20}$$

Definition 11. ***Normalized Discounted Cumulative Gain (NDCG@K)*** *is computed as the proportion between real Discounted Cumulative Gain (DCG@K) and Ideal Discounted Cumulative Gain (IDCG@K):*

$$NDCG@K = \frac{DCG@K}{IDCG@K} \,. \tag{21}$$

Definition 12. ***Mean Reciprocal Rank (MRR)*** *calculates the reciprocal of the rank at which the first relevant document was retrieved. With $|Q|$ defined as the total number of queries (or recommended rankings) and $rank_i$ representing the position of the top target in the result set for the i-th query:*

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \,. \tag{22}$$

Having defined recommender performance metrics, let us proceed to a review of recent, successful recommender models.

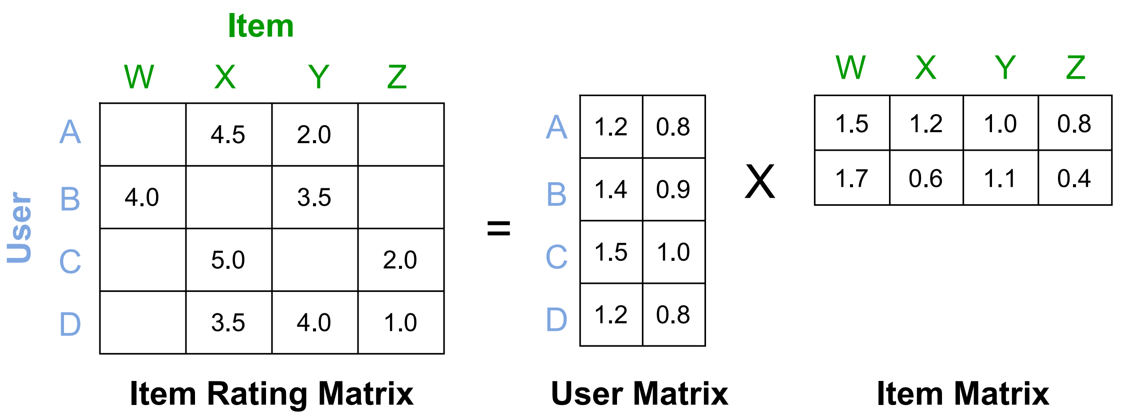
3.4 Collaborative Recommendation

Collaborative recommendation approaches the problem of modeling past user interactions with the products. Based on this, the aim is to foresee which products are likely going to receive some interaction from the user in the future. The prediction is made based on interactions of many users. The core idea is that a user will usually like the items which were favored by similar users in the past. The following sections introduce popular algorithmic approaches towards this problem.

3.4.1 Matrix Factorization Algorithms

Matrix Factorization (MF) methods belong to the earliest approaches in the history of collaborative recommendation. These algorithms operate on user-item interaction matrices, which express the historic behaviors of users (Figure 9). The matrix colums reflect all available items, while the rows express the users. The values within the matrix reflect some past relationship between a user and an item. This can be a purchase (a binary 1/0 value) or a rating score on some scale.

Matrix Factorization works by decomposing the user-item interaction matrix into the product of two lower-dimensionality matrices. The obtained matrices can be understood to contain feature vectors (also called *embedding vectors*) of users and items, which can be used to easily compute similarity between pairs of entities.



	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

W	X	Y	Z
1.5	1.2	1.0	0.8
1.7	0.6	1.1	0.4

Figure 9: Matrix Factorization used to split a user-item rating matrix into user representation matrix and item representation matrix.

Following the definition from Koren et al. [2009], let $p_i \in R^k$ denote the vector which contains the latent features of item (or, product) i, and let $u_j \in R^k$ denote the vector containing the latent features of user j. The latent features of items can be understood as various more or less abstract properties of items, which are expressed in a numeric form. Each characteristic corresponds to a set place in the feature vector, although it is usually hard to tell what human-understandable property is stored in which place. The feature vectors of users follow a similar logic - they characterize users in a more concise, numeric form. Intuitively, they can also represent user's level of interest in the corresponding characteristic in the item feature vector. Under such formulation, the dot product between a user and item vector $p_i^T u_j$ represents a total interest the user j has for the item i. A higher score can mean that the user is more likely to be interested in the item.

However, how exactly to obtain these feature vectors is an open question. A classic MF approach introduced in a blogpost describing the winning entry to the famous Netflix 2006

Prize contest[10] aims to factorize the user-item interaction matrix using a simplified version of Singular Value Decomposition (SVD) [Stewart, 1993]. A more advanced example is the SVD++ model (also proposed at Netflix Prize 2006 contest) which allows to incorporate other modalities, such as product ratings by users or item bookmarking. A wide array of publications inspired by variations of matrix factorization followed [Jiao et al., 2019, Shi et al., 2020, Xian et al., 2017].

However, matrix factorization methods have been found to possess certain drawbacks. One of the most serious of these is that they must be computed in full on the whole dataset whenever the input data changes (e.g. a new user arrives). Another serious issue is the cost of calculating matrix factorization on large matrices, which in practice renders these models inapplicable to modern amounts of data.

3.4.2 Model-Based Collaborative Recommendation

The drawbacks of Matrix Factorization algorithms have been addressed with the introduction of model-based recommendation approaches. Such algorithms usually rely on training with an objective represented by a loss function. They are usually much faster and more efficient to train, and do not have to be recomputed fully with each change of data. The most popular recent approaches involve neural networks and I will focus on them here.

The models which dominate the landscape of today's recommender research are recurrent neural networks (RNNs), graph neural networks (GNNs), and the mechanism of attention, which are suitable for data with temporal nature.

A classic approach, the Gru4Rec model [Hidasi and Karatzoglou, 2018] uses the gated recurrent unit (GRU) in a sequential setting (Figure 10). It feeds sequences of items which appear in user histories to a recurrent model, in a way similar to the processing of sequences of words in RNNs from the natural language processing area. The model returns scores for complete item sets, training with cross-entropy loss. However, the large item space proves to be a problem due to slow computation of scores for 100,000 - 1,000,000 items for each training example. To solve this problem, negative samples are limited in number and drawn from currently batched data (the part of data which is currently loaded onto the GPU).

NARM (Neural Attentive Recommendation Machine) [Li et al., 2017] builds on Gru4Rec adding an item-level attention mechanism. Attention models have been proven to surpass non-attentive RNNs, as attention offers a more fine-grained item similarity representation. NARM

[10] `https://sifter.org/~simon/journal/20061211.html`

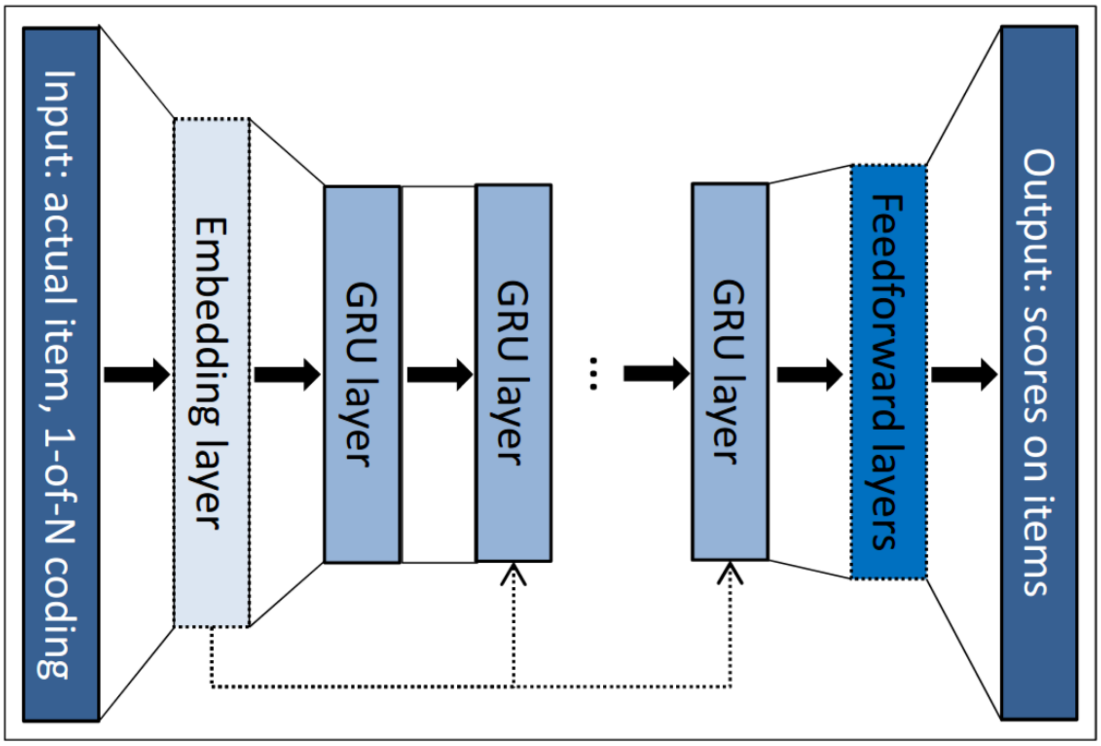


Figure 10: Gru4Rec model. Source: Hidasi et al. [2016].

features 2 GRU networks, for representing both local and global interest in user shopping sessions. These networks are denoted as local and global encoders. The representations computed by the local encoder are analyzed by an attention module, which is able to combine selected inputs deemed important for prediction, to provide very detailed analysis of the input sequence. The combined outputs of local attention and global representations input vectors are then used to compute a unified representation of next likely item c_t. Output ranking scores of items are returned after similarity computation of c_t and all item embeddings. Many variations based on this approach exist, such as STAMP (Short-Term Attention/Memory Priority model) [Liu et al., 2018] which is an attention-based non-RNN model using neural memory modules to represent item long-term and short-term sequences.

A very important class of models are models based on graph neural networks (GNNs). SR-GNN [Wu et al., 2019b] (shown in Figure 11) shape interaction data in the form of a graph, to which a graph neural network is applied. User shopping sessions build a graph of connections between nodes, which represent items. The purchase of the same items in various shopping sessions means that the sessions will become interconnected. This way, the model is able to exploit additional paths between sessions, which were not possible to analyze by sequential-

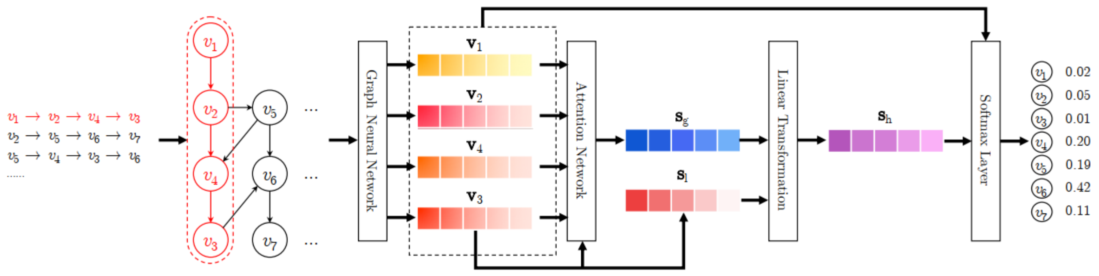


Figure 11: SR-GNN model. Source: Wu et al. [2019b].

based RNNs. This is because it links together item sequences which repeat within different shopping sessions. Item representations computed by the GNN are then fed to an attention model and used to compute an aggregate session representation s_h. Then a multiplication against the item embedding matrix returns ranking scores for all items.

Many approaches inspired by GNNs follow, such as the TAGNN [Yu et al., 2020]. It adds a target-aware attention module to the GNN-based model. This way, sessions are not squashed into a single vector s_h but are computed in variants based on a currently considered possible target item.

GNN-based approaches are able to outperform previously described RNN-based models. However, a disadvantage of the GNN-based models is that they do not scale to growing data volumes easily and thus usually must be evaluated on small datasets (e.g. TAGNN [Yu et al., 2020] is evaluated on just 1/64 of the large-scale RSC15 ACM RecSys 2015 Challenge dataset). On the other hand, non-GNN neural models are usually more scalable, but they can suffer from lower performance.

Finally, some more exotic models integrate ideas from seemingly far-away neural network areas, such as convolutional networks used in image processing. For example, NextItNet [Yuan et al., 2018] is based on stacked 1D dilated convolutions, to replace an RNN or GNN.

3.5 Content-Based Recommendation

Content-based recommendation systems make predictions without taking the interaction data into account, thus using item similarity as chief decision criterion. They are applied for example in cases where collaborative recommendation cannot be used due to lack of recorded historic data. When an item has just been introduced into the inventory and no customer has bought it yet, the only available data are the item characteristics themselves. Content-based recommendation can also be used in specific use-case scenarios, e.g. when an e-commerce wants to enable similar product recommendation. Such cases happen for example in fashion stores, where a user can upload a photo of a person wearing the desired piece of clothing, and a shop is expected to suggest a similar clothing from its item storage. Under this formulation, content-based recommendation is akin to item retrieval (search) based on various modalities:

1. **Uni-modal:** A single modality on both input and output, e.g. recommendation based on uploaded an image which is matched against a database of images.
2. **Cross-modal:** Various modalities on input and output, e.g. recommendation based on a textual description which is matched against a database of images.

Most existing uni-modal instance retrieval solutions train a deep learning model to obtain vector representations of images in such a way that examples from the same class should be close to each other [Jun et al., 2019, Wieczorek et al., 2020, Luo et al., 2020, Liu et al., 2016b, Yuan et al., 2020]. At the retrieval stage, the query embedding is scored against all gallery embeddings and the most similar ones are returned. Currently most works use comparative/ranking losses and the Triplet Loss is one of the most widely used approaches, often combined with auxiliary losses such as classification or center loss [Wieczorek et al., 2020, Luo et al., 2020, Wen et al., 2016b]. Triplet Loss is a loss formula which aims to simultaneously minimize distance between a given example representation (called *anchor*) and a positive input, and maximize the distance from the anchor to the negative input is maximized. A positive input can be for example a photo of the same dress in a different pose, and a negative input can be a photo of a completely different dress. Formally, it is expressed by the following formula:

$$\mathcal{L}_{triplet} = \left[\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha\right]_+ . \tag{23}$$

Here, A is the anchor example, P is the positive example, N is the negative example, and α is a margin parameter to control the magnitude of required difference between positive and negative examples.

Unfortunately, these models must run each query example separately against a full item gallery which is often of considerable size, which adds a big overhead to their operation in a production environment.

On the other hand, cross-modal instance retrieval models often use cross-modal attention in order to search for matching picture/text fragments. Cross-modal attention works just like the regular attention described in the case of previous models, but it is computed on vectors stemming from various domains (e.g. text and image vector representations). Chen et al. [2020a] proposed to use Iterative Matching with Recurrent Attention Memory (IMRAM) method to capture the mutual image and text alignments with cross-modal attention. Wang et al. [2020a] introduce Scene Graph Matching (SGM) - an approach using separate graphs for each modality to capture object features and their mutual relationships in corresponding modality. When the object-level and relationship-level features are extracted from both modalities, the final matching on both levels takes place. Stacked Cross Attention Network (SCAN) [Lee et al., 2018] can be defined as two complimentary formulations, where each modality is used as a context to each other and two separate similarity scores are inferred with the use of cross-attention. SGRAF [Diao et al., 2021] builds upon SCAN, extends it with self-attention and applies it to each modality separately. All these models, although effective, become inefficient in a real-life production setting due to the use of inherently slow attention mechanism.

3.6 Interpretability and Robustness of Recommenders

Model interpretability is a research area which aims to create ways of understanding model decisions and is currently developed within the Explainable Artificial Intelligence (XAI) science branch. Interpretability can be roughly categorized into universal methods which work for any kind of model (e.g. LIME [Ribeiro et al., 2016] which uses local models on many modifications of model input values, or SHAP [Lundberg and Lee, 2017] from cooperative game theory), and gradient attribution methods designed specifically for neural networks, which are often based on running an extra backward pass through the network and verifying which inputs had the biggest influence in terms of the produced gradients [Qi et al., 2019, Sundararajan et al., 2017]. Model interpretability is important in ascertaining whether a given model takes into account valid inputs as a basis for its decisions - so, whether it is reliable and can be trusted in a production setting.

Model robustness is another aspect of model reliability. Robustness expresses a model's invulnerability to adversarial examples. Adversarial examples are inputs to machine learning models that cause failures in models although they do not cause any problems for humans. An example of adversarial examples can be popular linguistic errors: in two descriptions *"blue round flower vase, 35 cm height"* and *"bleu round flower vase, 35 cm height"* the misspelling in *bleu* can make a model return a wrong answer, although it reacts to the first text correctly. Adversarial examples can either originate naturally or maliciously: an attacker can intentionally design them to cause the model to make a mistake. Adversarial examples are broadly studied in literature [Goodfellow et al., 2015, Athalye et al., 2018, Hendrycks et al., 2021].

3.7 Multimodal Embedders for Recommenders

The quality of a multimodal recommender is dependent on both the actual model making recommendations (be it collaborative or content-based), but also on auxiliary models which stand before them and produce input representations, ingested by recommendation models.

With the introduction of embedding models such as GloVE [Pennington et al., 2014] or BERT [Devlin et al., 2019] for text, or many node embedding approaches for graphs [Shi et al., 2018, Wang et al., 2018], embedding vectors have gained popularity as the optimal way to represent object properties in a compressed way. Embedding vectors have been found to be very convenient input for downstream models. In practice, many recommender and information retrieval systems exploit embeddings from upstream models as an input data abstraction (see Figure 8) [Jeong et al., 2020, Hassan et al., 2019, Diao et al., 2021]. Because such recommender models do not have access to the data itself but only to its representation learned by the embedders, it is crucial to ensure the highest possible quality of embedders themselves. In light of this, the embedders should be subject to the same kind of scrutiny we apply to the recommenders themselves, regarding their performance, scalability, reliability and interpretability obtained with XAI methods.

3.8 Datasets

Multiple public datasets are actively used in the field of recommenders. Some of these datasets are dedicated to a particular recommendation subtask, such as session-based or k-top recommendation. Some are quite versatile and can be used for multiple tasks. In this section, datasets used in experiments from this thesis are listed and described.

3.8.1 SNAP graph datasets

Many of the most popular recommender/graph learning datasets are hosted by the Stanford Large Network Dataset Collection [Leskovec and Krevl, 2014] [11]. Their advantages are accessibility, popularity in literature and the fact that they represent various underlying phenomena (various

[11] `https://snap.stanford.edu/data/`

social networks, citation networks, co-purchasing networks, road networks).

- **Facebook Dataset** [Rozemberczki et al., 2021] is a subgraph of the real-life Facebook social network, containing connections between verified Facebook sites. Nodes represent official Facebook pages while the links are mutual likes between sites. The dataset contains 22,470 nodes and 171,002 edges. Nodes belong to one of 4 categories defined by Facebook.

- **YouTube Dataset** [Yang and Leskovec, 2012] was sourced from the Youtube video hosting site. It contains the graph of friendship relations between users as well as user-defined groups. Youtube dataset has 1,134,890 nodes and 2,987,624 edges.

- **RoadNet Dataset** [Mahoney et al., 2009]] is road network of California. Intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges. The dataset contains 1,965,206 nodes and 2,766,607 edges.

- **LiveJournal Dataset** [Backstrom et al., 2006] is a graph dataset sourced from a free online community site LiveJournal with almost 10 million members. LiveJournal allows members to maintain journals, individual and group blogs, and it allows people to declare which other members are their friends they belong. The dataset contains 4,847,571 nodes and 68,993,773 edges.

- **Twitter Dataset** [Kwak et al., 2010] is a large graph of follower relationships from a snapshot of Twitter network in 2010. The dataset contains 41,652,230 nodes and 1,468,364,884 edges. This is an exceptionally massive graph which can be processed by a very limited set of algorithms. However, this graph size can be judged a realistic one, with many real-life graphs being of similar or even bigger sizes. Other mentioned graphs can be considered toy datasets, dedicated to various experimental algorithms, which do not reach adequate speed and scalability requirements to be used in practical problems.

3.8.2 Smaller graph datasets

The following small datasets are mainly used in Graph Neural Network research. They are well grounded in this area, having been used for years. However, their sizes do not correspond to most real-life problems. They mainly serve as benchmarks for comparison with other methods, and can be processed by even the least scalable GNNs. Due to their prevalence, they can be considered a mandatory benchmark set for GNN-based methods.

- **Cora** [McCallum et al., 2000] is a graph consisting of 2,708 nodes (scientific publications) classified into one of seven classes. The citation network consists of 5,429 edges.
- **CiteSeer** Giles et al. [1998] is a graph consisting of 3,312 nodes (also scientific publications) classified into one of six classes. The citation network consists of 4,732 edges.
- **PubMed** Sen et al. [2008] is a graph consisting of 19,717 nodes (scientific publications from PubMed database referring to diabetes) classified into one of three classes. The citation network consists of 44,338 edges.

3.8.3 Top-k Recommendation Datasets

Datasets described in this section are of substantial size and are especially suited to top-k recommendation. They record interaction data of users with movie content providers and store the details about the movies and users, together with ratings that the movies received by users.

- **Netflix Prize** [Bennett and Lanning, 2007]. The dataset contains movie ratings from viewers. It consists of about 100,000,000 ratings for 17,770 movies given by 480,189 users. Each rating in the training dataset consists of four entries: user, movie, date of grade, grade. Users and movies are represented with integer IDs, while ratings range from 1 to 5.
- **MovieLens20M**. Similarly to Netflix Prize dataset, this is a movie rating dataset with 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Additionally, it includes tag data with 12 million relevance scores across 1,100 tags.

3.8.4 Session-based Recommendation datasets

Datasets described in this section are big and taken from real-life e-commerces and content providers. They are especially suited to session-based recommendation, because they record user sessions and order of items purchased. Some of these datasets include additional multi-modal data, such as item properties.

- **RETAIL** dataset has been collected from a real-world e-commerce website RetailRocket. It contains 212,182 actions, in 59,962 sessions and with 31,968 items. The dataset consists of behaviour data and item properties such as price, category, vendor, product type and others. All values are hashed and each property is mapped into unique identifier.

- **DIGI** is e-commerce dataset shared by the company Diginetica. It contains 264 thousand actions, in 55 thousand sessions, and 32 thousand individual items. It contains additional multimodal data such as product name, textual tokens and items returned by search queries.

- **RSC15** is an e-commerce dataset used in the 2015 ACM RecSys Challenge. It was created by the company Yoochoose and contains a collection of shopping sessions, where each session contains the click events that the user performed in the session. For some of the sessions, there are also buy events. The dataset includes 5.43 million actions, 1.4 million sessions, and 28,582 items.

- **30Music** [Turrin et al., 2015] dataset consists of music listening logs obtained from Last.fm. It includes 638,933 events, 37,333 sessions, with 210,633 unique songs. Playlists created by the users form additional multimodal data.

- **AOTM** [McFee and Lanckriet, 2012] is a dataset from the website Art of the Mix. It includes playlists of songs, with 306,830 events, 21,888 playlists, and 91,166 unique songs. This dataset does not include multimodal data with matching item identifiers.

- **NOWP** [Zangerle et al., 2014] dataset features context- and content features of song listening events. It contains 271,177 music listening events of 27,005 sessions and 75,169 thousand songs collected from Twitter. This dataset does not include multimodal data with matching item identifiers.

3.8.5 Uni-Modal Content Recommendation Datasets

These datasets are composed of images and the aim is to find matching photos, e.g. demonstrating the same product from various angles.

- **DeepFashion** [Liu et al., 2016b] is a dataset which contains over 800,000 images, spread across several fashion related tasks. In particular, it includes *Consumer-to-shop Clothes Retrieval* subset that contains 33,881 unique clothing products and 239,557 images.

- **Street2Shop** [Kiapour et al., 2015] is one of the first modern large-scale fashion dataset. The dataset contains over 400,000 shop photos and 20,357 street photos. In total there are 204,795 distinct clothing items in the dataset.

3.8.6 Cross-modal Content Recommendation Datasets

These datasets serve for matching content representend with various modalities. One of the modalities is text, and the other is image. The aim is to find product/scene photos based on textual description, and the other way around.

- **MSCOCO** [Lin et al., 2014] dataset is a large-scale object detection, segmentation, and captioning dataset. It contains 123,287 photos, mainly from daily life scenes. Each image is annotated with 5 captions produced by crowdworkers from Amazon Mechanical Turk.
- **Flickr30K** [Plummer et al., 2015] dataset contains 31,783 images with 5 corresponding captions each. Generally, the Flickr30k captions are much longer and are in many cases more detailed than in MSCOCO [Guan et al., 2018].

4 A Robust, Multimodal Recommender System

In this section I describe in detail the robust, multimodal recommender system created in terms of this thesis.

The system is designed to achieve two main objectives: 1) high performance thanks to novel algorithms, 2) speed and scalability, to be applicable to real-life datasets. Figure 12 shows the multi-purpose recommender elements created in terms of this thesis together with the names of the models (in green boxes).

4.1 Fast Embedding Learning for a Recommendation Model

This section covers Hypothesis 1 from Section 1.2.
The publication on which this section is based: Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, Jacek Dąbrowski and Przemysław Biecek. ”Cleora: A Simple, Strong and Scalable Graph Embedding Scheme.” *Proceedings of the 28th International Conference on Neural Information Processing (ICONIP 2021)*, 140 MNiSW points.

4.1.1 Introduction

As noted in Sections 3.4 and 3.7, modern recommenders usually require *embeddings* - a form of a numeric representation of users and items, which could hold information on the similarity between encoded users and items. Such representations are usually ingested as model inputs by neural networks instead of larger and more inefficient structures such as user-item rating matrices (see Section 3.4.1 for an example of such a matrix). Embeddings can be computed for many types of data, such as text and image, using dedicated self-supervised or unsupervised models [Mikolov et al., 2013, Devlin et al., 2019]. In terms of recommenders, it is particularly common to compute embeddings of entities (users or items) based on event data generated from user actions (page views, purchases, and similar). It has been discovered that such behavioral data is especially well represented in the form of graphs, so graph embeddings are actively researched for this purpose.

Graphs are ubiquitous structures in recommendation systems. Customers and items are often modeled as graph nodes, while their interactions are expressed in the form of graph edges. The existence of an edge usually denotes interaction, while no edge denotes no interaction. As such, graph processing is a core step for current collaborative recommenders. Embeddings are usually

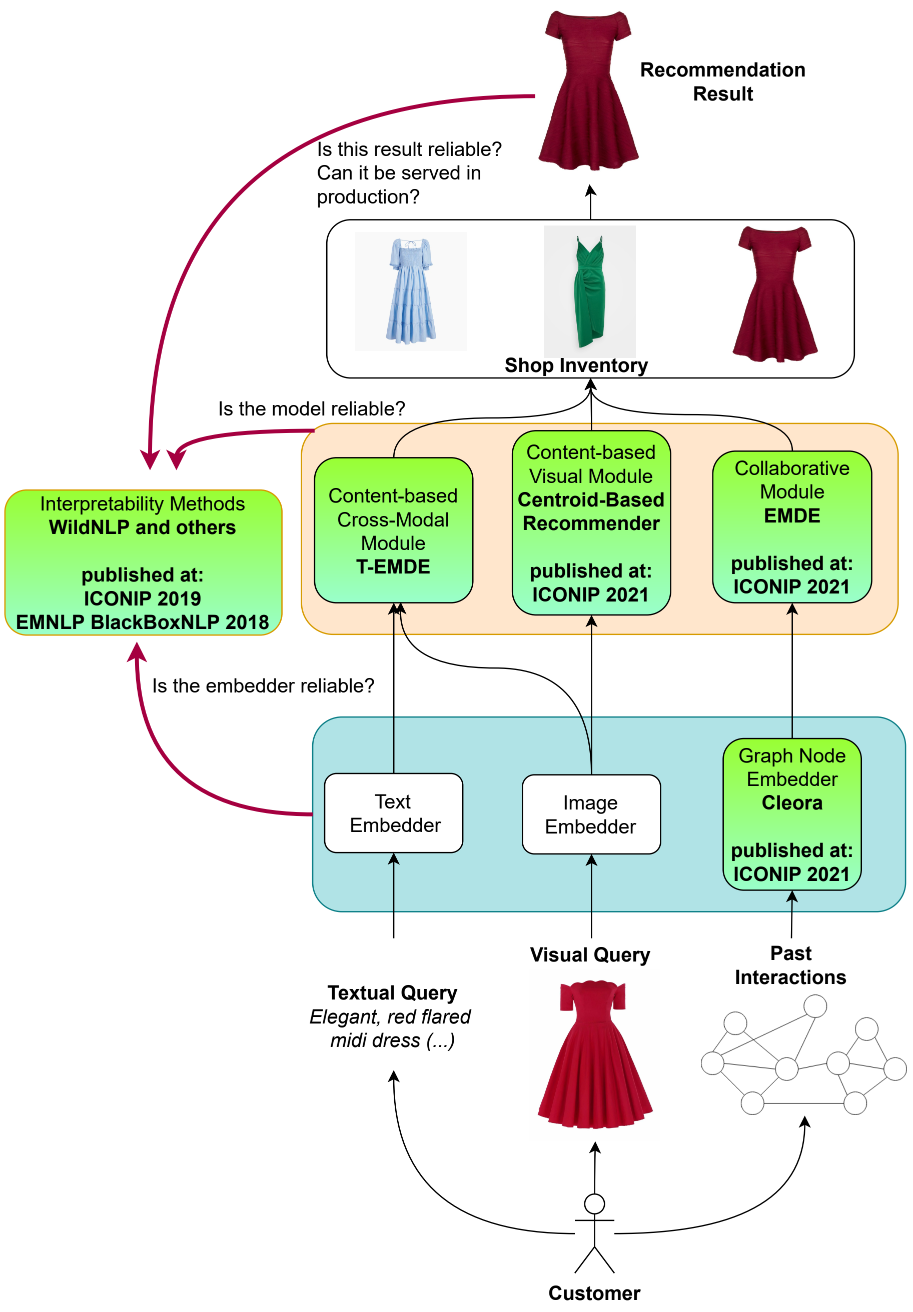

Recommendation Result
Is this result reliable? Can it be served in production?
Shop Inventory
Is the model reliable?
Content-based Cross-Modal Module T-EMDE
Content-based Visual Module Centroid-Based Recommender published at: ICONIP 2021
Collaborative Module EMDE published at: ICONIP 2021
Interpretability Methods WildNLP and others published at: ICONIP 2019 EMNLP BlackBoxNLP 2018
Is the embedder reliable?
Text Embedder
Image Embedder
Graph Node Embedder Cleora published at: ICONIP 2021
Visual Query
Past Interactions
Textual Query
Elegant, red flared midi dress (...)
Customer

Figure 12: A multi-purpose recommendation system. The parts in green boxes have been created in terms of this thesis. The modules have been presented and published in the proceedings of the following venues: **Graph Node Embedder Cleora** (ICONIP 2021, 140 MNiSW points) [Rychalska et al., 2021a], **Collaborative Module EMDE** (ICONIP 2021, 140 MNISW points) [Dąbrowski et al., 2021], Content-based Visual Module Centroid-Based Recommender (ICONIP 2021, 140 MNISW points) [Wieczorek et al., 2021], **WildNLP** (ICONIP 2019, 140 MNiSW points) [Rychalska et al., 2019], **other interpretability methods** – Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP [Rychalska et al., 2018], **Content-based Cross-Modal Module T-EMDE** – unpublished.

computed for individual graph nodes, exploiting various topological properties of the interaction graphs. The distance between two embeddings expresses node similarity as learned by the embedding algorithm. High node similarity is usually understood as a mix of various properties which are shared for both nodes. This could be similar patterns of connectivity (nodes which have edges with the same or similar nodes), or similar node degree (whether both nodes are hubs with large number of edges, or fairly unconnected nodes), or node type (item category, color, brand, or any other property) [Tang et al., 2015].

4.1.2 Background

Unfortunately, most current embedding systems are not scalable to large graphs (millions of nodes, billions of edges), while such graph sizes are common in real-life recommendation settings. Most recent high-quality approaches are based on Graph Neural Networks (GNNs), and in particular Graph Convolutional Neural Networks (GCNs) (see Section 2.5). GNNs are complex neural networks with known scalability issues, such as neighborhood explosion [Bai et al., 2022]. This phenomenon is caused by the fact that each node's neighborhood must be recursively loaded into memory in each step of the computation. With rich neighborhoods, this can scale to arbitrarily large memory footprints. The methods which are truly scalable, usually are not based on GCNs and utilize heavy paralellism, which makes them very complex and thus hard to understand and debug, or require specialized and very expensive resources such as GPU clusters [Lerer et al., 2019, Akyildiz et al., 2020].

However, the structure of GCNs warrants an alternative view, which could transform them into a faster algorithm. As previously noted, the classic GCN update rule [Kipf and Welling, 2017] looks like the following:

$$H^{[i+1]} = \sigma(W^{[i]} H^{[i]} A^*)\,. \tag{24}$$

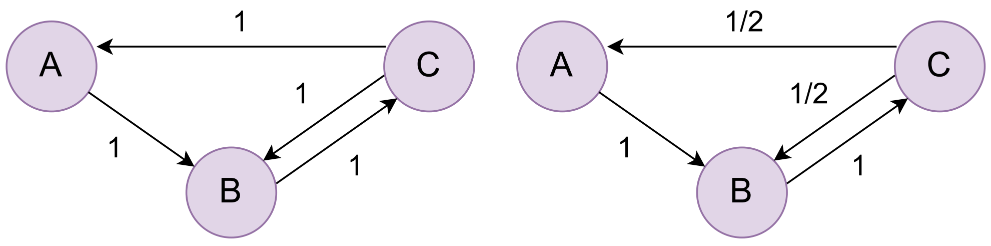
Here, $H^{[i]}$ and $H^{[i+1]}$ denote matrics containing hidden node representations from layer i and $i+1$, respectively. $W[i]$ is the trainable weight matrix from layer i, A^* is the normalized adjacency matrix of the graph, and σ is the network nonlinearity function. In order to train the network, weights W need to be optimized with backpropagation. However, my work shows that the complete removal of weights W and nonlinearity σ leads to a still viable (although non-neural) algorithm which is devoid of most scalablity issues of GCNs. This new, simplified approach can be understood as a form of weighted avearging of neighbor embedding vectors. As such, each node is represented with a weighted average of its neighbors. In my work I prove the hypothesis that such transformation of the GCN algorithm results in a highly efficient algorithm which at the same time produces high quality node embeddings.

4.1.3 Preliminaries

Formally, we can define the graph and node embeddings with the following terms:

Graphs. A graph G is a pair (V, E) where V denotes a set of nodes (also called vertices) and $E \subseteq (V \times V)$ as the set of edges connecting the nodes.

Node Embedding. A node embedding of a graph $G = (V, E)$ is a $|V| \times d$ matrix T, where d is the dimension of the embedding. The i-th row of matrix T (denoted as $T_{i,*}$) corresponds to a node $i \in V$ and each value $j \in \{1, ..., d\}$ in the vector $T_{i,j}$ captures a different feature of node i. Node embeddings are used to express node properties in a joint, compressed form.



0	1	0
0	0	1
1	1	0

Adjacency Matrix

0	1	0
0	0	1
1/2	1/2	0

Transition Matrix

Figure 13: Adjacency matrix and random walk transition matrix.

We also need a definition of the Random Walk Transition Matrix, which is a variation of neigborhood-representing matrix:

Random Walk Transition Matrix. The Random Walk Transition Matrix is akin to the adjacency matrix, in that it holds information about graph connectivity. It has a dimensionality of $|V| \times |V|$, and each nonzero value v_{ij} within the matrix means that there exists an edge between nodes i and j. Each nonzero value within the matrix corresponds to the probability of choosing an exact path during a random walk, also taking into account that edges might have weights attached. For example, if node A has an edge to node B with weight 3, and to node C with weight 1, v_{AB} will have a value of 3/4, and v_{AC} will have a value of 1/4. This is illustrated in Figure 13.

4.1.4 Proposed Algorithm

Drawing from annotations introduced in the previous section, the Cleora algorithm can now be formally introduced. Cleora relies on iterative multiplications of the embedding matrix T_i and the random walk transition matrix M. Each multiplication outputs a new embedding matrix T_{i+1}. Intuitively, this results in each node's embeddings being gradually more and more similar to its nodes' embeddings, weighted by their strength of connection. The embeddings are normalized between iterations, in order to prevent value collapse.

Due to the absence of any trainable weights, Cleora is a non-neural algorithm. As a consequence, the necessity of computing forward and backward passes on large node neighborhoods does not exist and the direct cause of GCN inefficiency can be avoided.

Formally, as stated in Rychalska et al. [2021a]: Given a graph $G = (V, E)$, we define the random walk transition matrix $M \in \mathcal{R}^{V \times V}$ where $M_{ab} = \frac{e_{ab}}{deg(v_a)}$ for $ab \in E$, where e_{ab} is the number of edges running from node a to b, and $deg(v_a)$ is the degree of node a. For (a, b) pairs which do not exist in the graph, we set $e_{ab} = 0$. For (a, b) pairs which exist in the graph, we set $e_{ab} = 1$. Optionally we can scale the scores by external edge weights. The embedding matrix is first initialized randomly as follows: $T_0 \in R^{|V| \times d} \sim U(-1, 1)$, where d is the embedding dimensionality. Then for $i \in \{1, ..., n\}$ iterations, we multiply matrix M and T_{i-1}, and normalize rows to keep constant L_2 norm. T_n is our final embedding matrix. The algorithm is shown in Figure 14.

Simultaneously with first works on Cleora, there appeared research papers confirming the initial idea that GCNs could indeed be over-parametrized. As noted in Wu et al. [2019a], GCNs can be stripped down to a repeated AH^i multiplication with the same (or even better) performance as the base network. As such, Cleora can be interpreted as a form of a GCN which is stripped down to its most effective core operation. In Cleora, the adjacency matrix A is replaced with the random walk transition matrix to take into account edge weights, representing for example the number of times certain items were bought together. Node representations from the current i-th layer H^i are replaced with the embedding matrix from the previous Cleora iteration.

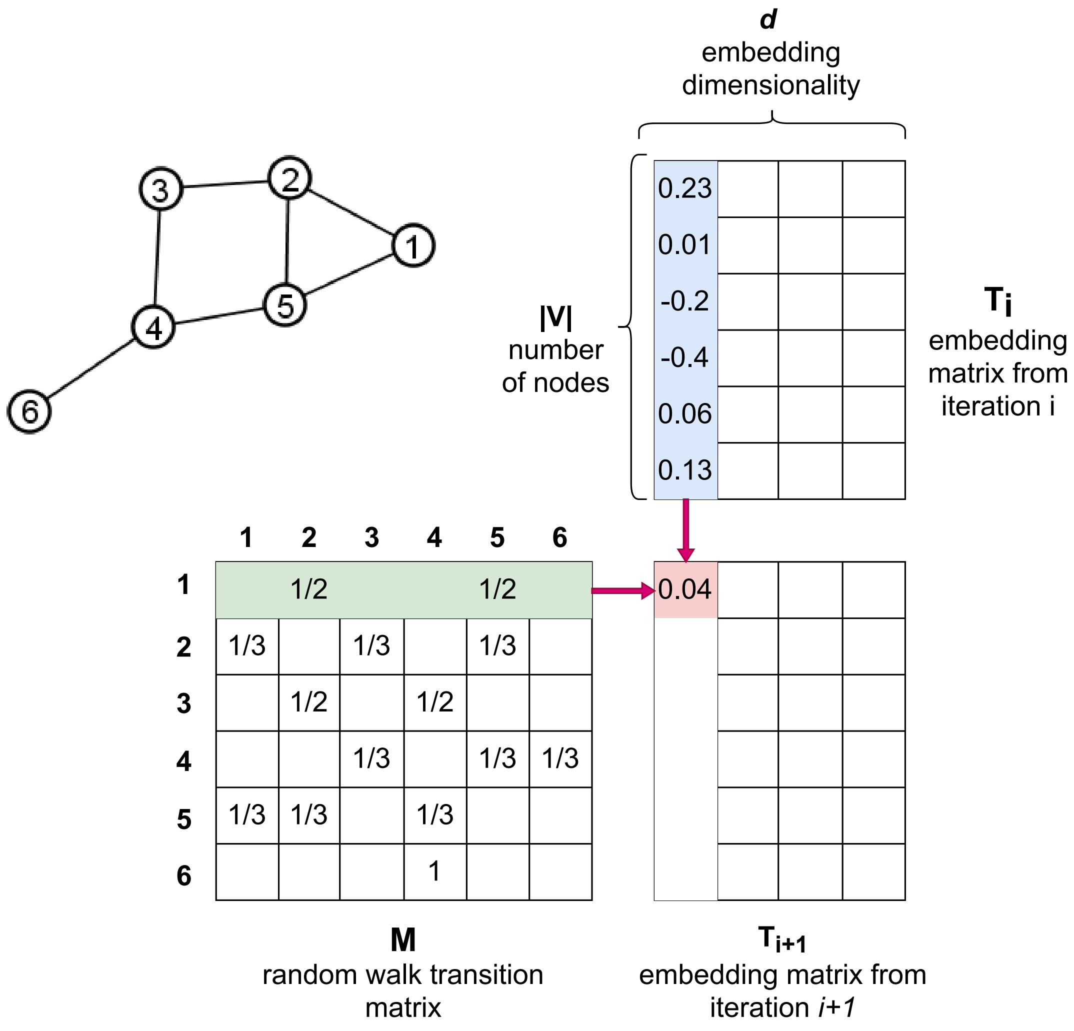


Figure 14: The Cleora algorithm. The M matrix is the random walk transition matrix, representing the proximity of nodes and their strength of connection. The T_0 matrix represents node embeddings and is first initialized at random. Then a repeated multiplication against the M matrix follows. After n iterations, the matrix T_n holds the desired node embeddings.

Cleora algorithm has a memory complexity of $O(|V|(1 + 2d) + 2|E|)$. The most computationally heavy operation is the multiplication of the sparse matrix M and dense matrix T, which sets the computational complexity at $O(|V| \cdot average\_node\_degree \cdot d)$.

4.1.5 Experimental Procedure and Results

Competitors and Datasets. Cleora is evaluated on both small and large graph datasets, including popular datasets from the SNAP Stanford group collection (Section 3.8.1) and datasets traditionally used to evaluate GNNs (Section 3.8.2). The datasets are summarized in Tables 3 and 4.

Cleora is evaluated together with various popular graph embedding algorithms and libraries, which are divided into two groups:

1. Approaches optimized for speed, such as PyTorch BigGraph [Lerer et al., 2019] or GOSH [Akyildiz et al., 2020]. These libraries are not based on GCNs, instead offering fast implementations of older algorithms (such as simple feed-forward neural networks) amenable to scalable multithreaded or distributed processing. They implement, among others:

   - DistMult [Yang et al., 2015] - in this approach embeddings are learned via a neural network. Nodes are represented as vectors, and the edges as relations (mapping functions). The first network layer projects a pair of input entities to low dimensional vectors, and the second layer combines these two vectors to a scalar for comparison via a scoring function with relation-specific parameters.
   - VERSE [Tsitsulin et al., 2018] - a single-layer neural network trained to preserve the distribution of a selected node-to-node similarity measure. It offers a number of similarity measures to choose from, e.g. personalized PageRank similarity.

   Due to their remarkable scalability, these libraries are evaluated on large datasets from Table 3.

2. GCN-based approaches. These algorithms are not implemented in terms of fast, optimized libraries and are significantly less scalable than Cleora. However, I take them into account in order to check whether the GCN simplification approach in Cleora has an impact on final embedding quality. GCNs are evaluated on datasets from Table 4.

Name	**Facebook**	**YouTube**	**RoadNet**	**LiveJournal**	**Twitter**
Number of Nodes	22,470	1,134,890	1,965,206	4,847,571	41,652,230
Number of Edges	171,002	2,987,624	2,766,607	68,993,773	1,468,365,182
Average Degree	12	5	3	16	36
Directed	No	No	No	Yes	Yes

Table 3: Large datasets characteristics.

Name	**Cora**	**CiteSeer**	**PubMed**
Number of Nodes	2,708	3,327	19,717
Number of Edges	5,429	4,732	44,338
Directed	No	No	No

Table 4: Small datasets characteristics.

Evaluation Methods and Metrics. Evaluation is conducted on popular target tasks: link prediction and node classification.

- **Link Prediction** - it is one of the most common proxy tasks for embedding quality evaluation [Liben-Nowell and Kleinberg, 2007]. It consists in determining whether two nodes should be connected with an edge or not. Link prediction considers the positive pairs (real edges found in the graph), and negative pairs - non-existent edges generated by taking a node from the positive pair and assigning it to a node drawn with random sampling. Technically, the negative pairs are created in the same way as in the Pytorch BigGraph paper [Lerer et al., 2019]: 10,000 most popular graph nodes are selected, which are then paired with the start node of each positive pair to create negative examples. The final performance metrics are the Mean Reciprocal Rank (MRR) and Hit Rate (HR) (see Section 3.3) which consider the position of the real pair among the 10,001 total examples.

  For each pair, the Hadamard product is computed between their embeddings, which is a standard procedure observed to work well for performance comparison [Grover and Leskovec, 2016, Dalmia et al., 2018]. The obtained Hadamard products are fed to a simple logistic regression classifier which predicts the existence of edges.

- **Node Classification.** Some graph datasets have classes assigned to each node (for example, denoting some interest groups a user belongs to in social network graphs). In order to predict these classes, a logistic regression classifier is trained by feeding node embeddings as inputs and predicting the node class. A usual metric used in node embedding literature for node classification is simple Accuracy, so I follow this practice.

Algorithm	**Facebook**	**YouTube**	**RoadNet**	**LiveJournal**	**Twitter**
Total embedding time (hh:mm:ss)					
Cleora	00:00:43	00:12:07	00:24:15	01:35:40	25:34:18
PBG	00:04.33	00:54:35	00:37:41	10:38:03	-*
Deepwalk	00:36:51	28:33:52	53:29:13	*timeout*	*timeout*
Training time (hh:mm:ss)					
Cleora	00:00:25	00:11:46	00:04:14	01:31:42	17:14:01
PBG	00:02:03	00:24:15	00:31:11	07:10:00	-*

Table 5: Calculation times baseline methods. `Total embedding time` encompasses the whole training procedure, including data loading and preprocessing. `Training time` encompasses just the training procedure itself. * - training crashed due to excessive resource consumption. Source: Rychalska et al. [2021a].

	Facebook		**YouTube**		**RoadNet**		**LiveJournal**		**Twitter**	
Algorithm	MRR	HR@10	MRR	HR@10	MRR	HR@10	MRR	HR@10	MRR	HR@10
Scalable methods										
Cleora	0.0724	0.1761	0.0471	0.0618	0.9243	0.9429	0.6079	0.6665	0.0355	0.076
PBG	0.0817*	0.2133*	0.0321*	0.0640*	0.8717*	0.9106*	0.5669*	0.6730*	-**	-**
GOSH	0.0924*	0.2319*	0.0280*	0.0590*	0.8756*	0.8977*	0.2242*	0.4012*	-**	-**
Non-scalable methods										
Deepwalk	0.0803*	0.1451*	0.1045*	0.1805*	0.9626*	0.9715*	*timeout*	*timeout*	*timeout*	*timeout*

Table 6: Link prediction performance results. * - results with statistically significant differences to Cleora according to the Wilcoxon two-sided paired test (p-value lower than 0.05). ** - training crashed due to excessive resource consumption/unscalable architecture. Source: Rychalska et al. [2021a].

Embedding training times on multiple small-to-large datasets ranging from over 170,000 (Facebook dataset) to around 1,500,000,000 edges (Twitter dataset) are displayed in Table 5. PyTorch BigGraph (PBG) was identified as the fastest high-performing competitor method, and

Deepwalk [Perozzi et al., 2014] serves as a baseline example of the non-scalable methods. Cleora is significantly more scalable than PBG in each case. Training is up to 5 times faster than PBG and over 200 times faster than Deepwalk. Moreover, other algorithms have consumed more resources, which resulted in training crashes. The performance results in terms of embedding quality shown in Table 6 suggest that Cleora does not suffer from a disadvantage compared to competitors, and in many cases delivers state-of-the-art results.

Table 7: Classification accuracy - comparison with Graph Convolutional Networks. Source: Rychalska et al. [2021a].

Algorithm	**Cora**	**CiteSeer**	**PubMed**
AS-GCN	0.874	0.797	0.906
Cleora	0.868	0.757	0.802
AdaGCN	0.855	0.762	0.798
N-GCN	0.830	0.722	0.795
GCN	0.815	0.703	0.790

Moreover, I also compare Cleora to Graph Convolutional Network approaches. As baselines I evaluate the following algorithms:

- the original GCN model [Kipf and Welling, 2017],
- N-GCN [Abu-El-Haija et al., 2019]: a marriage of GCN with random walks over the graph - it trains multiple instances of GCNs over node pairs discovered at different distances in random walks,
- AdaGCN [Sun et al., 2021]: a GCN with integrated AdaBoost for exploitation of higher-order neighbors,
- AS-GCN [Huang et al., 2018]: a GCN with adaptive sampling of node neighbors for faster training and variance reduction.

The results in Table 7 show that Cleora gives significantly better results than vanilla GCN and even more advanced approaches such as N-GCN and AdaGCN. This suggests that desipte its simplicity, Cleora does not suffer from significant performance drops.

4.1.6 Summary

The obtained results show that both objectives of the Cleora algorithm have been achieved: it is both faster and at least as good as competitors in terms of embedding quality. As such, Hypothesis 1 from Section 1.2 is proven.

4.2 Probability Density Estimation for a Recommendation Model

This section covers Hypothesis 2 from Section 1.2.

The publication on which this section is based: Jacek Dąbrowski*, Barbara Rychalska*, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Bąbel, Andrzej Michałowski, Adam Jakubowski. ”An Efficient Manifold Density Estimator for All Recommendation Systems.”, *Proceedings of the 28th International Conference on Neural Information Processing (ICONIP 2021)*, 140 MNiSW points. * – equal contribution

4.2.1 Introduction

An increasingly common problem setting in the recommendation domain is that predictions must be made from multi-modal interaction data, with many interaction types, object metadata coming from multiple domains, and having different types of attributes. Interactions often have one or more weight values attached, representing e.g. cardinality, strength, duration, or monetary value. Popular modalities found in e-commerce data are images (photos of products), text (descriptions of products), and graph-based data (various types of behavioral data of users' actions). Real-life data volumes are usually large, comprising millions of users and billions of interactions, as evidenced in Section 4.1 in the discussion of the Cleora algorithm. Moreover, there exist multiple collaborative recommendation-based sub-tasks such as session-based recommendation (where the input is an ordered collection of items based on a single shopping session), or top-k recommendation (where the item collection is an unordered set of items that the user bought or found interesting, usually over a longer timespan). The area of recommender systems is thus a vast and complex field, with many objectives and extreme variety in available data sources.

Having introduced the Cleora node embedding algorithm in Section 4.1, I now proceed to the discussion of a recommender algorithm which could ingest Cleora graph embeddings, along with embeddings for other essential modalities (text, images). I aim to define a new top-k recommendation algorithm, which combines many modalities while offering good scalability and speed.

4.2.2 Background

A central problem of top-k recommendation is how to merge various historic events of a certain user into one representation, which could sum up the user's behavioral profile. Historically, the approaches towards top-k recommendation were often based on matrix factorization [Koren et al., 2009]. These methods have now grown obsolete due to the reliance on costly matrix operations, which scale very poorly to growing numbers of users and items. Nowadays, embeddings have taken over as a basic representation method of various entities and a basic building block of neural systems. However, the use of embeddings gives rise to an important problem called *embedding aggregation*. Given a set of embeddings of items a user has bought in the past, it is unclear how to create a single user embedding out of them. A naive approach of averaging embeddings of all items a user ever bought proves to destroy the internal logic of embeddings and is generally not considered in the literature. The most popular recent approaches of embedding aggregation involve graph neural networks and/or attention, as described in Section 3, however these methods are very slow and rarely allow for easy incorporation of various kinds of embeddings. Thus, the question of how to represent a user with their purchased items is an open research problem.

Interestingly, density estimation of user/item representation spaces could be a way to aggregate items into user behavioral profiles, in a way that would work for all embedding modalities. To understand this idea, we need to remember that the embedding representation space spanned by the embedding vectors is highly multi-dimensional. Its dimensionality is equal to the dimensionality of embeddings themselves, e.g. in Cleora it is often 1024. Based on an adequate density estimation method, it is possible to partition the embedding space into regions. This in turn would allow to represent each embedding with one number only (region identifier) instead of 1024 decimal numbers of the embedding vector. Such density estimation and quantization is modality-independent, meaning that any kind of embedding can be treated this way. Each user could then be represented with a multimodal vector, which holds the count of each region's appearances in user history, without any averaging or counting attention. The whole procedure could be performed at input preprocessing. Thus, it departs from existing approaches performing embedding aggregation within the actual neural network. The elimination of network-based embedding aggregation could largely simplify the neural architecture and speed up computation.

Based on these considerations, I formulate a hypothesis that a multidimensional density estimator can be used as a foundation of a model to produce an efficient and multimodal recommendation system.

To verify the hypothesis, a density estimation method is needed as an entry point. Many density estimation approaches have been created to date [Greengard and Strain, 1991] yet only a small fraction is fast enough to process highly multidimensional input data [Gramacki, 2017].

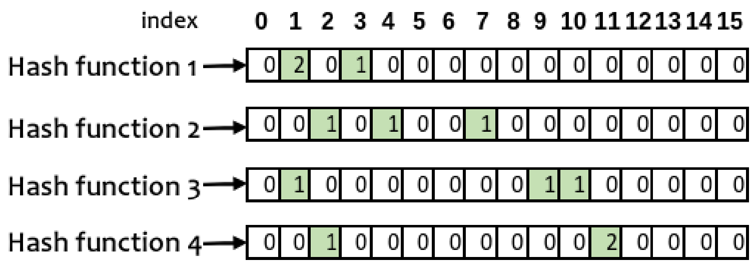


Figure 15: Count-Min Sketch algorithm. The image shows a CMS hash table with four hash functions (each with 16 possible output values) and three items hashed. A hash conflict arises in buckets with the value 2. Source: Dąbrowski et al. [2021].

With embedding sizes of usually 512 to 1024 dimensions depending on method, it is a crucial requirement. Thus, I selected the Efficient Manifold Density Estimator (EMDE) [Dąbrowski et al., 2021] - a fast multidimensional density estimator created at Synerise - as the basis of further research.

4.2.3 Preliminaries

Below I present the Latent Sensitive Hashing and Count-Min Sketch algorithms, which offer some intuitions into EMDE - the probability density estimator used in this research.

Locality Sensitive Hashing (LSH) [Indyk and Motwani, 2000] is an algorithmic technique that applies hashing to obtain representations of all inputs. It is assumed that similar inputs will be hashed into the same ”buckets” (or ”regions” in the item space) with high probability, thus maximizing the probability of hash collisions of similar items. Since the number of buckets is much smaller than the set of possible input items, LSH allows to decrease dimensionality space and save on compute resources.

Count-Min Sketch (CMS) is a probabilistic data structure that serves as a frequency table of events for streaming data. Whenever a new item arrives in the stream, CMS hashes it using a set of N hash functions. Subsequently, a CMS hash table is incremented by 1 in the cells (hashing buckets) corresponding to the obtained hash values. The occurrence count of any element can be retrieved at any time by checking the values which correspond to its hashing buckets and taking the minimum value of all retrieved values. Hash collisions result in too large counts, thus the obtained results of occurrence counts are approximate. An advantage of CMS is that it uses only sub-linear space, at the expense of overcounting some events due to collisions. An example of the CMS procedure is shown in Figure 15.

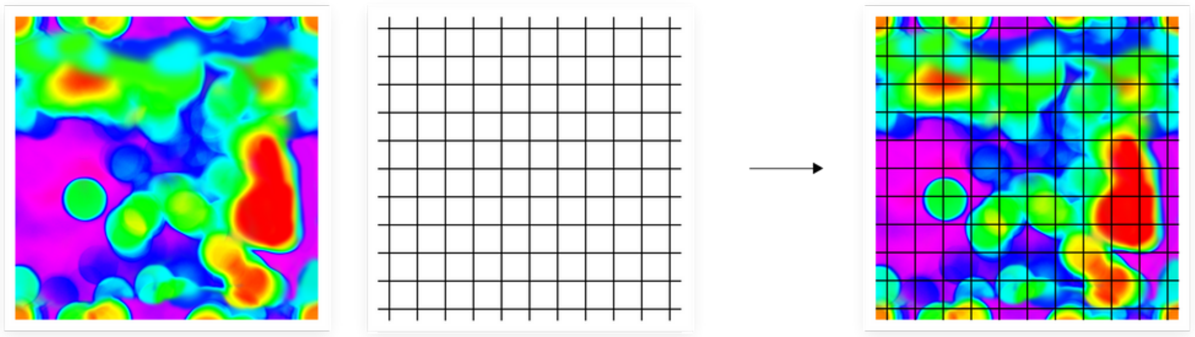


Figure 16: Example of a density-agnostic LSH. A 2D space with a certain probability density of datapoints is cut into equally-sized ”buckets”. Many cuts are created in both high-density and low-density areas. The scale of colors denoting probability densities runs from red (highest density) through yellow, green, and blue, to violet (lowest density). Source: `https://sair.synerise.com/emde-illustrated/`.

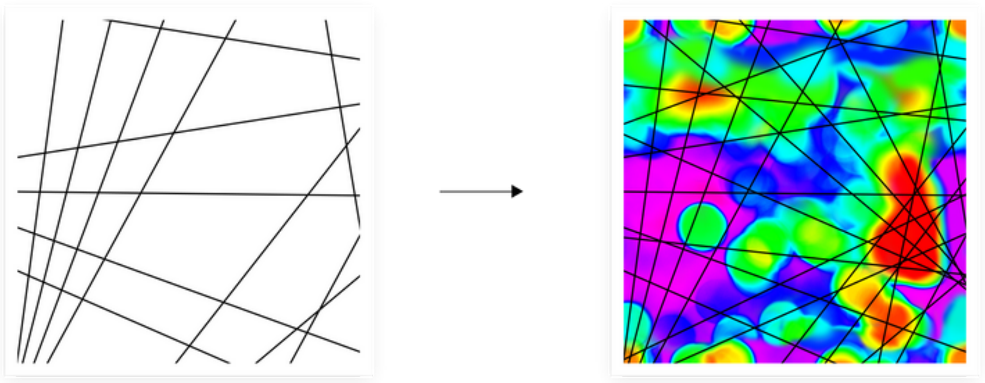


Figure 17: Example of a density-aware LSH. A set of hyperplanes is created based on data densities and a 2D space is cut in such a way that most cuts are produced in the most populous areas. The scale of colors denoting probability densities runs from red (highest density) through yellow, green, and blue, to violet (lowest density). Source: `https://sair.synerise.com/emde-illustrated/`.

4.2.4 The EMDE Algorithm

EMDE is a one-pass algorithm for efficient representation of smooth probability densities on Riemannian manifolds.

Given samples from a data manifold $A \subset R^n$, EMDE first constructs a density-dependent representation called a *sketch*. To this end, it utilizes a modified version of LSH algorithm called a density-aware LSH or D-LSH. In D-LSH, after standardizing the data and projecting to a desired target dimensionality, for $v \in A$ and random vectors r, it holds that $hash(v) = sgn(vr+b)$, drawing the bias value b from $Q(U \sim \text{Unif}[0,1])$, where Q is the quantile function of vr. In contrast to vanilla LSH, this scheme is density-dependent, ensuring that manifold cuts are run through regions where the data is present in high densities. The difference between density-agnostic and density-aware LSH is exemplified in Figures 16 and 17. D-LSH also makes sure that the manifold is always cut into two non-empty parts, avoiding unutilized regions. Multiple independent binary hash values are combined into bit strings, which are interpreted as short integers, giving a partitioning of the input manifold into 2^K regions. To account for the randomness in the process and the fact that stochastically drawn manifold cuts might not always be perfect, the whole procedure is repeated N times, giving N independent manifold divisions. This results in a *sketch* of *width* 2^K and *depth* N.

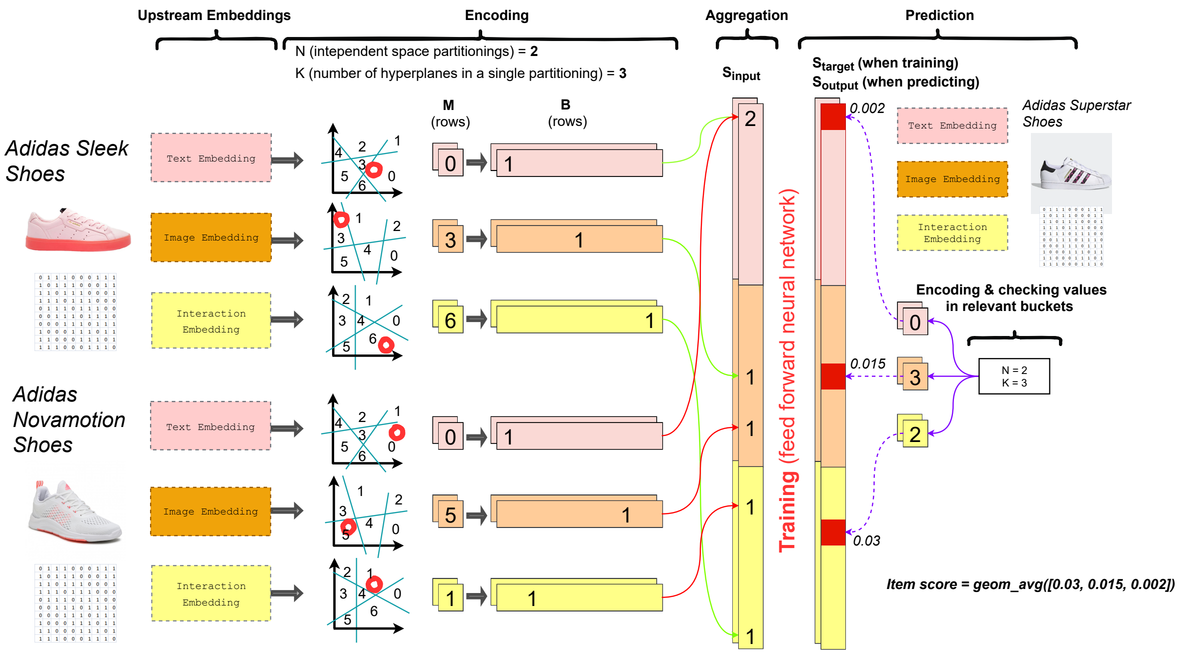
The sketch is thus a compressed map covering the data manifold: every point on the manifold is assigned to N regions, one for each partitioning. Let $V(x) : M \rightarrow (e_1, \dots, e_N)$ where $e_i \in \{1, \dots 2^K\}$ be the mapping function. An empty $sketch$ is instantiated as 2-dimensional array $N \times 2^K$. Local similarity of the data manifold allows $V(x)$ to assign semantically similar inputs to the same sketch regions frequently. This effect captures the essence of the underlying representation learning method which produced the vectors M.

4.2.5 EMDE-based Top-K Recommender

EMDE exhibits a number of unique properties which are beneficial to real-life multimodal tasks:

- The agnosticity of EMDE towards the meaning of the underlying multidimensional space means that any modality which can be represented with an embedding vector can be treated by EMDE. Thus, it is viable to verify that a simple concatenation of multiple EMDE sketches can produce a meaningful multimodal input.

- It produces density estimates over multidimensional spaces, which intuitively links to successful probabilistic approaches in top-k recommendation [Liang et al., 2018]. The previous evidence stressing the importance of estimating user's probability distributions on the item space gives an inspiration that direct density estimates in user-item feature spaces might be a computationally simpler yet successful approach. Thus, an approach where we compute EMDE sketches (understood as approximate density estimates) of user history over all items seems viable.



Upstream Embeddings
Encoding
Aggregation
Prediction
N (intependent space partitionings) = 2
K (number of hyperplanes in a single partitioning) = 3
M (rows)
B (rows)
S_{input}
S_{target} (when training)
S_{output} (when predicting)
Adidas Sleek Shoes
Adidas Novamotion Shoes
Adidas Superstar Shoes
Text Embedding
Image Embedding
Interaction Embedding
Training (feed forward neural network)
Encoding & checking values in relevant buckets
N = 2
K = 3
0.002
0.015
0.03
Item score = geom_avg([0.03, 0.015, 0.002])

Figure 18: The EMDE Algorithm. The example shows a shoe recommendation based on previously purchased shoes. Each modality embedding enters the EMDE space partitioning (”Encoding”) stage, which produces *sketches*. Sketches are aggregated into one joint sketch in the ”Aggregation” phase. A simple neural network then ingests the joint sketch and produces the target sketch with predicted buckets. Source: Dąbrowski et al. [2021].

- The *sketches* have constant size independent of the number of samples and the original embedding dimensions. This bears a significant consequence for a potential EMDE-based recommendation system. The vector recording user history can stay constant and the size of a downstream neural model does not depend on the number of samples and embedding dimensionality. Very large user interaction histories do not increase the model size.

- The readout method from a *sketch* is simple and fast. It is the same as the readout operation from CMS. Thus, having a density estimate of user interests over items encoded with an EMDE sketch, individual scores for items can be retrieved with a simple operation.

In order to apply EMDE to recommendation tasks, I propose a model presented in Figure 18. It involves a neural network which maps from an input sketch to an output sketch vector.

However, to create a summary sketch of all items a user interacted with, it is necessary to formulate a sketch aggregation algorithm, which works analogously to the Count-Min Sketch computation. Formally, it proceeds as follows: given a set H of samples along with their weights W, an en empty sketch structure is instantiated: a vector $sketch$ filled with zeros, which corresponds to zero density, and perform an encoding procedure of the form $sketch[V(x)] := sketch[V(x)] + w$ for all $x \in H$ and corresponding $w \in W$. Our final representation is $sketch := sketch/\sum w \in W$. An additional advantage is that such a sketch can be constructed incrementally in a streaming setting.

Such aggregated EMDE sketches are fed at the input of a feed forward network. Each bucket from EMDE sketches forms a separate input unit. The input EMDE sketch represents a part of user shopping history, with individual sketches of products gathered into one total sketch by simple summation and normalization. Similarly, the output of a network is also an EMDE sketch - but this time it is the sketch of the missing part of user shopping history, the one which needs to be predicted. In such scenario, the neurons of the feed-forward neural network learn the dependencies between various input and output buckets. Picture 18 shows the algorithm on an example of product recommendation. The mapping between input *sketches* and output *sketches* is done with a simple feed-forward neural network. The role of the network is to map the values from input sketch buckets to output buckets.

4.2.6 Experimental Procedure and Results

Competitors and Datasets.

Table 8: Top-k recommendation results. Source: Dąbrowski et al. [2021].

Model	Recall@1	NDCG@5	Recall@5	NDCG@10	Recall@10	NDCG@20	Recall@20
			MovieLens 20M				
EMDE	**0.0529**	**0.2017**	**0.1662**	**0.2535**	**0.2493**	0.3053	0.3523
EASE Steck [2019]	0.0507	0.1990	0.1616	0.2530	0.2465	**0.3078**	**0.3542**
MultVAE Liang et al. [2018]	0.0514	0.1955	0.1627	0.2493	0.2488	0.3052	0.3589
SLIM Ning and Karypis [2011]	0.0474	0.1885	0.1533	0.2389	0.2318	0.2916	0.3350
RP3beta Paudel et al. [2016]	0.0380	0.1550	0.1279	0.2004	0.2007	0.2501	0.3018
			Netflix Prize				
EMDE	**0.0328**	0.1512	0.1101	0.1876	0.1652	0.2332	0.2432
EASE Steck [2019]	0.0323	**0.1558**	**0.1120**	**0.2050**	**0.1782**	**0.2589**	**0.2677**
MultVAE Liang et al. [2018]	0.0313	0.1485	0.1109	0.1957	0.1756	0.2483	0.2645
SLIM Ning and Karypis [2011]	0.0307	0.1484	0.1062	0.1952	0.1688	0.2474	0.2552
RP3beta Paudel et al. [2016]	0.0243	0.0946	0.0595	0.1191	0.0863	0.1578	0.1390

Table 9: Results of adding multimodal data to our top-k EMDE recommenders. **EMDE-MM** denotes EMDE with usage of multimodal data, and **EMDE** denotes the non-multimodal version. Source: Dąbrowski et al. [2021].

Model	Recall@1	NDCG@5	Recall@5	NDCG@10	Recall@10	NDCG@20	Recall@20
			MovieLens 20M				
EMDE-MM	**0.0670**	**0.2378**	**0.1963**	**0.2890**	**0.2780**	**0.3358**	**0.3710**
EMDE	0.0529	0.2017	0.1662	0.2535	0.2493	0.3053	0.3523
			Netflix Prize				
EMDE-MM	**0.0388**	**0.1574**	**0.1253**	**0.2155**	**0.1875**	**0.2619**	**0.2645**
EMDE	0.0328	0.1512	0.1101	0.1876	0.1652	0.2332	0.2432

Experiments are conducted on two popular, real-world, large datasets: MovieLens20K and Netflix Prize, described in Section 3.8.3.

As to the competitors, I rely on the best models detected by Ferrari Dacrema et al. [2019]. This work contains a systematic comparison of neural and non-neural algorithms, often pointing to surprisingly superior results of older non-neural approaches. Amongst the leading models identified in this paper are SLIM [Ning and Karypis, 2011] and RP3beta [Paudel et al., 2016]. SLIM (Sparse Linear Mapping) algorithm produces a sparse aggregation coefficient matrix W, which is learned by solving an L1-norm and L2-norm regularized optimization problem. The multiplication of W against the user-item interaction matrix produces a matrix filled with recommendation scores for all items per user. RP3beta is a simple algorithm without an optimization objective. It represents user interactions with items in terms of a user-item bipartite graph. It performs two-steps random walks from users to items and vice-versa, and assigns recommendation scores based on probabilities of the random walks. According to Ferrari Dacrema et al. [2019] many advanced neural algorithms published at top conferences performed worse than such simple approaches, which suggests incomplete evaluation in many research papers.

Additionally, I include a more recent state-of-the-art VAE-based neural model: MultVAE [Liang et al., 2018], and a non-neural algorithm EASE [Steck, 2019]. EASE (Embarassingly Shallow Autoencoder for Sparse Data) works directly on the user-item interaction matrix and relies on a linear model that is geared toward sparse data, for which a closed-form solution is derived in the paper.

Evaluation Methods and Metrics. I reuse the code for dataset preprocessing, training and evaluation from Ludewig et al. [2019] [12]. In particular, I follow their data preprocessing, and exact implementation of performance metrics, to ensure a fair comparison of all algorithms.

On top of EMDE, I use a simple sparse one-hidden-layer feed-forward neural network with 12,000 neurons. 80% of randomly shuffled user items were put into the input sketch and the remaining 20% into the output sketch to reflect the train/test split ratio of items for a single user. In the best-performing top-k multimodal configuration I include the interactions of users with disliked items (items which received a rating lower than 4).

Results. As shown in Table 8, EMDE is found to outperform the competitors by a significant margin. Thanks to the multimodal nature of EMDE, it can ingest many types of embeddings. In this case, these are Cleora embeddings of liked items (rating of 4 or 5) and disliked items (rating lower than 4) computed with 1 or 2 Cleora iterations, and computed on separate graphs and treated as separate modalities. This proves the main hypothesis that EMDE can outperform the competitors and make adequate use of multimodal data. On multiple large-scale datasets, it achieves mostly state-of-the-art results. The ability to incorporate multimodal data is found to be crucial for performance, as results of the multimodal versions were up to 9-10% better than their non-multimodal counterparts, as displayed in Table 9. In terms of the multimodal version, it was enough to include the disliked items in the form of an additional sketch to achieve substantially improved results.

Additionally, we also try the EMDE model in a session-based scenario, with equally positive outcomes compared to both neural and non-neural competitors, shown in Table 10.

[12]`https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation`

Table 10: Session-based recommendation results. Top 5 competitors are displayed for each dataset. `EMDE MM` denotes multimodal configurations, while `EMDE` denotes non-multimodal configurations. Source: Dąbrowski et al. [2021].

Model	MRR@20	P@20	R@20	HR@20	MAP@20	Model	MRR@20	P@20	R@20	HR@20	MAP@20
			RETAIL						RSC15		
EMDE MM	**0.3664**	**0.0571**	**0.5073**	**0.6330**	**0.0309**	EMDE MM	**0.3116**	**0.0743**	0.5000	0.6680	0.0352
EMDE	**0.3524**	0.0526	**0.4709**	**0.5879**	0.0282	EMDE	**0.3104**	0.0730	0.4936	0.6619	0.0346
VS-KNN	0.3395	**0.0531**	0.4632	0.5745	0.0278	CT	0.3072	0.0654	0.471	0.6359	0.0316
S-KNN	0.337	0.0532	0.4707	0.5788	**0.0283**	NARM	0.3047	**0.0735**	**0.5109**	**0.6751**	**0.0357**
TAGNN	0.3266	0.0463	0.4237	0.5240	0.0249	STAMP	0.3033	0.0713	0.4979	0.6654	0.0344
Gru4Rec	0.3237	0.0502	0.4559	0.5669	0.0272	SR	0.301	0.0684	0.4853	0.6506	0.0332
NARM	0.3196	0.044	0.4072	0.5549	0.0239	AR	0.2894	0.0673	0.476	0.6361	0.0325
			DIGI						NOWP		
EMDE MM	0.1731	**0.0620**	**0.3849**	**0.4908**	**0.0268**	EMDE MM		*no multimodal data found*			
EMDE	0.1724	**0.0602**	**0.3753**	**0.4761**	**0.0258**	EMDE	**0.1108**	**0.0617**	**0.1847**	**0.2665**	0.0179
VS-KNN	**0.1784**	0.0584	0.3668	0.4729	0.0249	CT	0.1094	0.0287	0.0893	0.1679	0.0065
S-KNN	0.1714	0.0596	0.3715	0.4748	0.0255	Gru4Rec	0.1076	0.0449	0.1361	0.2261	0.0116
TAGNN	0.1697	0.0573	0.3554	0.4583	0.0249	SR	0.1052	0.0466	0.1366	0.2002	0.0133
Gru4Rec	0.1644	0.0577	0.3617	0.4639	0.0247	SR-GNN	0.0981	0.0414	0.1194	0.1968	0.0101
SR-GNN	0.1551	0.0571	0.3535	0.4523	0.0240	S-KNN	0.0687	0.0655	0.1809	0.245	**0.0186**
			30MU						AOTM		
EMDE MM	**0.2703**	**0.1106**	**0.2503**	**0.4105**	**0.0331**	EMDE MM		*no multimodal data found*			
EMDE	**0.2673**	**0.1102**	**0.2502**	**0.4104**	**0.0330**	EMDE	**0.0123**	0.0083	0.0227	0.0292	0.0020
CT	0.2502	0.0308	0.0885	0.2882	0.0058	CT	0.0111	0.0043	0.0126	0.0191	0.0006
SR	0.241	0.0816	0.1937	0.3327	0.024	NARM	0.0088	0.005	0.0146	0.0202	0.0009
Gru4Rec	0.2369	0.0617	0.1529	0.3273	0.015	STAMP	0.0088	0.002	0.0063	0.0128	0.0003
NARM	0.1945	0.0675	0.1486	0.2956	0.0155	SR	0.0074	0.0047	0.0134	0.0186	0.001
VS-KNN	0.1162	0.109	0.2347	0.383	0.0309	S-KNN	0.0054	**0.0139**	**0.039**	**0.0417**	**0.0037**

4.2.7 Summary

In this section I have proven that density estimation-based modeling of user-item spaces is a valid recommendation method, which increases recommendation quality for top-k recommendation. At the same time, it allows for easy incorporation of extra modalities, which allows for further boosts in recommendation quality. As such, Hypothesis 2 from Section 1.2 is proven.

4.3 Uni-modal content-based recommender

This section covers Hypothesis 3 from Section 1.2.

The publication on which this section is based: Mikołaj Wieczorek, Barbara Rychalska, Jacek Dąbrowski. ”On the Unreasonable Effectiveness of Centroids in Image Retrieval.” *Proceedings of the 28th International Conference on Neural Information Processing (ICONIP 2021)*, 140 MNiSW points. * – equal contribution

4.3.1 Introduction

Instance retrieval in terms of content-based recommendation is the problem of matching an item from a query to objects present in an item set (gallery). This task is particularly important when no interactions exist for a given item yet, and the only way to recommend the item can be via its visual, textual, or categorical features. A basic scenario of this task is the uni-modal case, where the query modality is the same as the gallery's modality. The uni-modal content-based recommender proposed in this section is focused on visual-visual matching, i.e. searching in an image database based on an image query. However, it can be easily transformed into a uni-modal content-based recommender for any modality, be it text, audio, or graph data.

4.3.2 Background

Most existing instance retrieval solutions use Deep Metric Learning methodology, in which a deep learning model is trained to transform images into a vector representation, so that samples from the same class are close to each other [Liu et al., 2016b, Luo et al., 2020]. At the retrieval stage, the query embedding is scored against all gallery picture embeddings and the most similar ones are returned. Until recently, a lot of articles used classification losses for the training of retrieval models. Currently, most works use comparative/ranking losses and the Triplet Loss is one of the most widely used approaches. Triplet Loss works on an anchor image A, a positive (same class) example P and a negative example belonging to another class N. The objective is to minimize the distance between $A - P$, while pushing away the N sample. The loss function is formulated as follows:

$$\mathcal{L}_{triplet} = \left[\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha\right]_+, \tag{25}$$

where $[z]_+ = max(z, 0)$, f denotes embedding function learned during training stage and α is a margin parameter which defines the required difference between positive and negative samples.

The computation of similarities with individual images as performed in existing systems ignores an important bit of information - we always know which images belong to the same gallery. This inspires an idea for the reduction of the search space. Computation of similarity between query and a representation of a whole gallery instead of each picture presents itself as a viable way for speedup and potentially a form of regularization.

4.3.3 Proposed Algorithm

We propose a system of matching query images to representations of whole galleries, called the *centroids*. Usually, each product from the item stock is characterized by multiple pictures (e.g. the same product photographed from multiple angles or in multiple sceneries). Thus, instead of trying to match the query to each product photo individually, the idea is to first compute a centroid (average) of all embeddings of a single product's pictures, and use it for matching. Figure 19 shows the concept of centroid-based vs. instance-based retrieval.

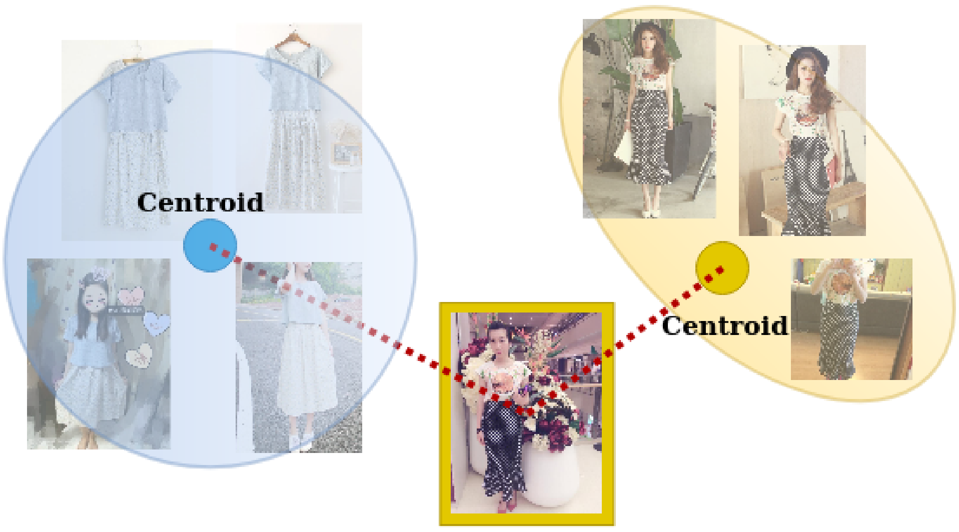
In order to use centroids, the classic Triplet Loss needs to be upgraded to Centroid Triplet Loss (CTL). Instead of comparing the distance of an anchor image A to positive and negative instances, CTL computes the distance between A and class centroids c_P and c_N representing either the same class as the anchor or a different class respectively. CTL is therefore formulated as:

$$\mathcal{L}_{triplet} = \left[\|f(A) - c_P\|_2^2 - \|f(A) - c_N)\|_2^2 + \alpha_c\right]_+ . \tag{26}$$

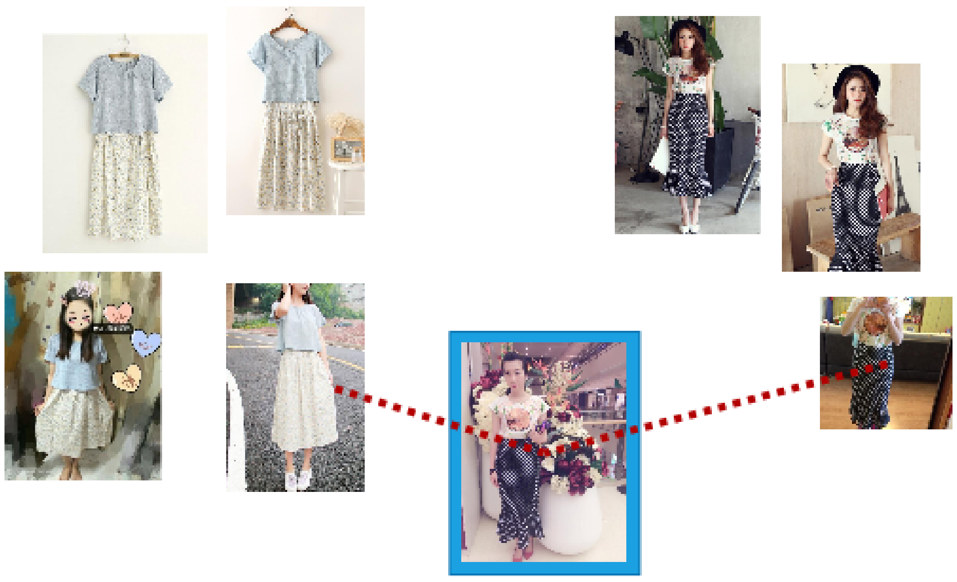
Such formulation of loss can help regularize the model and make it more robust against irrelevant variations in the item galleries.

The Centroid Triplet Loss is introduced on top of other loss functions in a recent state-of-the-art model in visual retrieval [Wieczorek et al., 2020]. The model from Wieczorek et al. [2020] embeds images with a variation of the ResNet architecture [He et al., 2016] dedicated to image processing, and passes them through a simple feed-forward architecture with tricks such as average pooling and batch normalization. Three separate loss functions are computed at various stages of forward propagation: the Triplet Loss, ordinary classification loss, and so-called Center Loss [Wen et al., 2016a] which is a helper to Triplet Loss. Center Loss punishes the distances between representations of photos from the same gallery. In contrast to Triplet Loss, it punishes absolute distances between representations instead of relative distances.

The CTL is added next to other elements to specifically target intra-gallery noise. Figure 20



(a) Centroid-based retrieval



(b) Instance-based retrieval

Figure 19: Centroid-based versus instance-based image retrieval. The top image shows the setting of centroid based retrieval, where each product gallery is represented by a single centroid. In this case, only comparisons between the query example and the centroids are performed. The bottom image shows the popular instance-based retrieval. Images from each gallery are treated as separate instances and many more comparisons must be performed. Source: Wieczorek et al. [2021].

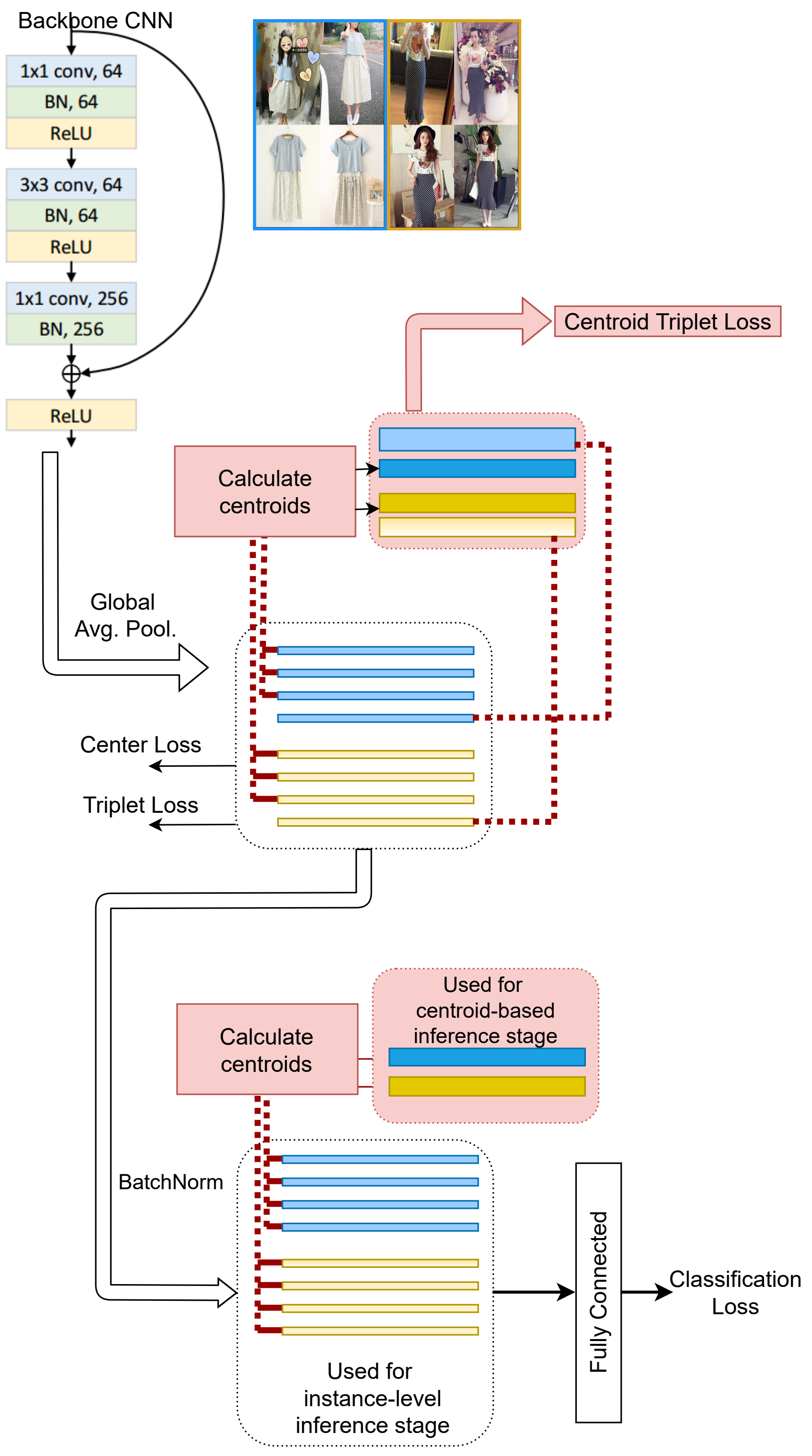


Figure 20: The uni-modal content-based recommender architecture. The parts in red represent the mechanisms added over [Wieczorek et al., 2020]. The model from [Wieczorek et al., 2020] is the current state-of-the-art in image retrieval. It is based on a simple neural model with multiple contrastive losses computed in an instance-based manner. The new parts are drawn in red: the centroid computations for Centroid Triplet Loss and extra modules for centroid-based retrieval. Source: Wieczorek et al. [2021].

shows the full architecture. Next to CTL, centroid-based retrieval is introduced using gallery representations from the last layer of the model. Since the model learns to both represent individual pictures as well as whole galleries during training, it can be expected that centroids computed from internal model representations will work with individual picture representations. Note that this model, once trained, can be still do ordinary instance-based retrieval next to centroid-based retrieval.

Table 11: Fashion Retrieval Results. S or L in the model name indicates the input image size, either Small (256x128) or Large (320x320). R50 or R50IBN suffix indicates which backbone CNN was used, Resnet50 or Resnet50-IBN-A respectively. **CE** at the end of the model name denotes centroid-based evaluation (retrieval). Source: Wieczorek et al. [2021].

Dataset	Model	mAP	Acc@1	Acc@10	Acc@20	Acc@50
DeepFashion	SOTA (S-R50) [Wieczorek et al., 2020]	0.324	0.281	0.583	0.655	0.742
	CTL-S-R50	0.344	**0.298**	0.612	0.685	0.770
	CTL-S-R50 **CE**	**0.404**	0.294	**0.613**	**0.689**	**0.774**
	SOTA (L-R50IBN) [Wieczorek et al., 2020]	0.430	**0.378**	0.711	0.772	0.841
	CTL-L-R50IBN	0.431	0.376	0.711	0.776	0.847
	CTL-L-R50IBN **CE**	**0.492**	0.373	**0.712**	**0.777**	**0.850**
Street2Shop	SOTA (S-R50) [Wieczorek et al., 2020]	0.320	0.366	0.611	0.606	-
	CTL-S-R50	0.353	0.418	0.594	0.643	0.702
	CTL-S-R50 **CE**	**0.498**	**0.432**	**0.619**	**0.660**	**0.721**
	SOTA (L-R50IBN) [Wieczorek et al., 2020]	0.468	**0.537**	0.698	0.736	-
	CTL-L-R50IBN	0.459	0.533	0.689	0.728	0.782
	CTL-L-R50IBN **CE**	**0.598**	**0.537**	**0.709**	**0.750**	**0.792**

4.3.4 Experimental Procedure and Results

Competitors and Datasets. The centroid-based model is evaluated on popular datasets from the visual uni-modal retrieval area: DeepFasion and Street2Shop, described in Section 3.8.5. The scores of two models are compared: ”SOTA” denotes the model presented in Wieczorek et al. [2020], and ”CTL” - the proposed centroid-based model. The previous state-of-the-art model from Wieczorek et al. [2020] forms the basis of the CTL model (the CTL loss and retrieval are added on top of this model).

Evaluation Methods and Metrics. The models are evaluated on standard metrics in the

Table 12: Comparison of storage and time requirements between instance and centroid-based models across tested datasets. Source: Wieczorek et al. [2021].

Dataset	Mode	# in gallery	Embeddings filesize (MB)	Total eval time (s)
Deep Fashion	Instances	22k	175	81.35
	Centroids	16k	130	59.83
Street2Shop	Instances	350k	2700	512.30
	Centroids	190k	1500	146.28

domain: mAP (Mean Average Precision) and Accuracy@K, described in detail in Section 3.3.

Results. Evaluation results are presented in Table 11. Each CTL-based model was evaluated in two modes: 1) standard instance-level evaluation on a per-image basis and 2) centroid-based evaluation.

The CTL model performs better than the current state-of-the-art in most metrics across all tested datasets. Especially noticeable is the surge in mAP metric, which can be explained by the fact that the usage of centroids reduces the search space. The reduction of the search space with centroid-based evaluation is coupled with a reduction in the number of positive instances (from several to just one). In summary, the model obtains very competitive results which set the current state-of-the-art in the area. Additionally, based on Table 12, centroid-based retrieval is up to 4 times faster than instance-based retrieval, due to the restriction of the search space.

4.3.5 Summary

Thanks to the combined effects of centroid-based retrieval and the centroid loss, the proposed model is found to increase both the quality and speed of uni-modal retrieval. As such, Hypothesis 3 from Section 1.2 is confirmed.

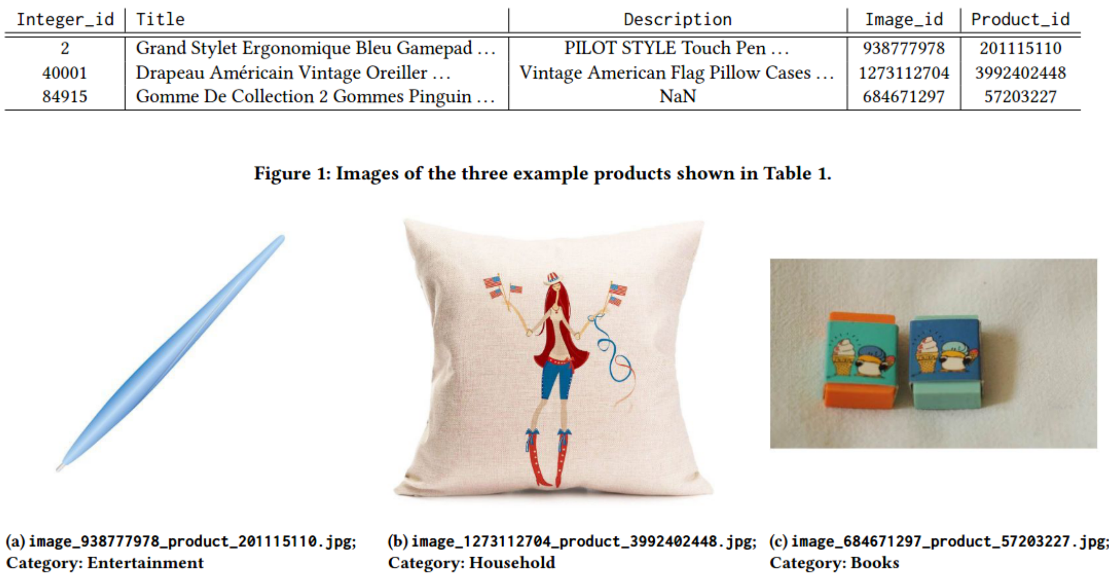
4.4 T-EMDE: Cross-modal content-based recommender

This section covers Hypothesis 4 from Section 1.2.

The publication on which this section is based: Barbara Rychalska*, Mikołaj Wieczorek*, Jacek Dąbrowski. ”T-EMDE: Sketching-based Global Similarity for Cross-Modal Retrieval.” *unpublished*, 2021. * – equal contribution

4.4.1 Introduction

Cross-modal retrieval is yet another important task for content-based recommendation, which relies on item similarities based on their native properties, such as visual features or descriptions instead of user behaviors involving the items. In contrast to the uni-modal retrieval problem described in the previous section, cross-modal retrieval concerns matching items based on two different modalities. Cross-modal retrieval models are usually evaluated on textual-visual modality pair and the task is as follows: given a textual description, find a matching image. A reverse setting also exists: given an image, find a matching textual description. An example of a real-world cross-modal dataset is shown in Figure 21.



Integer_id	Title	Description	Image_id	Product_id
2	Grand Stylet Ergonomique Bleu Gamepad …	PILOT STYLE Touch Pen …	938777978	201115110
40001	Drapeau Américain Vintage Oreiller …	Vintage American Flag Pillow Cases …	1273112704	3992402448
84915	Gomme De Collection 2 Gommes Pinguin …	NaN	684671297	57203227

Figure 1: Images of the three example products shown in Table 1.

(a) image_938777978_product_201115110.jpg; Category: Entertainment

(b) image_1273112704_product_3992402448.jpg; Category: Household

(c) image_684671297_product_57203227.jpg; Category: Books

Figure 21: An example of cross-modal retrieval data from SIGIR eCom Workshop 2020 Data Challenge. Each item from an e-commerce shop is assigned a textual description and a photo image. The task is to match the descriptions to the correct photos (or ”search” for photos using their descriptions). Source: `https://sigir-ecom.github.io/ecom2020/files/Rakuten_Data_Challenge.pdf`.

A natural problem which arises in settings where a comparison of different modalities is required is the *heterogeneity gap* [Cheng and Gu, 2021]. It denotes the hardship of comparing representations which stem from non-related feature spaces. For example, visual representations are usually produced by models dedicated just to vision. They cannot be expected to encode semantic features in the same or similar way as models which process text. Thus, the comparison of representations of images and texts is a notoriously difficult task.

4.4.2 Background

Current state-of-the-art cross-modal recommenders are usually built around the attention mechanism which results in quadratic complexity with regard to the input sequence length (e.g. sequence of words in the textual query) [Diao et al., 2021, Hu et al., 2019, Chen et al., 2020a, Lee et al., 2018]. Thus, the use of attention causes poor scalability with growing input sizes. However, attention and its variation called self-attention (attention computed between each element of a sequence and all other elements of the same sequence) give very high-quality results in practice. This is thanks to the detailed comparison of various tokens or image patches which (self-)attention can compute. Thus, attention-based networks offer high metric results at the cost of speed and scalability.

The idea proposed in my work attempts to bypass this by first transforming representations to a common, abstract representation space, and then comparing such common representations. The idea is centered around the assumption that if the network can learn such translations of various modalities in a cross-modal scenario, then the computed representations should be already easy to compare, representing ”summary similarity”. The demand for detailed token-by-token and patch-by-patch comparisons would then likely disappear. To verify this point, I resort to the ideas within EMDE [Dąbrowski et al., 2021] described in Section 4.2.4 and propose T-EMDE: a trainable version of EMDE, which can encode any modality. The T-EMDE algorithm has linear complexity, which is much more favorable computationally than attention.

4.4.3 Proposed Algorithm

T-EMDE aims to partition the underlying data manifold, similarly to EMDE. However, instead of a static assignment of inputs to specific regions of the manifold, it uses trainable centroids. The centroids are the core concept of item representation - each token/image segment will be characterized by a vector of distances to all centroids. An overview of T-EMDE is shown in

Figure 22 with comparison to basic EMDE.

First, we initialize a trainable centroid tensor of dimensionality $N \times K \times D$. The tensor will hold K centroids for each of N independent manifold divisions. Each centroid will be located within D-dimensional space and their coordinates will change during training to represent data assignments most appropriately, as they will be represented by neural network trainable parameters. The parameter N corresponds to sketch $depth$ from Dąbrowski et al. [2021] (number of independent manifold partitionings), while K corresponds to sketch $width$ (number of regions produced by each partitioning).

Subsequently, we feed each item's modality embedding to a simple linear layer to obtain a projected representation of dimensionality $N \times K \times D$. This way we bring the input embedding into the D-dimensional space in which the cluster centroids live. After this transformation, input data representation can be represented by its proximity to centroids.

After bringing input data into the centroid space, we represent the distance between each example and each centroid as the squared Euclidean distance across all centroid dimensions. Finally, the obtained distance representations are normalized. The resulting vectors can be thought of as representing soft assignments to K centroids each, over N independent space divisions. The vectors are then fed to subsequent layers of the neural network, which trains with any target.

Apart from linear complexity, this algorithm has an additional advantage: it brings all modalities into the same conceptual sphere of centroid-based distances, which potentially can facilitate comparison among various modalities.

Relationship with SGRAF model. Since T-EMDE is a potential replacement for self-attention, it needs to be introduced as a module to a solid cross-modal model for evaluation purposes. Thus, T-EMDE is applied on top of the state-of-the-art cross-modal retrieval model at the time of writing the paper - the SGRAF model [Diao et al., 2021]. The SGRAF is a model of considerable complexity, composed of many layers of deep nets for processing various aspects of two modalities, including convolutional layers, RNNs such as GRU, and various attention mechanisms. For our case, it is necessary to describe its Global Similarity Module, where per-modality self-attention is used. T-EMDE is introduced in this module instead of self-attention.

The Global Similarity Module works as follows: first, all embeddings of image segments and textual tokens are run through separate sequences of multiple linear layers alternating with batch normalization, to form *local* L and *global* G variants of each modality representations. *Global* representation is obtained by averaging per-segment or per-token representations, while

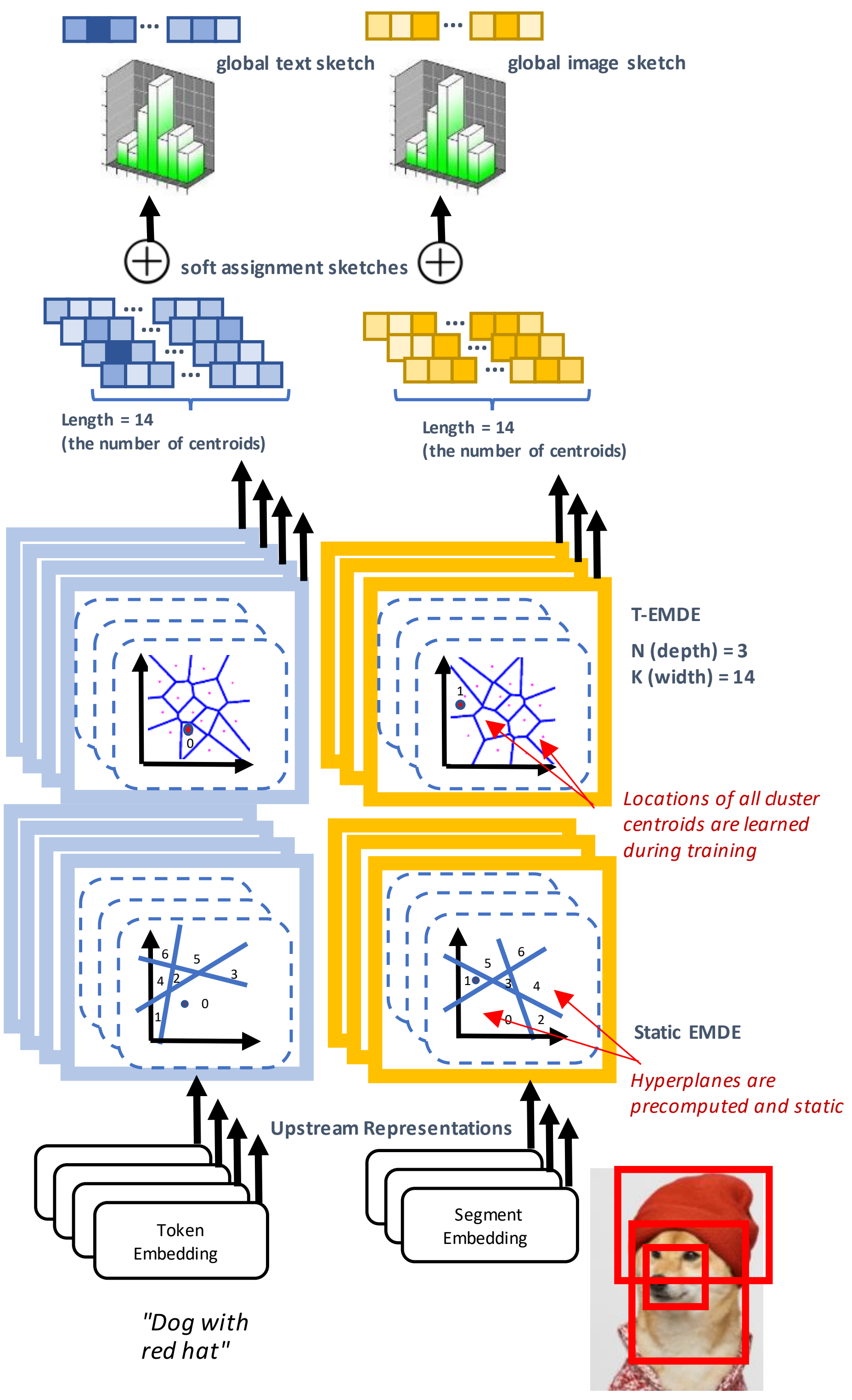


Figure 22: T-EMDE architecture. The figure compares static EMDE to T-EMDE. While the concept of sketches is understood in a similar manner as in EMDE, in T-EMDE the space partitioning is done around cluster centroids rather than by space cuts performed with hyperplanes. Source: Rychalska et al. [2021b].

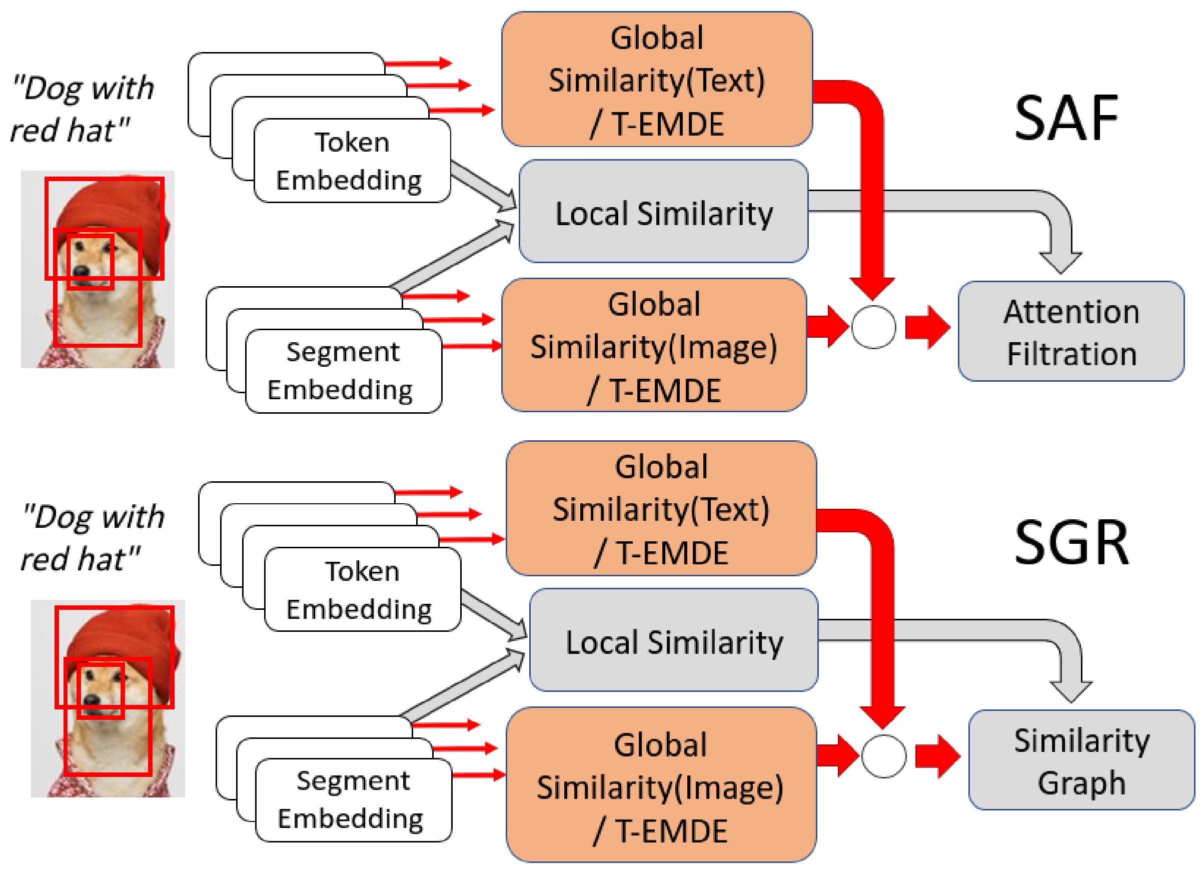


Figure 23: An overview of the SAF and SGR elements of SGRAF, showing where T-EMDE is introduced. Source: Rychalska et al. [2021b].

local representation consists of token/segment-level vectors. Then, self-attention is computed as dot product over the two inputs, with G repeated to match the number of segments/tokens within L:

$$attention\_weights = softmax(L \cdot repeat(G, len(L)))\,,$$

$$new\_G = \sum_{i=1}^{n\_segments} attention\_weights_i \cdot L_i\,.$$

After additional L2-normalization, new_G is the final per-modality global representation. Global similarity vector is computed by subtracting the global caption representation from the global image representation.

After Global Similarity Module, the SGRAF architecture diverges into two alternative processing paths via specialized modules, details of which are not essential to our considerations. The two modules are:

- **SAF** (Similarity Attention Filtration), where text-image representation alignments are evaluated and important alignments are detected,

Table 13: Performance results on Flickr30K (averaged over 5 runs). Source: Rychalska et al. [2021b].

Model	**Flickr30K dataset**							
	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	MRR	R@1	R@5	R@10	MRR
CAAN Zhang et al. [2020a]	70.1	91.6	97.2	-	52.8	79.0	87.9	-
DP-RNN Chen and Luo [2020]	70.2	91.6	95.8	-	55.5	81.3	88.2	-
PFAN Wang et al. [2019]	70.0	91.8	95.0	-	50.4	78.7	86.1	-
VSRN Li et al. [2019]	71.3	90.6	96.0	-	54.7	81.8	88.2	-
IMRAM Chen et al. [2020a]	74.1	93.0	96.6	-	53.9	79.4	87.2	-
SAF no global	71.1	91.6	96.3	0.803	54.1	80.7	87.3	0.660
SAF reported	73.7	93.3	96.3	-	56.1	81.5	88.0	-
SAF reproduced	74.9	93.9	97.1	0.833	56.3	81.8	88.0	0.676
T-EMDE	75.2	94.2	97.1	0.829	57.1	82.2	88.3	0.682
SGR no global	52.8	83.7	92.8	0.664	49.6	77.3	85.1	0.618
SGR reported	75.2	93.3	96.6	-	56.2	81.0	86.5	-
SGR reproduced	76.3	93.2	96.9	0.835	55.2	80.7	88.0	0.666
T-EMDE	77.5	93.1	97.2	0.845	56.9	82.0	87.5	0.679
SGRAF reported	78.4	**94.6**	**97.5**	-	58.2	83.0	89.1	-
SGRAF reproduced	77.5	94.4	97.0	0.849	58.4	83.2	89.0	0.693
T-EMDE	**78.8**	94.4	**97.5**	**0.858**	**59.6**	**83.6**	**89.2**	**0.702**

- **SGR** (Similarity Graph Reasoning), which builds a graph between text-image representation alignments and performs graph reasoning to discover complex matching patterns between representations (similarly as in attention).

Both SAF and SGR can be used simultaneously to create the full SGRAF model.

The Global Simialrity Module is one of the computationally heavy parts of the architecture, especially that it needs to be computed two times - always both for text and image. It is replaced with T-EMDE in our solution. All other parts of the SGRAF model are left as-is.

Table 14: Performance results on MSCOCO-1k (averaged over 5 runs). Source: Rychalska et al. [2021b].

Model	**MSCOCO 1K dataset**							
	Text Retrieval				Image Retrieval			
	R@1	R@5	R@10	MRR	R@1	R@5	R@10	MRR
CAAN Zhang et al. [2020a]	75.5	95.4	98.5	-	61.3	89.7	95.2	-
DP-RNN Chen and Luo [2020]	75.3	95.8	98.6	-	62.5	89.7	95.1	-
PFAN Wang et al. [2019]	76.5	**96.3**	**99.0**	-	61.6	89.6	95.2	-
VSRN Li et al. [2019]	76.2	94.8	98.2	-	62.8	89.7	95.1	-
IMRAM Chen et al. [2020a]	76.7	95.6	98.5	-	61.7	89.1	95.0	-
SAF no global	76.2	95.5	98.1	0.845	61.5	89.3	95.1	0.736
SAF reported	76.1	95.4	98.3	-	61.8	89.4	95.3	-
SAF reproduced	77.2	95.7	98.5	0.853	62.1	89.7	95.4	0.741
T-EMDE	78.3	95.7	98.5	0.858	62.3	89.7	95.2	0.745
SGR no global	76.6	95.8	98.4	0.848	60.9	89.0	95.1	0.731
SGR reported	78.0	95.8	98.2	-	61.4	89.3	95.4	-
SGR reproduced	76.9	95.5	98.5	0.849	61.5	89.4	95.4	0.737
T-EMDE	77.1	95.9	98.5	0.852	61.6	89.5	95.1	0.737
SGRAF reported	79.2	96.2	98.5	-	63.2	**90.7**	96.1	-
SGRAF reproduced	78.9	96.2	98.9	0.864	**63.6**	90.4	**95.8**	**0.752**
T-EMDE	**79.6**	**96.3**	98.7	**0.868**	63.5	90.4	95.6	**0.752**

Table 15: T-EMDE Performance results on the MSCOCO 5K dataset (averaged over 5 runs). Source: Rychalska et al. [2021b].

Model	**MSCOCO 5K dataset**					
	Text Retrieval			Image Retrieval		
	R@1	R@10	MRR	R@1	R@10	MRR
SGM [Wang et al., 2020a]	50.0	87.9	-	35.3	76.5	-
SCAN [Lee et al., 2018]	50.4	90.0	-	38.6	80.4	-
CAAN [Zhang et al., 2020a]	52.5	90.9	-	41.2	**82.9**	-
VSRN [Li et al., 2019]	53.0	89.4	-	40.5	81.1	-
IMRAM [Chen et al., 2020a]	53.7	91.0	-	39.7	79.8	-
SAF noglobal	53.7	90.4	0.663	39.8	80.0	0.532
SAF reported	53.3	90.1	-	39.8	80.2	-
SAF reproduced	55.6	90.7	0.677	40.3	80.3	0.536
T-EMDE	56.7	90.7	0.682	40.3	80.4	0.537
SGR noglobal	54.2	90.3	0.668	39.3	79.3	0.525
SGR reported	56.9	90.5	-	40.2	79.8	-
SGR reproduced	55.7	90.3	0.677	39.8	79.9	0.531
T-EMDE	57.0	91.0	0.685	40.0	80.1	0.533
SGRAF reported	58.8	91.6	-	41.6	81.5	-
SGRAF reproduced	57.9	91.6	0.697	**41.8**	81.5	0.550
T-EMDE	**59.1**	**91.8**	**0.703**	**41.8**	81.7	**0.551**

Table 16: Training and inference times of various configurations (averaged over 5 runs). Source: Rychalska et al. [2021b].

	Flickr 30K		**COCO 1K**		**COCO 5K**
Model	Inference	Training	Inference	Training	Inference
SAF	69.68 s	13h 35m	68.1 s	24h 54m	1725.8 s
T-EMDE	59.68 s	10h 23m	58.1 s	16h 45m	1457.6 s
SGR	83.6 s	17h 35m	84.0 s	48h 52m	2075.0 s
T-EMDE	75.4 s	14h 01m	74.9 s	41h 22m	1830.2 s
SGRAF	161.4 s	-	768.6 s	-	4064.3 s
T-EMDE	135.5 s	-	695.6 s	-	3342.2 s

4.4.4 Experimental Procedure and Results

Datasets and Competitors. Experiments are performed on very popular cross-modal retrieval datasets: MSCOCO [Lin et al., 2014] and Flickr30K [Young et al., 2014] datasets, described in Section 3.8.6.

The selected competitors are high performing, state-of-the-art models. IMRAM [Chen et al., 2020a] captures the mutual image and text alignments with cross-modal attention. Wang et al. [2020a] introduce Scene Graph Matching (SGM) - an approach using separate graphs for each modality to capture object features and their mutual relationships in the corresponding modality. When the object-level and relationship-level features are extracted from both modalities, the final matching on both levels takes place. Stacked Cross Attention Network (SCAN) [Lee et al., 2018] can be defined as two complimentary formulations, where each modality is used as a context to each other and two separate similarity scores are inferred with the use of cross-attention. The already mentioned SGRAF [Diao et al., 2021] builds upon SCAN, extends it with self-attention and applies it to each modality separately. All these models, although effective, become inefficient in a real-life production setting due to the use of the inherently slow attention mechanism.

Evaluation Methods and Metrics. The evaluation is aligned with Diao et al. [2021] for a fair comparison, we reuse their evaluation code[13] for full credibility and evaluate on the same datasets.

[13] `https://github.com/Paranioar/SGRAF`

The evaluation uses a popular cross-modal retrieval split of the MSCOCO dataset (the “Karpathy” split) which assigns 113,287 images for training, 5,000 images with 25,000 matching captions for validation and 5,000 images with 25,000 matching captions for testing. This makes the training, validation, and test sets the same as the competition in their published works. Two validation protocols are used:

- MSCOCO 1K - the results are computed by averaging over 5 folds, each of 1000 test images.
- MSCOCO 5K - the results are computed on the full set of 5000 images.

Regarding Flickr dataset, the split from Frome et al. [2013] is used, with 1,000 images for validation, 1,000 images for testing and the rest assigned for training.

As regards metrics, Recall@1, Recall@10, and Mean Reciprocal Rank (MRR) are used. Recall@1 and Recall@10 have been widely used in recent publications, such as Diao et al. [2021]. Additionally, MRR is computed on T-EMDE, SAF and SGR, because their performance differences are sometimes very small in terms of Recall@1 and Recall@10. By the introduction of an extra metric, it is possible to differentiate their performance results better and provide better confidence in quality results.

Results. T-EMDE can be shown to be both faster and bring higher quality than the attention-based models. As evidenced in Tables 13, 14, and 15, we can observe that T-EMDE brings significant improvements in most metrics, especially the Recall@1 metric, in many cases establishing new state-of-the-art results, which leads us to think that it facilitates text/image matching while providing very fine-grained representations. The performance gains can be especially important in practical scenarios, as the Recall@1 metric denotes the top retrieved items which appear at the start of the user recommendation list and determine the first user experience.

It turns out that the results reported in Diao et al. [2021] differ from the results I obtain for the competitors. For this reason, in Tables 13, 14, and 15 ”SAF reported”/”SGR reported” denotes scores from the SGRAF paper, and ”SAF reproduced”/”SGR reproduced” denotes scores reproduced by me. Notably, the reproduced results proved to be markedly better. ”SAF no global” and ”SGR no global” are used as baselines to show their versions without the global module where T-EMDE and attention are applied.

Concerning model speed, T-EMDE is found to be up to 15%-20% faster than attention-based models, according to latency tests reported in Table 16. The speed gains are hampered by the overall complexity of operations that happen in the SGRAF architecture besides the T-EMDE part. However the gain from removing attention is noticeable and is very meaningful in a practical setting in production. It is of crucial importance that speedups are found in both training and inference phases.

4.4.5 Summary

In this section I have proven that the transformation of embeddings from non-matching multimodal feature spaces into a shared representation space increases the quality of cross-modal retrieval. Additionally, it decreases latency, thanks to the favorable computational complexity in comparison to self-attention. As such, Hypothesis 4 from Section 1.2 is proven.

4.5 Robustness Enhancement of Input Modalities with WildNLP

This section covers Hypothesis 5 from Section 1.2.

The publications on which this section is based:

- Barbara Rychalska*, Dominika Basaj*, Alicja Gosiewska, and Przemysław Biecek. ”Models in the Wild: On Corruption Robustness of Neural NLP Systems.” *Proceedings of the 26th International Conference on Neural Information Processing (ICONIP 2019)*, 140 MNiSW points. * – equal contribution
- Barbara Rychalska*, Dominika Basaj*, Anna Wróblewska, and Przemyslaw Biecek. ”Does It Care What You Asked? Understanding Importance of Verbs in Deep Learning QA System.” *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP 2018*. * – equal contribution

4.5.1 Introduction

Neural network models have been recently proven to be vulnerable to so-called adversarial examples, which can cripple performance substantially. A famous case of adversarial example is displayed in Figure 24. Here, an imperceptible (from the human perspective) change to input images results in a drastic change in prediction. After adding just slight noise, the image's predicted class changes from the correct prediction of ”dog” to an incorrect and unexpected prediction of ”ostrich”.

Such problems arise in any type of model input, text, image, graph data, and others [Zhang et al., 2020b]. They can appear either by accident (detriment of input quality, e.g. image compression, adding frames, change of wording in text, unexpected edge in a graph), or by malicious action of an attacker. A number of works study the process of deliberate creation of adversarial examples [Zhao et al., 2018, Athalye et al., 2018, Zhao et al., 2018].

Unfortunately, vulnerability to adversarial examples appears in all kinds of neural models [Ilyas et al., 2019]. Because models can be so brittle, it is important to know their weaknesses so that fixing measures can be attempted. Because adversarial examples can often appear by accident, susceptibility to them is a major threat in production systems.

In the following sections, I describe an approach of testing adversarial robustness of models, together with an approach to make the models more robust. This is done in terms of the WildNLP framework [Rychalska et al., 2019]. I focus mainly on natural-looking adversarial examples in

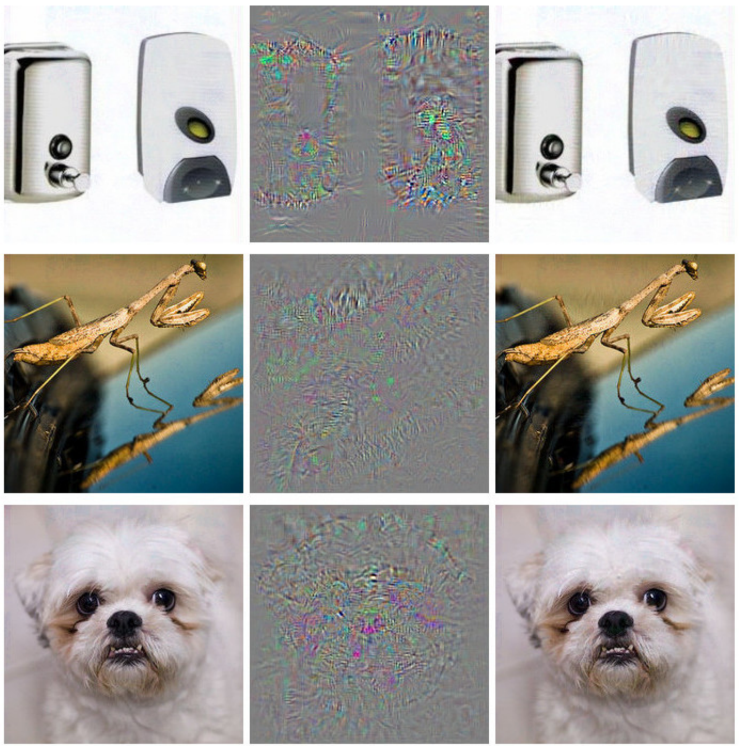


Figure 24: A famous example of adversarial examples: all images in the left column are correctly classified. The middle column shows the imperceptible error added to the images to produce the images in the right column. The slightly disturbed images are all categorized (incorrectly) as “Ostrich”. Source: Szegedy et al. [2014].

the textual domain, which can appear e.g. in textual descriptions of items in online stores, and can be regarded as a major threat to multimodal recommenders.

The work on the benchmark was precluded by a small study of what inputs are actually important for a model, that is which inputs are most threatened by adversarial perturbations.

4.5.2 Introductory Results: Which Inputs are Important to Models

Before diving into the area of robustness enhancement, I pose an introductory question: which parts of the input are actually important for models? Are they consistent with human intuition as to what should be considered important?

Unfortunately, it has been shown before that models sometimes base their predictions on totally unexpected or faulty inference choices. One of the first observations of this erroneous behavior appeared in the Visual Question Answering (VQA) area. VQA is the task of providing a textual answer for a textual question, based on an image - for example, given a picture of a dog, a question might be ”What is the color of the dog’s collar?.” Agrawal et al. [2018] show that

VQA models could in fact answer correctly many simple questions based on an all-black picture. It has been found that questions like ”How many people are in the picture” can be answered with a generic answer ”Two”, and the construction of the dataset was such that it was indeed the right answer for the majority of the ”How many (...)” questions. The correct answer was selected by the model by observing the starting phrase ”How many”, disregarding other inputs. Thus, the models did not learn to interpret the images enough, but mainly to fit to inherent data biases. It can be assumed that VQA is no exception and unexpected patterns of reasoning can be found in other models.

Inspired by the VQA problem, I designed a probe for the textual modality, focusing on textual Question Answering. Question Answering is the task of producing a textual answer to a textual query, based on a block of text (called *context*). In contrast to Visual Question Answering, in textual-only Question Answering questions are far more complex and reliance on context is indispensable. Such models usually cannot just use a generic phrase like ”Two” in answers, because the questions are usually asking about a concrete date or a name appearing in context. I was interested in which parts of questions are in fact important to models, and whether the importance of particular words corresponds to humans' expectations at all.

I operated on the SQuAD dataset [Rajpurkar et al., 2016] and used a high-quality Question Answering model from Chen et al. [2017] for testing. I aimed to answer the following questions: Given that the verb is the core part of speech, does a model's prediction change when the verb is swapped to the opposed meaning (its antonym) in the query? The model should not ignore the verb as swapping changes the whole meaning of the sentence. Thus, the verb could be most threatened by adversarial perturbations. I created an automated procedure of swapping verbs for their antonyms using WordNet [Miller, 1995] antonym base. For auxiliary verbs, a simple negation was inserted (e.g. *is* - *isn't*). For words without antonyms in WordNet, a random verb was inserted. This resulted in verb swaps like the following:

- *How many teams* ***participate*** *in the Notre Dame Bookstore Basketball tournament?* vs. *How many teams* ***drop out*** *in the Notre Dame Bookstore Basketball tournament?*
- *Which art museum* ***does*** *Notre Dame administer?* vs. *Which art museum* ***doesn't*** *Notre Dame administer?*

The obtained experimental results of the model's performance on both original questions and swapped questions show that unexpectedly, swapping verbs for their antonyms did little to change the model outputs. The model predictions were unchanged in 90.5% of cases and model confidence stayed at a constant level too (average decision certainty of 0.6 for modified questions vs 0.61 for original questions). Antonym swapping produced questions which were markedly different from the human perspective, yet the model saw no difference.

Moreover, I have analyzed the internal layers of the 3-layer LSTM recurrent neural network, used within the examined system to compute attention scores for words in the questions. The results showed that the part of speech which the greatest variance in the values within the hidden layers was the noun. This observation aligned with the fact that nouns received the highest attention scores. The average absolute attention score for nouns was found to be 5.43, 2.32 for verbs, and 2.39 for other parts of speech.

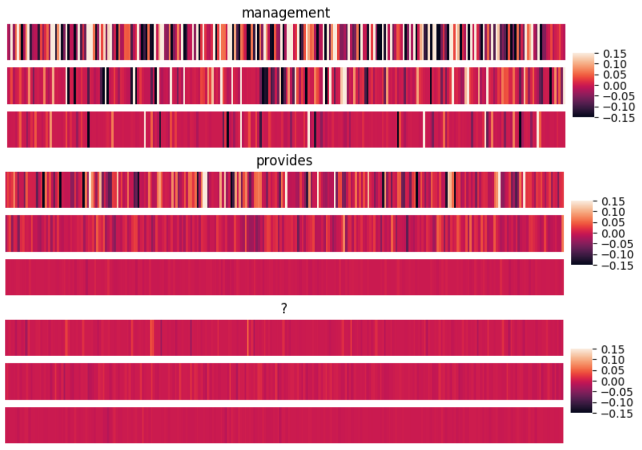


Figure 25: Visualization of values in LSTM hidden layers for a noun (top), verb (middle) and question mark (bottom). Each heatmap shows the 256 values returned by each layer, in 3 layers for each word. Source: Rychalska et al. [2018]

.

Thus, verbs were found to be surprisingly unimportant for the model, seemingly for the sake of nouns. This contradicts the primary role of verbs as creators of meaning in language. A careful analysis of the underlying dataset reveals a likely explanation of this phenomenon. In the case of the SQuAD dataset, a specific noun (usually a named entity) appears in a single sentence in a single context, so contrasting the nouns is enough to extract the answer. Verbs are not needed for this purpose within this particular dataset.

This introductory result shows that because the models sometimes focus on a very limited part of available information, the disturbance of keywords with adversarial examples could lead to even more disastrous results. After obtaining this result, I focused on researching adversarial examples and ways of combating them.

4.5.3 The WildNLP Framework - Background

The results from the previous section support the assumption that model robustness validation should not be limited only to supposedly ”important” parts of the input. Whole inputs should be indiscriminately tested for robustness failures. In response to this, I introduce WildNLP - a testing benchmark for modality robustness against adversarial examples, with a complementary approach for robustness enhancement. The WildNLP benchmark takes into account the methods which produce textual input embeddings to a recommender. In multimodal recommenders, many modality-specific embedding methods will usually be used (see Figure 18). Automated procedures to quickly analyze multiple model candidates and many embedding algorithms are indispensable. WildNLP was researched as a benchmark dedicated to the textual modality. Analogous benchmarking and robustness enhancing procedures can be applied to other input modalities as well. In fact, a number of works for other modalities, such as image data and graph data, have already been published [Hendrycks et al., 2021, Liu et al., 2016a].

The WildNLP is a fully automatic robustness enhancement benchmark which focuses on naturally occurring robustness vulnerabilities, which will continuously appear in production settings in a recommender - e.g. product descriptions written by users, with language errors and inconsistencies regarding the notation of numbers. It can be easily transformed to the visual domain by introducing commonly appearing image corruptions such as poor quality pictures of items. It can be also transferred to the graph domain by introducing erroneous transactions or missing chunks of transactions - such as could happen with a faulty logging system of an e-commerce business.

4.5.4 The WildNLP Framework

WildNLP simulates errors by introducing multiple types of corruptions called *aspects*. For example, it tests whether mixing in improper articles produces valid adversaries (*the*, *a*, or no article). Another example is the swapping of letters based on QWERTY-type keyboard location of characters. Yet another aspect is the misspelling - with examples taken from the list of common orthographic errors in the English language. In total, 11 such corruption aspects are formulated (see Table 17).

Because such corruptions can be generated automatically in large volumes, I propose to amend the robustness problems with so-called adversarial training [Bai et al., 2021b]: create large amounts of corrupted data and fine-tune the models on it in hope of obtaining more robust

Table 17: Examples of text corruptions introduced by WildNLP aspects. Source: Rychalska et al. [2019].

Aspect	Example sentence
Original	Warsaw was believed to be one of the most beautiful cities in the world.
`Article`	Warsaw was believed to be one of **a** most beautiful cities in world.
`Swap`	Warsaw **aws** believed to be one **fo teh** most beautiful cities in the world.
`Qwerty`	Wa**d**saw was b**d**lieved to be one of the most beautiful citie**e** in the world.
`Remove_char`	Warsaw was believed to be one **o th** most **eautiful** cities in the world.
`Remove_space`	Warsaw was believed **tobe** one of the most beautiful cities in the world.
Original	You cannot accidentally commit vandalism. It used to be a rare occurrence.
`Misspelling`	You can not **accidentaly** commit vandalism. It used to be a rare **occurrance**.
Original	Bus Stops for Route 6, 6.1
`Digits2words`	Bus Stops for Route **six**, **six point one**
Original	Choosing between affect and effect can be scary.
`Homophones`	Choosing between **effect** and effect can **bee** scary.
Original	Laughably foolish or false: an absurd explanation.
`Negatives`	**Laughab*y fo*lish** or **fal*e**: an **a*surd** explanation.
Original	Sometimes it is good to be first, and sometimes it is good to be last.
`Positives`	Sometimes it is **go*d** to be first, and sometimes it is **goo*** to be last.
`Marks`	Sometimes, it is good to be first and sometimes**,** it**,** is good to be last.

versions of the models (i.e. models which do not degrade in performance while an adversarial example is encountered). Adversarial training was used before with regard to NLP robustness enhancement; however, there were significant differences from the WildNLP approach:

- The perturbations of text were not focused on generating natural-looking errors, likely to appear in real life. They were usually focused on perturbing word embeddings (instead of words themselves) [Sato et al., 2018] or designed for some specific NLP model, such as machine translation [Glockner et al., 2018, Belinkov and Bisk, 2018]. In the case of multimodal recommenders, we are not concerned with such perturbations directed at concrete downstream NLP tasks, because we never perform them - we just need high-quality embeddings for use in a recommender model.

- A novel insight of WildNLP is that the WildNLP aspects are related in their nature. For example, Swap and QWERTY aspects both rely on swapping individual characters within words. As such, it can be hypothesized that adversarial training on one aspect should amend robustness to related aspects to a certain degree.

- WildNLP aspects can be introduced with varied levels of severity, meaning that more characters can be altered within a single word, or more words in a sentence can be affected by the corruptions. Increasing severity in adversarial training could amend the models' robustness to the same aspect of lesser severity. With this in mind, the severity concept is introduced to the framework and experiments are conducted on adversarial training on various severity levels.

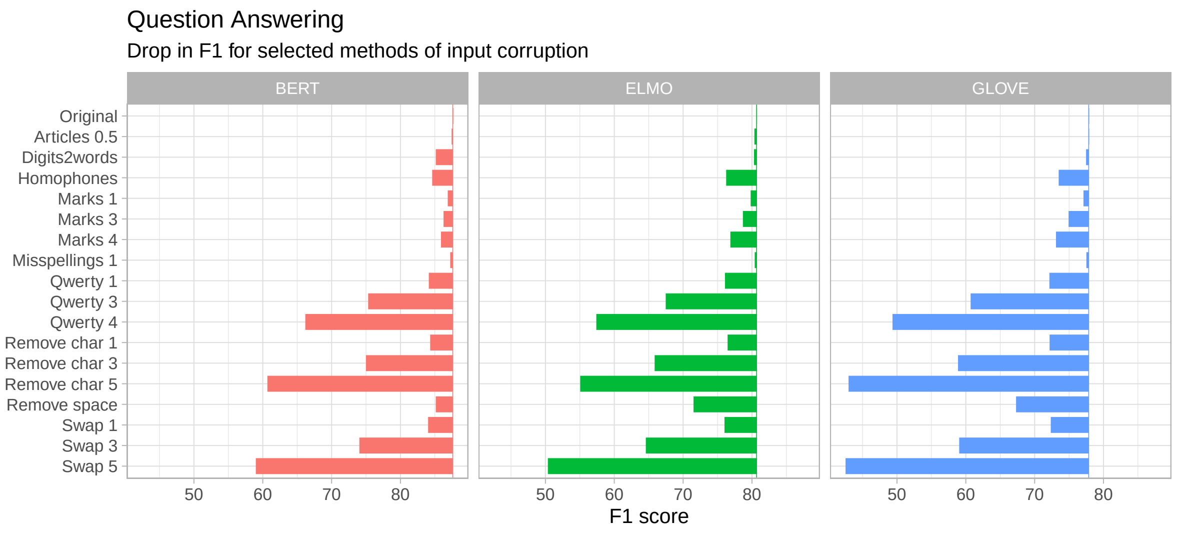


Figure 26: Robustness testing results for Q&A models. Each bar starts in the F1 score for the original data and ends in F1 score for data after the selected method of corruption is applied. The shorter the bar the more robust a given method is. Source: Rychalska et al. [2019].

4.5.5 Experimental Procedure and Results

WildNLP is tested on many embedding methods and downstream tasks to be able to see whether the embeddings' quality degrades or not. The tested embedding models are both modern contextualized models such as BERT [Devlin et al., 2019] and ELMO [Peters et al., 2018], as well as older non-contextualized models such as GloVe [Pennington et al., 2014]. Contextualized models compute embeddings based on current context, which means that the embedding of the same word will be different in every sentence. Non-contextualized models precompute word embeddings once and use these precomputed embeddings irrespective of the current context.

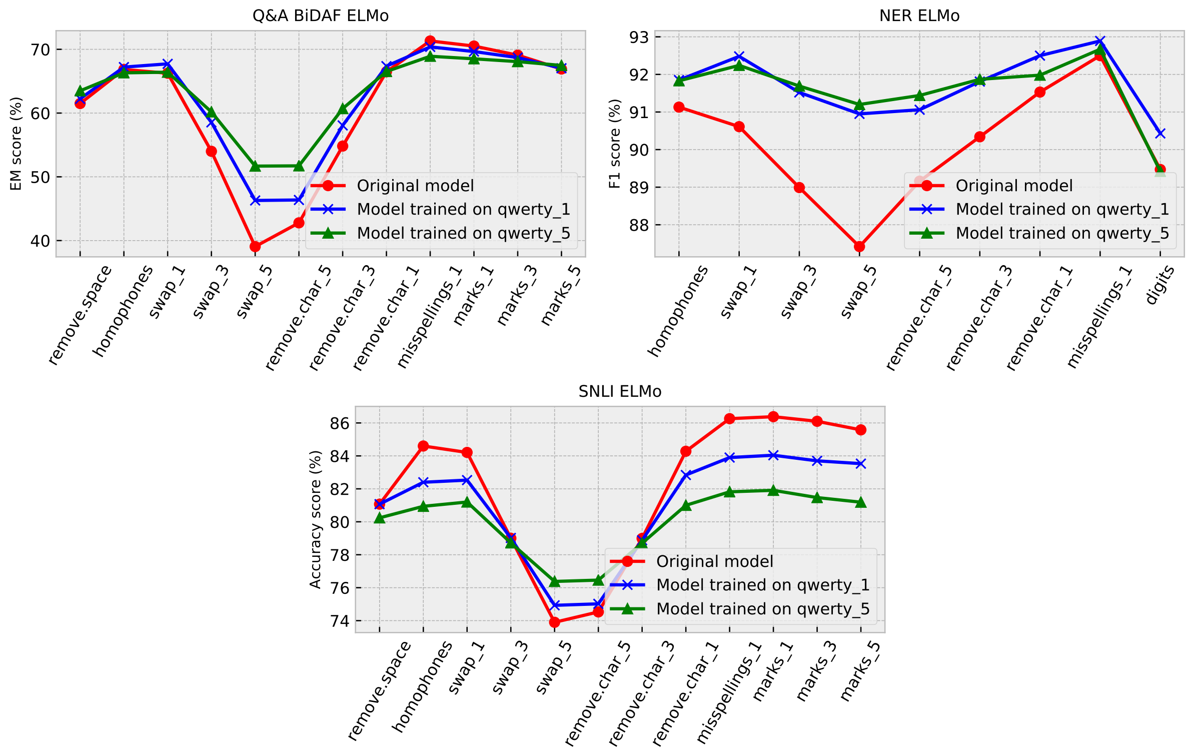


Figure 27: Influence of training on data corrupted with `Qwerty` aspects on testing on other aspects. Source: Rychalska et al. [2019].

Contextualized embedding models enhance standard performance metrics greatly, although it is unclear whether this translates to increased adversarial robustness as well.

I test the performance of various downstream tasks, such as Question Answering, NER (Named Entity Recognition), SNLI (Stanford Natural Language Inference), and Q&A (textual Question Answering). Such a wide array of tasks allows to test the designed method of increasing model robustness from many angles.

Results. As shown in Table 26, the tested models suffer greatly from adversarial examples. BERT and ELMo-based systems were found to mitigate performance loss to some degree compared to GloVe. However, their performance loss pattern across corruptions was found to be similar to GloVe. Generally, contextualized models receive an advantage due to their reliance on the context around affected words (thus, an error in a certain word is not that important). However, the differences in robustness were small, despite huge performance differences in standard performance between contextualized and non-contextualized models. This suggests that improving robustness is a much more difficult task than standard downstream tasks. The small improvements are not yet satisfactory, as even the most sophisticated models lost up to 30%-50% of their performance (depending on corruption severity) due to adversarial examples.

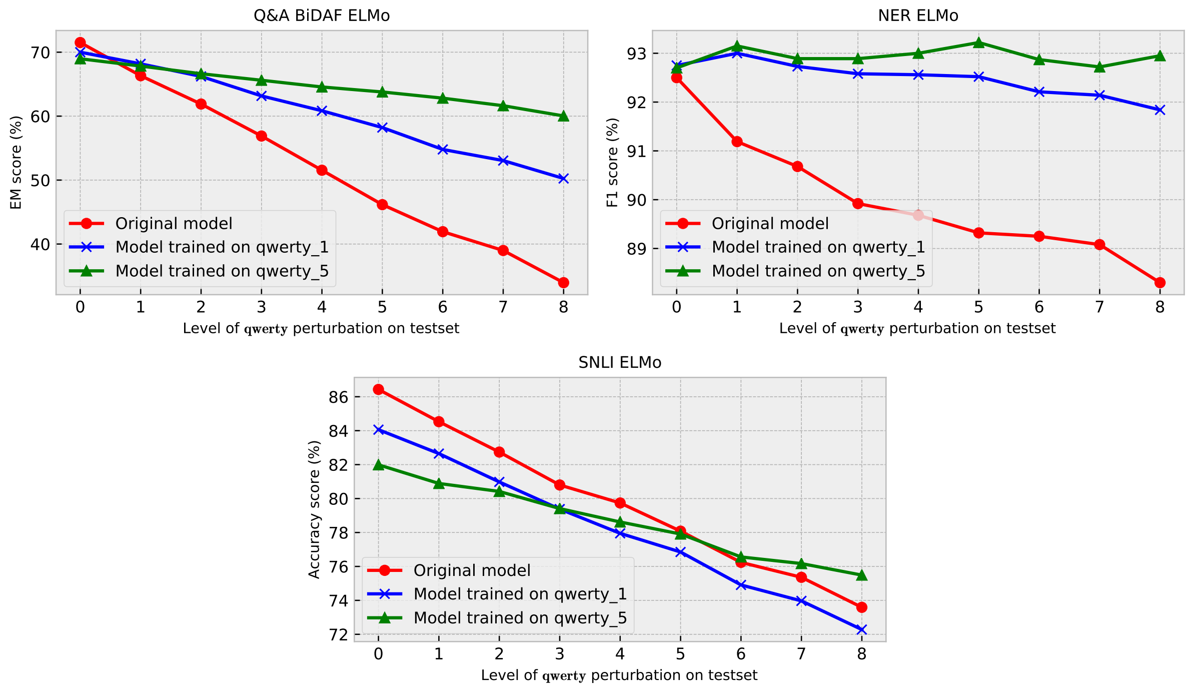


Figure 28: The effect of aspect severity on adversarial training. It can be observed that greater severity usually leads to greater protection from adversaries. Source: Rychalska et al. [2019].

Adversarial training is found to be an effective measure of combating this severe robustness problem. The results are presented in Figure 27. For example, a Q&A ELMo-based model suffered a 15% performance loss after adversarial training, while with no adversarial training it lost around 50% performance. Furthermore, the results show that adversarial training on one of the related corruptions increases performance for all related corruptions. On the example of the QWERTY aspect, adversarial training on this aspect was able to raise performance by up to over 10 p.p (Q&A task), for related aspects such as Swap and RemoveChar. On the other hand, aspects unrelated to QWERTY such as Marks or Homophones did not experience a meaningful beneficial result.

Moreover, increased severity of corruption during training does increase performance on data corrupted with the same aspect type but lesser severity. Figure 28 shows plots for multiple models, tested on QWERTY corruptions of two severity levels - 1 (low severity, 1 word will be affected) and 5 (high severity, 5 words will be affected). Training on severity 5 significantly increased the performance for lesser severities.

4.5.6 Summary

In this section I have shown that adversarial training is an effective countermeasure to the adversarial robustness problem, especially when we take into account the related corruptions and corruption severity. As such, Hypothesis 5 from Section 1.2 is proven.

4.6 Industry Applications of My Research

This section describes how approaches presented in this thesis (Sections 4.1-4.5) contribute to industry products.

The publication on which this section is based: Jacek Dąbrowski and Barbara Rychalska. ”Synerise Monad - Real-Time Multimodal Behavioral Modeling.” *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM ’22)*, 140 MNiSW points.

4.6.1 Introduction

Due to the successful results of the previously described research (Sections 4.1-4.5) me and my colleagues at the company Synerise have made an attempt to commercialize this unique intellectual property. There are currently two products which incorporate this technology: the Synerise Platform and the Monad Platform.

4.6.2 Synerise Platform

The first product based on our algorithms is the Synerise Platform, a real-time interaction management and automation platform (Figure 29). The machine learning side of the platform works analogously to the full recommender system presented in Figure 12. The platform collects user data from customer websites, recording clicks, purchases, page views and similar behavioral event data. It then trains Cleora (Section 4.1) and EMDE (Section 4.2) models for recommendation and related tasks. The platform also contains content-based recommendation and search modules based on our work (Sections 4.3 and 4.4). The customers can choose which subtasks of recommendation and other related tasks (such as search and propensity prediction) will be modeled. All available tasks are solved by virtually the same proprietary algorithms, used exactly as described in this thesis. The platform returns predictions on which the customers can base their sales or marketing strategies. For example, they can discover which users are more likely to buy a certain product in the future, or which products will be more popular at a certain time. The Synerise platform is dedicated to product specialists and marketing teams at customer companies, who do not have enough expertise to train the models themselves. Due to the scalability of our algorithms, the platform can run on basic hardware and retrain the models very frequently.

The Synerise Platform also includes a large number of non-AI submodules, which allow to shape the relationship between the company and its clients. The company can for example observe the dashboards summing up the activity of clients. Certain events generated by users can be programmed to trigger automated responses. The platform can automatically communicate with users by sending emails, SMS messages, and other forms of content. It also assists with many common e-commerce activities such as the creation of promotions or vouchers.

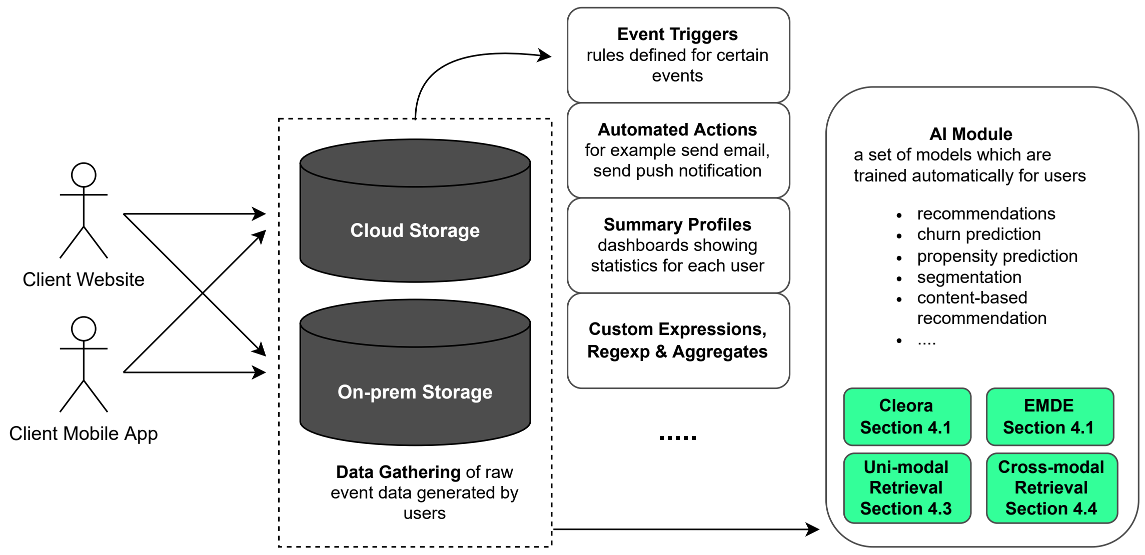

Client Website
Client Mobile App
Cloud Storage
On-prem Storage
Data Gathering of raw event data generated by users
Event Triggers rules defined for certain events
Automated Actions for example send email, send push notification
Summary Profiles dashboards showing statistics for each user
Custom Expressions, Regexp & Aggregates
.....
AI Module a set of models which are trained automatically for users
recommendations
churn prediction
propensity prediction
segmentation
content-based recommendation
....
Cleora Section 4.1
EMDE Section 4.1
Uni-modal Retrieval Section 4.3
Cross-modal Retrieval Section 4.4

Figure 29: Architecture of the Synerise Platform.

4.6.3 Monad Platform

Monad Platform is our second product, this time dedicated to machine learning professionals - Data Scientists, Researchers and Analysts (Figure 30). Currently, time-sensitive heterogeneous and multimodal data is on the rise in industry-grade data lakes (similar to the event data gathered by the Synerise Platform). This data is often spread over multiple databases and tables with complex relationships between them. The sheer size and complexity of the multimodal data formats are a problem for most existing commercial machine learning platforms. Although the Synerise Platform can in principle handle such data, it has a drawback from the perspective of machine learning practitioners - it cannot be freely configured due to its black box nature and predefined use cases.

Monad combines Cleora (Section 4.1), EMDE (Section 4.2), and interpretability methods (Section 4.5) in order to create a flexible tool which can connect to many internal customer data sources, combine various data types and train our algorithms in ways defined by machine

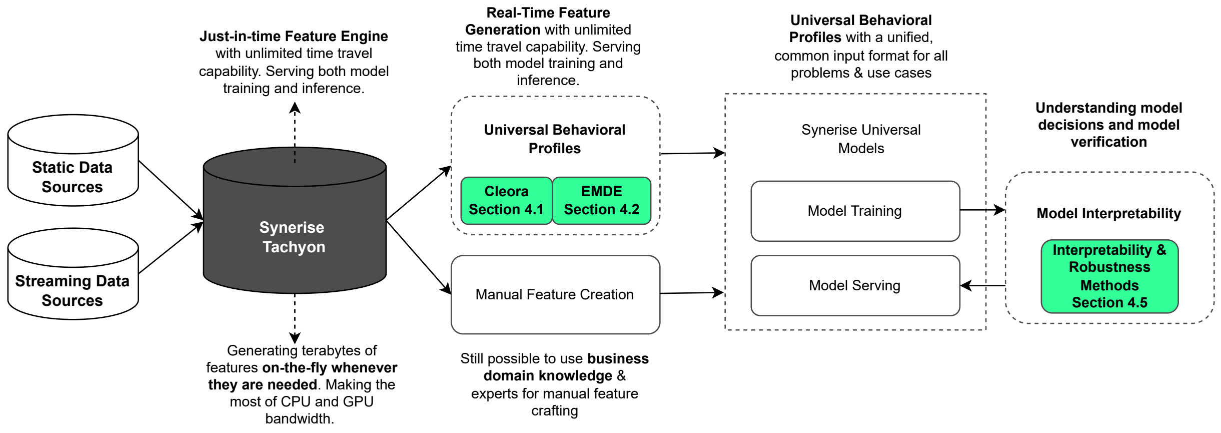

Just-in-time Feature Engine with unlimited time travel capability. Serving both model training and inference.
Static Data Sources
Streaming Data Sources
Synerise Tachyon
Generating terabytes of features on-the-fly whenever they are needed. Making the most of CPU and GPU bandwidth.
Real-Time Feature Generation with unlimited time travel capability. Serving both model training and inference.
Universal Behavioral Profiles
Cleora Section 4.1
EMDE Section 4.2
Manual Feature Creation
Still possible to use business domain knowledge & experts for manual feature crafting
Universal Behavioral Profiles with a unified, common input format for all problems & use cases
Synerise Universal Models
Model Training
Model Serving
Understanding model decisions and model verification
Model Interpretability
Interpretability & Robustness Methods Section 4.5

Figure 30: Architecture of the Monad Platform.

learning experts.

Monad is composed of the following parts (based on Dąbrowski and Rychalska [2022]):

1. **Static Sources and Streaming Sources** (outside the scope of this thesis) – Monad has connectors for various static and streaming data sources, such as Snowflake, Clickhouse and Google BigQuery. It connects to complex, multi-table databases which the companies usually store. These databases can hold various types of events (clicks, page views, add to cart, purchase, and others), belonging to many event types (credit card purchases, money transfers, navigation on websites). Monad also connects to data sources storing user and item properties.

2. **Tachyon** (outside the scope of this thesis) – an analytic data engine, extensively tuned for fast data retrieval. Tachyon is dedicated to serving event data with can be aggregated by a certain criterion, e.g. when we need all events belonging to a single user. It supports inherent temporal data ordering and schemaless data. Tachyon allows to create arbitrary data access/processing functions, which are written by the users themselves. Such functions are compiled and merged as building blocks into data processing pipelines. A core function of Tachyon is that it allows to run the data batches through pipelines, and then discard it after feeding to a machine learning model. Data is never materialized on disk. This makes it suitable for real-time applications.

3. **Self-Supervised Universal Behavioral Profiles** – a distributed engine for computation of entity representations and aggregating them into profiles of entities, for example, users or products. This suite of methods performs automatic feature creation and selection. This

is the stage at which Cleora and EMDE are used. The output of EMDE - the multimodal *sketch* - is understood as a behavioral profile, summing up the activity of a certain user in terms of a single vector. The behavioral profile is served as input to further training and prediction using machine learning models. This stage also includes smart transformation of categorical and decimal features into a small and normalized form, which is outside of the scope of this thesis.

4. **Synerise Universal Models** – a set of automatic model training resources and model architectures for multiple popular use cases in behavioral modeling, such as recommendations, propensity/churn prediction, or user segmentation. Any other type of model can be easily programmed by the user. This stage includes interpretability and robustness methods (Section 4.5) for verification of obtained models.

An exemplary use case for Monad is from the banking industry. The client, a large bank, uses Monad on a dataset of over 1 TB, containing 35 million user entities, with multiple tables and dozens of columns with cardinality of up to 200,000,000 entities. Some event tables have up to 2 billion data points. The data is processed by Monad on a single machine with 256 GB RAM, a single Tesla p100 16 GB GPU card, and 12 CPUs - a standard machine affordable for most companies. This client uses Monad for propensity training on 45 output classes (each output denotes propensity to buy a certain product, with 45 products total). Together with interpretability insights, the processing and training time is 3 hours, with also 3 hours of inference for all 35 million users. Monad automatically produces a total of over 35,000 input features for downstream models. The throughput of real-time predictions is 10,000 users per second.

4.6.4 Summary

The practical advantages of the presented algorithms make them suitable for wrapping in various kinds of commercial products. A chief advantage is speed, combined with high quality of results and multimodal capabilities. To date, two commercial products have been released - the Synerise Platform and the Monad Platform. Each of these platforms is dedicated to different use cases, which creates venues for further research and improvements.

5 Summary

In this work, I have proven all research hypotheses defined in the introduction. As a result of this work, a complete recommender system has been created, which currently works in production within the Synerise Platform and the Monad Platform.

First, I have observed that graph neural networks (GNNs), often found in modern recommender systems, can be simplified to obtain a purely unsupervised algorithm of weighted averaging of node embeddings. Based on this observation, a novel algorithm was proposed and evaluated. It achieves high quality results and can be much faster to train than a full GNN or popular fast graph embedding libraries.

Secondly, I have proposed a novel method of combining the EMDE algorithm with neural networks to use in top-k recommendation. My experiments show that density estimators can be used with success for modeling user item spaces. A combination with sparse feed-forward network results in a highly accurate algorithm, which can additionally integrate multiple modalities with exceptional ease.

Furthermore, I have tackled the task of content-based recommendation. I have shown that centroid-based image retrieval combined with a neural model allows to increase quality and speed of unimodal image-based content recommendations. To verify this hypothesis, I proposed a centroid-based image retrieval approach. By extension, I have also tackled the task of cross-modal content recommendation. I proposed a novel algorithm for matching multimodal embedding spaces using trainable centroids in multidimensional representation space. I proved that his unified feature space allows to both increase the quality and speed of cross-modal content recommendations. The proposed algorithm reaches state-of-the-art results, without using the expensive attention mechanism.

Finally, I have presented the WildNLP framework. I have proven that increasing the robustness of models against some adversarial examples allows to strengthen the model against related adversarial examples. To validate this hypothesis, I have created the WildNLP framework which exemplifies the hypothesis on the textual modality.

In conclusion, all research objectives have been reached and all research hypotheses have been proven. As an additional advantage, this research finds usage in real-world production settings where multimodal data is used, making it suitable for multimodal recommender systems.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019.

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Taha Atahan Akyildiz, Amro Alabsi Aljundi, and Kamer Kaya. Gosh: Embedding big graphs on small hardware. In *49th International Conference on Parallel Processing (ICPP)*, 2020.

Salman Aslam. Twitter by the numbers: Stats, demographics & fun facts. 2021. URL `https://www.omnicoreagency.com/twitter-statistics/`.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

Jiyang Bai, Yuxiang Ren, and Jiawei Zhang. Ripple walk training: A subgraph-based training framework for large and deep graph neural network. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021a.

Jiyang Bai, Yuxiang Ren, and Jiawei Zhang. Measuring and sampling: A metric-guided subgraph learning framework for graph neural network. *International Journal of Intelligent Systems*, 2022.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *Proceedings of the Thirtieth International Joint Conference* *on Artificial Intelligence, IJCAI-21*. International Joint Conferences on Artificial Intelligence Organization, 2021b.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018.

Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. Technical report, 2020. URL `https://github.com/allenai/longformer`.

James Bennett and Stan Lanning. The Netflix Prize. In *Proceedings of KDD Cup and Workshop in conjunction with KDD*, 2007.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han. IMRAM: Iterative Matching with Recurrent Attention Memory for Cross-Modal Image-Text Retrieval. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020a.

Tianlang Chen and Jiebo Luo. Expressing objects just like words: Recurrent visual embedding for image-text matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020b.

Qingrong Cheng and Xiaodong Gu. Bridging multimedia heterogeneity gap via graph representation learning for cross-modal retrieval. *Neural Networks*, 134, 2021.

Ayushi Dalmia, Ganesh J, and Manish Gupta. Towards interpretation of node embeddings. In *Proceedings of the International World Wide Web Conference (WWW)*, 2018.

Michał Daniluk, Jacek Dabrowski, Barbara Rychalska, and Konrad Gołuchowski. Synerise at kdd cup 2021: Node classification in massive heterogeneous graphs. *Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) KDD Cup Open Graph Benchmark (OGB) Challenge*, 2021a. URL `https://ogb.stanford.edu/paper/kddcup2021/mag240m_SyneriseAI.pdf`.

Michał Daniluk, Barbara Rychalska, Konrad Gołuchowski, and Jacek Dąbrowski. Modeling multi-destination trips with sketch-based model. *14th ACM International Web Search and Data Mining Conference (WSDM) WebTour Workshop on Web Tourism*, 2021b. URL `http://ceur-ws.org/Vol-2855/challenge_short_3.pdf`.

Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential / session-based recommendation. *Fifteenth ACM Conference on Recommender Systems*, 2021.

Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. Taamr: Targeted adversarial attack against multimedia recommender systems. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020.

Haiwen Diao, Ying Zhang, Lin Ma, and Huchuan Lu. Similarity Reasoning and Filtration for Image-Text Matching. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Jacek Dąbrowski and Barbara Rychalska. Synerise Monad - Real-Time Multimodal Behavioral Modeling. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, 2022.

Jacek Dąbrowski, Barbara Rychalska, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Bąbel, Andrzej Michałowski, and Adam Jakubowski. An Efficient Manifold Density Estimator for All Recommendation Systems. In *Neural Information Processing: 28th International Conference, ICONIP 2021*, 2021.

Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*, 2017.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*, 2019.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc' Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, 2013.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries (DL)*, 1998.

Max Glockner, Vered Shwartz, and Yoav Goldberg. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2018.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12), 1992.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

Artur Gramacki. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 3319716875.

Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1), 1991.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2016.

Zhibin Guan, Kang Liu, Ma Yan, Xu Qian, and Tongkai Ji. Sequential dual attention: Coarse-to-fine-grained hierarchical generation for image captioning. *Symmetry*, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, Alessandro Micarelli, and Joeran Beel. Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation? In *The ACM Conference Series on Recommender Systems (RecSys) (Late-Breaking Results)*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Zhibin Hu, Yongsheng Luo, Jiong Lin, Yan Y. Yan, and Jian Chen. Multi-level visual-semantic alignments with relation-wise dual attention network for image and text matching. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19s*, 2019.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2019.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 2000.

Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

Chanwoo Jeong, Sion Jang, Eunjeong Park, and Sungchul Choi. A context-aware citation recommendation model with bert and graph convolutional networks. *Scientometrics*, 124(3): 1907–1922, 2020.

Jiangli Jiao, Xueying Zhang, Fenglian Li, and Yan Wang. A novel learning rate function and its application on the svd++ recommendation algorithm. *IEEE Access*, 8:14112–14122, 2019.

Rong Jin. Deep Learning at Alibaba. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 11–16, 2017.

HeeJae Jun, ByungSoo Ko, Youngjoon Kim, Insik Kim, and Jongtack Kim. Combination of multiple global descriptors for image retrieval. *ArXiv*, 2019. URL `http://arxiv.org/abs/1903.10663`.

Mohammad Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C. Berg, and Tamara L. Berg. Where to buy it: Matching street clothing photos in online shops. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, 2010.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, 2019.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, 2017.

Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual semantic reasoning for image-text matching. In *2019 IEEE/CVF International Conference on Computer Vision, (ICCV)*, 2019.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the International World Wide Web Conference (WWW)*, 2018.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference*, 2014.

Li Liu, Paul Fieguth, Xiaogang Wang, Matti Pietikäinen, and Dewen Hu. Evaluation of LBP and deep texture descriptors with a new robustness benchmark. In *Computer Vision – ECCV 2016*. Springer International Publishing, 2016a.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. Stamp: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of The International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b.

Linyuan Lu, Matus Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics reports*, 519(1):1–49, 2012.

Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Hao Luo, Wei Jiang, Youzhi Gu, Fuxu Liu, Xingyu Liao, Shenqi Lai, and Jianyang Gu. A strong baseline and batch normalization neck for deep person re-identification. *IEEE Transactions on Multimedia*, 22:2597–2609, 2020.

Michael W. Mahoney, Anirban Dasgupta, Jure Leskovec, and Kevin J. Lang. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. In *Internet Mathematics 6 (1)*, 2009.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3, 2000.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Brian McFee and Gert R. G. Lanckriet. Hypergraph models of playlist dialects. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*. ISMIR, 2012.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995. URL `http://www1.cs.columbia.edu/~vh/courses/LexicalSemantics/Ontologies/miller-wordnet95.pdf`.

Mordor Intelligence Research. Recommendation engine market - growth, trends, covid-19 impact, and forecasts (2021 - 2026). 2021. URL `https://www.mordorintelligence.com/industry-reports/recommendation-engine-market`.

Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender sysms. In *IEEE 11th International Conference on Data Mining*, 2011.

Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Trans. Interact. Intell. Syst.*, 2016.

Aleksandra Pawlicka, Marek Pawlicki, Rafał Kozik, and Ryszard S Choraś. A systematic review of recommender systems and their applications in cybersecurity. *Sensors*, 21(15):5248, 2021.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018.

B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, volume 2, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2), 2021.

Barbara Rychalska and Jacek Dabrowski. Synerise at sigir rakuten data challenge 2020: Efficient manifold density estimator for cross-modal retrieval. *The 43th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) eCom Workshop Challenge*, 2020. URL `https://sigir-ecom.github.io/ecom20DCPapers/SIGIR_eCom20_DC_paper_1.pdf`.

Barbara Rychalska, Dominika Basaj, Anna Wróblewska, and Przemyslaw Biecek. Does it care what you asked? understanding importance of verbs in deep learning QA system. In *Pro-*

ceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2018.

Barbara Rychalska, Dominika Basaj, Alicja Gosiewska, and Przemysław Biecek. Models in the wild: On corruption robustness of neural nlp systems. In Tom Gedeon, Kok Wai Wong, and Minho Lee, editors, *Neural Information Processing*. Springer International Publishing, 2019.

Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, Jacek Dąbrowski, and Przemysław Biecek. Cleora: A simple, strong and scalable graph embedding scheme. In Teddy Mantoro, Minho Lee, Media Anugerah Ayu, Kok Wai Wong, and Achmad Nizar Hidayanto, editors, *Neural Information Processing*, pages 338–352, Cham, 2021a. Springer International Publishing.

Barbara Rychalska, Mikołaj Wieczorek, and Jacek Dąbrowski. T-emde: Sketching-based global similarity for cross-modal retrieval. 2021b. URL `https://arxiv.org/pdf/2105.04242.pdf`.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18. AAAI Press, 2018.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61, 2015.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2018.

Wenchuan Shi, Liejun Wang, and Jiwei Qin. User embedding for rating prediction in svd++-based collaborative filtering. *Symmetry*, 12(1):121, 2020.

Brent Smith and Greg Linden. Two decades of recommender systems at Amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017.

Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *Proceedings of the International World Wide Web Conference (WWW)*, 2019.

Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35 (4):551–566, 1993.

Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Adagcn: Adaboosting graph convolutional networks into deep models. In *International Conference on Learning Representations (ICLR)*, 2021.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML)*. JMLR.org, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the International World Wide Web Conference (WWW)*, 2015.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2021.

Alok Tripathy, Katherine Yelick, and Aydın Buluç. Reducing communication in graph neural network training. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, 2018.

Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 30music listening and playlists dataset. In Pablo Castells, editor, *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.

Sijin Wang, Ruiping Wang, Ziwei Yao, Shiguang Shan, and Xilin Chen. Cross-modal scene graph matching for relationship-aware image-text retrieval. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, 2020a.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. 2020b. URL `http://arxiv.org/abs/2006.04768`.

Yaxiong Wang, Hao Yang, Xueming Qian, Lin Ma, Jing Lu, Biao Li, and Xin Fan. Position focused attention network for image-text matching. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*. Springer International Publishing, 2016a.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *2016 European Conference on Computer Vision (ECCV)*, 2016b.

Mikolaj Wieczorek, Andrzej Michalowski, Anna Wroblewska, and Jacek Dabrowski. A strong baseline for fashion retrieval with person re-identification models. In Haiqin Yang, Kitsuchart Pasupa, Andrew Chi-Sing Leung, James T. Kwok, Jonathan H. Chan, and Irwin King, editors, *Neural Information Processing*, 2020.

Mikołaj Wieczorek, Barbara Rychalska, and Jacek Dąbrowski. On the unreasonable effectiveness of centroids in image retrieval. In *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part IV*, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019a.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019b.

Zhengzheng Xian, Qiliang Li, Gai Li, and Lei Li. New collaborative filtering algorithms based on svd++ and differential privacy. *Mathematical Problems in Engineering*, 2017.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR Conference Track Proceedings*, 2015.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12. Association for Computing Machinery, 2012.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Trans. Assoc. Comput. Linguistics*, 2014.

Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Tagnn: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20. Association for Computing Machinery, 2020.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple but hard-to-beat baseline for session-based recommendations. 2018. URL `http://arxiv.org/abs/1808.05163`.

Ye Yuan, Wuyang Chen, Yang Yang, and Zhangyang Wang. In defense of the triplet loss again: Learning robust person re-identification with fast approximated triplet loss and label distillation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. nowplaying music dataset: Extracting listening behavior from twitter. WISMM '14, 2014.

Qi Zhang, Zhen Lei, Zhaoxiang Zhang, and Stan Z. Li. Context-aware attention network for image-text retrieval. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020b.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.