ON CONVOLUTIONAL NEURAL NETWORKS FOR KNOWLEDGE GRAPH EMBEDDING AND COMPLETION

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by

Chen Shen
August, 2020

Examining Committee Members:

Eduard Dragut, Advisory Chair, Dept. of Computer and Information Sciences
Yuhong Guo, School of Computer Science, Carleton University
Kai Zhang, Dept. of Computer and Information Sciences
Justin Shi, Dept. of Computer and Information Sciences
Weiyi Meng, External Member, Dept. of Computer Science, Binghamton University

ABSTRACT

Data plays the key role in almost every field of computer sciences, including knowledge graph field. The type of data varies across fields. For example, the data type of knowledge graph field is knowledge triples, while it is visual data like images and videos in computer vision field, and textual data like articles and news in natural language processing field. Data could not be utilized directly by machine learning models, thus data representation learning and feature design for various types of data are two critical tasks in many computer sciences fields. Researchers develop various models and frameworks to learn and extract features, and aim to represent information in defined embedding spaces. The classic models usually embed the data in a low-dimensional space, while neural network models are able to generate more meaningful and complex high-dimensional deep features in recent years.

In knowledge graph field, almost every approach represent entities and relations in a low-dimensional space, because there are too many knowledge and triples in real-world. Recently a few approaches apply neural networks on knowledge graph learning. However, these models are only able to capture local and shallow features. We observe the following three important issues with the development of feature learning with neural networks. On one side, neural networks are not black boxes that work well in every case without specific design. There is still a lot of work to do about how to design and propose more powerful and robust neural networks for different types of data. On the other side, more studies about utilizing these representations and features in many applications are necessary. What's more, traditional representations and features work better in some domains, while deep representations and features perform better on other domains. Transfer learning is introduced to bridge the gap between domains and adapt various type of features for many tasks.

In this dissertation, we aim to solve the above issues. For knowledge graph learning

task, we propose a few important observations both theoretically and practically for current knowledge graph learning approaches, especially for knowledge graph learning based on Convolutional Neural Networks. Besides the work in knowledge graph field, we not only develop different types of feature and representation learning frameworks for various data types, but also develop effective transfer learning algorithm to utilize the features and representations. The obtained features and representations by neural networks are utilized successfully in multiple fields. Firstly, we analyze the current issues on knowledge graph learning models, and present eight observations for existing knowledge graph embedding approaches, especially for approaches based on Convolutional Neural Networks. Secondly, we proposed a novel unsupervised heterogeneous domain adaptation framework that could deal with features in various types. Multimedia features are able to be adapted, and the proposed algorithm could bridge the representation gap between the source and target domains. Thirdly, we propose a novel framework to learn and embed user comments and online news data in unit of sessions. We predict the article of interest for users with deep neural networks and attention models. Lastly, we design and analyze a large number of features to represent dynamics of user comments and news article. The features span a broad spectrum of facets including news article and comment contents, temporal dynamics, sentiment/linguistic features, and user behaviors. Our main insight is that the early dynamics from user comments contribute the most to an accurate prediction, while news article specific factors have surprisingly little influence.

ACKNOWLEDGMENTS

During my seven years of Ph.D. study, I would like to thank a lot of people that support and help me for the research, study and life.

I give my deepest thanks to my two advisors Dr. Eduard Draguts and Dr. Yuhong Guo. Dr. Eduard Dragut has much insight in many fields. He guides me for how to do research and how to discover news idea. He always encourage me to think more about possible improvement and extensions on existing research, which bring me a new way of thinking. Dr. Yuhong Guo teaches me so much about machine learning and attitude of researching. She helps me a lot on both research and life, and opens a new chapter for my Ph.D. career.

I would like to thank my committee members, Dr. Kai Zhang, Dr. Justin Shi and Dr. Weiyi Meng for their valuable time and advice on my dissertation. Dr. Kai Zhang kindly acts as my instructor for the knowledge graph embedding project. He teaches me the essential parts of deep learning. Dr. Justin Shi always encourages me and gives me advice if I encounter bottleneck on my research. I believe I have the best committee members in the university and would like to thank them so much.

I would like to thank Dr. Zoran Obradovic, Dr. Slobodan Vucetic, Dr. Bo Ji, Dr. Longin Latecki and Dr. Jim Korsh for their great courses and suggestion on research.

I would like to thank my lab-mates and many other students for their advice and help these years.

Finally, thanks very much for my wife Lihong, my son and my parents, for their love and support all the time.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

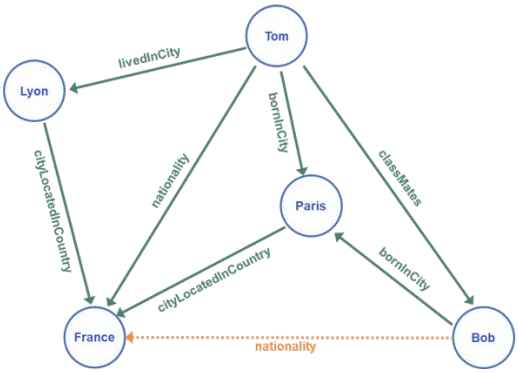
1.1 Feature and Representation Learning

Data plays a key role in almost every field of computer sciences. Researchers develop a lot of techniques to detect or generate features for the data, and extract suitable representations for the data according to the tasks. These techniques are usually developed just for specific data or task, thus they may not work for other fields.

There are many types of data in computer sciences fields. For example, as shown in Figure 1.1, visual data in computer vision field is in the format of images and videos, textual data in natural language processing field is in the format of articles and sentences. The knowledge data in knowledge graph field is abstract, and it is in triple and graph format. There are also high-level data that we try to represent, i.e. user's interest in recommendation system.

Feature and representation learning techniques could be divided into supervised and unsupervised groups. In unsupervised setting, models are learned based on unlabeled data. These techniques aim to use low-dimensional features to capture the hidden relation and similarity of the input data. K-means clustering [42] and Principal component analysis[125] are well-known techniques of unsupervised setting. In supervised setting, labels are input along with data to the model. The supervised learning techniques utilize label information as feedback to generate better features and representations. In these years, Neural Networks [33] attract a lot of attention and are widely used in many fields. In most cases, the feature learning of Neural

Temple University is an R1 institution that serves more than 38,000 students and places in the top 55 among public colleges and Universities by the U.S. News ranking.



	Movie 1	Movie 2	Movie 3
Tom	1		1
Lyon		1	1
Bob	1		
Jerry	1	1	?

Figure 1.1: Data types in different fields

Networks is supervised, while there are also unsupervised methods.

1.2 Knowledge Graph

One of the key objectives of Artificial Intelligence is to utilize human knowledge to machine learning tasks. Human knowledge represents the information that human-beings developed from the real-world, and it's a high-level information that represent the relations among real-world entities. Knowledge from real-world could not be understood by machine or algorithms directly. People use graph as the structure to represent knowledge.

Knowledge Graph is knowledge in graph format. In knowledge graph, entities in real-world are represented by nodes while relations are represented by edges. Other properties of the entities are stored in attributes of the nodes. In real world, there are

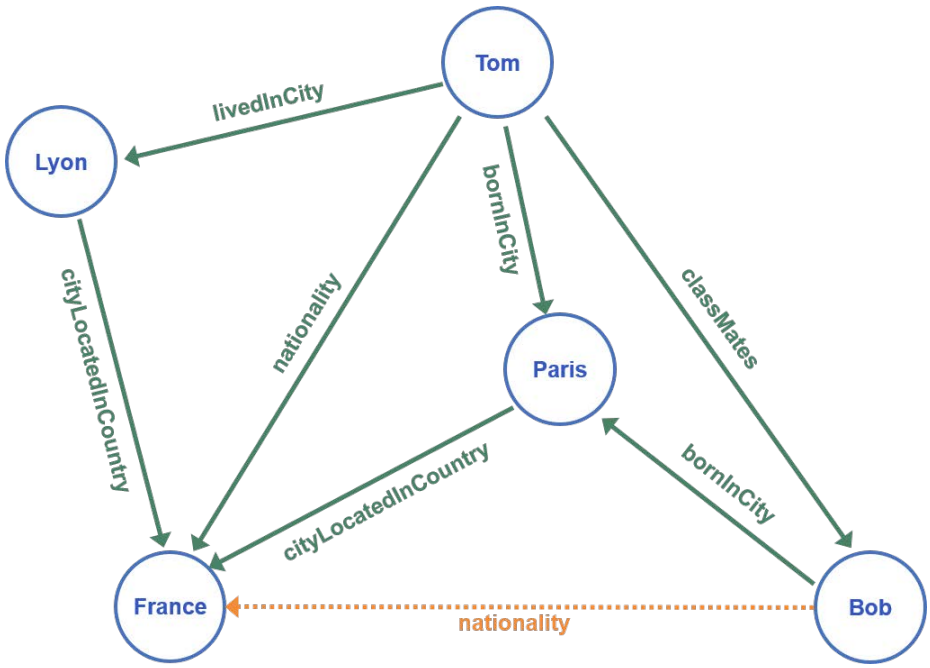


Figure 1.2: An example of missing link prediction

different kinds of knowledge graph bases such as WordNet [65], Google Knowledge Graph and DBpedia [53]. Knowledge graph learning aims to learn the representations of nodes and edges in knowledge graph, and apply the representations for task like missing link prediction.

Missing link prediction is an important task in knowledge graph learning. Usually, knowledge graphs suffer from incompleteness. People try to exploit new triplets based on the existing incomplete graph: (1) given a head/tail and one kind of relationship ℓ, find the tail/head in the entity set; (2) given one head h and one tail t, find their relationship ℓ. As shown in Figure 1.2, if two entities (Tom and Bob) have quite similar structure information in a knowledge graph, we can infer that there should be one missing link "Bob, nationality, France" base on other information in this knowledge graph. Various models [76] are proposed to learn knowledge graph representations and apply them on missing link prediction task. Recently, a few approaches adopt neural network in knowledge graph learning.

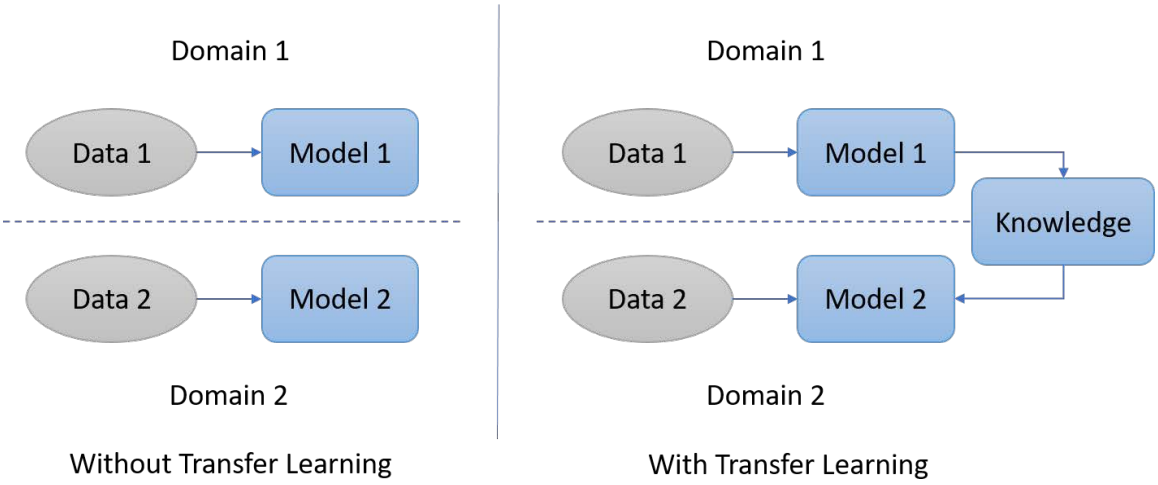


Figure 1.3: Transfer Learning

1.3 Transfer Learning

Labels of data are essentially important for both supervised and semi-supervised approaches. There are many scenarios and tasks for real-world data. Usually feature learning models trained in one domain/task could not be used in other domains/tasks directly. The insufficient labeled data in new domain is an obvious bottle-neck for feature learning models. Transfer learning is an important tool in machine learning to apply knowledge from one problem to a different but related problem [83].

As shown in Figure 1.3, without the help of transfer learning, data and model in domain 1 could not be used by data and model in domain 2. Transfer learning is able to take the knowledge and information learned in domain 1 and adapt them to the model in domain 2.

Domain adaptation is a subclass of transfer learning. It aims to transfer label knowledge from a source domain to a target domain for a same task. Domain adaptation has attracted a lot of attention recently in computer vision [76] and many other fields[127]. Much existing domain adaptation work assumes a homogeneous cross-domain feature space, which hinders the applicability of domain adaptation in

many real world scenarios where source domain with different input feature space can be readily available for the prediction task in a given target domain. Heterogeneous domain adaptation techniques, which tackle domain adaptation problems with different cross-domain feature spaces, hence are in high demand.

The major challenge for heterogeneous domain adaptation lies in bridging the disjoint cross-domain features spaces. Supervised methods tackle this problem by building cross-domain connections based on the existence of a small set of labeled instances in the target domain. The performance of such methods however is highly restricted by the scarcity of the labeled instances in the target domain. Semi-supervised heterogeneous domain adaptation approaches hence further exploit the unlabeled instances in the target domain to alleviate this restriction and improve the learning of feature transformation or classifiers [110, 125, 128, 134]. These method still depend on the existence of labeled target instances. We will present an unsupervised heterogeneous domain adaptation in Chapter 2, which doesn't require any labeled data in target domain.

1.4 Neural Networks

Artificial Neural network, also called Neural Network in computer science field, is a machine learning method inspired by biological neural networks that constitute brain [149]. Neural Networks have become dominant in various computer sciences fields for feature learning. Compared with traditional methods that require manually designed features, neural networks automatically learn and generate high-dimensional features from input, and may produce better representations.

There are many types of neural networks. Among them, Convolutional Neural Networks and Recurrent Neural Networks are two most popular types of neural networks that proved to be powerful in various fields. Convolutional Neural Networks are based on shared weights architecture. They have become the standard feature

learning models in computer vision field. Recurrent Neural Networks exhibit temporal dynamic sequence behavior with directed graph nodes, and are extremely useful for data with sequence information.

One challenge of neural networks is that the choice of most suitable model varies based on many factors, i.e. data types, data properties, tasks and applications. Neural networks are not black boxes that work well in every case without specific design. There is still a lot of work to do about how to design and propose more powerful and robust neural networks for different types of data. More studies about utilizing these representations and features in many applications are necessary.

1.5 Contribution

In this dissertation, we develop various feature and representation learning methods that can effectively represent information for data in various fields. We also develop a few frameworks for read-world applications, i.e. news recommendations, article comment volume prediction, knowledge graph completion, and we utilized both features designed manually and features learned in neural networks.

More specifically, our contributions are:

Firstly, in Chapter 2, we study knowledge graph representation learning problem. In this chapter, we present a few observations for knowledge graph embedding and completion approaches, especially for approaches base on CNN. We analyze the limitations and issues in current knowledge graph approaches, and present eight observations about knowledge graph representation learning and completion both theoretically and practically. These observations infer that the convolutional operations among different dimensions of embedding vectors are probably redundant.

Secondly, in Chapter 3, we proposed a novel sparse feature transformation method for unsupervised heterogeneous domain adaptation. The method transforms the source domain features into the target domain feature space by matching the paral-

lel instances and aligning the empirical second moments of the transformed source feature distribution and target domain feature distribution. We further exploit two types of sparsity inducing norms to regularize the linear transformation model. We develop an ADMM based optimization algorithm to solve the induced problem. The results show that our proposed model is able to adapt features from various and disjoint domains and fields.

Thirdly, in Chapter 4, we propose a deep neural network to learn and capture the unique temporal user commenting dynamics with the following capabilities: (1) we uses RNN-based models to capture a user's sequential behavior; (2)our model includes an attention mechanism to model a user's uneven commenting actions and another to model the importance of a reader's historical actions; and it includes a recency-based regularization penalize the prediction scores of fresh articles (smaller recency)less. The representation of sessions is able to captures temporal dynamics observed in news recommendation, and our model output-performs all baselines in real-world news dataset.

Lastly, in Chapter 5, we study the features for user comments and news articles. We compile and learn five groups of features: topic, article, user comment, news factor and miscellaneous, both old and novel. Some of these features are learned by neural networks. Our main insight is that the early dynamics of user comments contribute the most to an accurate prediction, while news article specific factors have surprisingly little influence. Furthermore, we show that the early arrival rate of comments is the best indicator of the eventual number of comments. We made a few conclusions about the prediction of comment volume of news articles. This framework is a representative of utilizing traditional and deep features for real-world application.

In Chapter 6, we discuss some directions that worth exploring in future works.

CHAPTER 2

KNOWLEDGE GRAPH EMBEDDING LEARNING WITH CONVOLUTIONAL NEURAL NETWORK AND OBSERVATIONS

In this chapter, we research on knowledge graph learning with neural networks and present a few observations for current techniques on knowledge graph learning. Knowledge graphs are structured knowledge in graph format. Existing knowledge graphs are only a very small subset of real-world knowledge, thus they suffer from incompleteness. Recently knowledge graph embedding and completion approaches based on CNN achieved impressive results on entity link prediction task. However, these approaches have a few concerns theoretically. The convolution among dimensions is unintuitive, and the models are local and shallow. In this chapter, we presents eight observations for existing knowledge graph embedding approaches, especially for approaches based on CNN. Based on the observations, we propose a compact but effective CNN model with convolutional filter width of only 1. These observations infer that the convolutional operations among different dimensions of embedding vectors are probably redundant.

2.1 Introduction

Knowledge graphs (KG) store real-world information as directed multi-relational structured graphs. In KG, entities are graph nodes, and relationships are different kinds of directed edges. Each piece of knowledge is represented as (head, relationship,

tail) or (h, ℓ, t). For example, "Donald Trump is a Politician of USA" will be stored as (Donald Trump, isPoliticianOf, USA) where "Donald Trump" is the head entity, "isPoliticianOf" is the relationship and "US" is the tail entity. In real world, there are different kinds of knowledge graph bases such as WordNet [65], Google Knowledge Graph and DBpedia [53]. WordNet is a large lexical database of English, in which words are grouped into cognitive synonyms (synsets) and these synsets are interlinked with different relations. Google Knowledge Graph is a system that Google launched to understand facts about people, places and things and how they are connected. DBpedia [53] extracts information from Wikipedia as a structured knowledge base.

Usually, knowledge graphs suffer from incompleteness. People try to exploit new triplets based on the existing incomplete graph: (1) given a head/tail and one kind of relationship ℓ, find the tail/head in the entity set; (2) given one head h and one tail t, find their relationship ℓ. Various models [76] are proposed to learn knowledge graph representations and solve these problems. Among them a group of methods are mainly based on tensor factorization [75, 106, 77]. Another group of models such as TransE [13], TransH [121], TransR [58] and TransA [40] learn low-dimensional representations for entities and relationships by treating relationships as translations from heads to tails. The drawback of these models is that the translation structure assumption between entities and relationships is simple but in reality the connections between entities and relationships might be much more complex.

Based on the translation models, people try to learn better representations to get more accurate predictions. In [57], PTransE is proposed to improve TransE. This model considers "relation paths" instead of relationship as translations between entities. Similarly, in [28], the authors compose several relationships together as new relationships, and then new triplets are added to the training set to learn knowledge graph embedding. In [130], a manifold based embedding principle is proposed.

Models based on Neural Networks are also proposed to learn knowledge graph em-

bedding. Different from translation based models, the connections between entities and relationships are learned in neural networks. In [100], a neural tensor network (NTN) is proposed to learn the heads and tails over different relationships. In [95], ProjE is proposed, which uses an entity representation combination operation and a non-linear mapping to calculate the score for a triplet (h, ℓ, r).

In recent years, Convolutional Neural Networks (CNN) is utilized in knowledge graph embedding [21]. The 2-D convlutional operation over embbeddings could be looked as multi-channel linear combination of entities and relations' embedding. Models based on CNN are able to generate multiple combinations of same input with the help of channels. However, CNN in knowledge graph is local and shallow compared with deep CNN in other fields, and is doubt to be intuitive [10].

In this chapter, we present a few observations for knowledge graph embedding and completion approaches, especially for approaches base on CNN. We analyze and present the limitations and issues in knowledge graph approaches. We argue that the convolutional operation among different dimensions of embedding vectors is redundant.

In summary, we have the following contributions: We make the following contributions in this work:

- We we presents eight observations for existing knowledge graph embedding approaches, especially for approaches based on CNN. These observations contain both theoretical and practical analysis on the issues of knowledge graph embedding models. And we summarize and present a few conclusions that may benefit future model development.

- We suggest that CNN base on linear weighted sum could be simplified. The reshape and convolution among dimensions could be removed from the model without scarifying the performance.

2.2 Related Work

2.2.1 Tensor Factorization Based Embedding Learning

In RESCAL [75], given the knowledge graph, for each relation ℓ, a binary matrix R_ℓ is created for entity pairs. If the i-th entity e_i and j-th entity e_j are connected with ℓ, (e_i, ℓ, e_j) hold and $R_\ell(i,j) = 1$, otherwise $R_\ell(i,j) = 0$. Finally a three-dimensional binary tensor R with size $n \times n \times m$ is created for all the relationships, where n and m are the entity and relationship number respectively. After that, entity and relationships embedding are learned by using tensor factorization. A new tensor will be reconstructed based on the learned embedding.

2.2.2 Translation Based Embedding Learning

In TransE [13], entities and relationships are represented as low-dimensional embeddings. If a triplet (h, ℓ, t) holds, the tail entity t can be represented as the head entity e with a simple translation by using the relationship ℓ: ideally, $h + \ell = t$. A positive triplet (h, ℓ, t) will have a small distance between $h + \ell$ and t while a negative triplet (h', ℓ, t') will have a large distance between $h' + \ell$ and t'. Pairwise ranking method is applied to learn the entity and relationship embedding.

Similarly, TransH [121] also uses relationship translation to connect head and tail entities. Different from TransE, in TransH, there is one hyper-plane for every relationship. Given one relationship, head and tail will be projected to the hyper-plane first as relationship-specified embedding, after that translation will be performed on this hyper-plane.

In TransR, entities and relationships are represented in two different spaces. For every relationship, a mapping function is learned to map the head and tail from entity space to relationship space. Similar to TransE and TransH, a translation is finally performed to connect the new head and tail in the relationship space.

2.2.3 Neural Network Based Knowledge Graph Learning

Besides tensor factorization and translation models, algorithms based on neural networks for knowledge graph learning are also proposed. Neural networks are used to learn more complex translations to connect head h, relation ℓ and tail t. In [100], a neural tensor network structure (NTN) is proposed to reason over relationships between one pair of head and tail. Given a triplet (h, ℓ, t), the neural tensor network uses a bilinear tensor operation to compute a confidence score. Embedding and NTN structure are learned simultaneously. In [95], a two-layer neural network is applied on triplet $(\mathbf{e}, \ell)$. List-wise and point-wise ranking methods are used in their model. ConvE [21] reshapes head entity and relation embedding vectors as 2D matrix, which is similar as the idea of CNN on images. It applies 2D convlutional filters over the reshaped matrix and multiply the concatenated output with the embedding matrix of all candidate tail entities. Even if the results is impressive, there is almost space information on the reshaped 2D embedding matrix, which is different with CNN in visual field essentially. ConvKB [72] concatenates the pretrained embedding of TransE and obtains a $3 * dim$ matrix. A convolution filter is applied over the matrix and the output is projected to 1-dimensional score. However, this model relies on output of TransE and uses a quite small learning rate, while the performance is close to TransE. And the best parameter for convolutional filter is 1. Thus it is more like a fine-tuning over distance model. CapsE[73] replace the convolution layer with capsule layer, which add non-linearly compared with linear CNN. However, this model requires embedding from ConvKB as input. The nested requirement of pre-training embedding makes it hard to evaluate the model solely. HypER [9] transforms relation embedding vector as filters over head entity embedding vector, which replace the linear CNN operation with non-linear product. However, this model is essentially equivalent to tensor factorization model, and there is no external weight for convolutional filters. It's based on local tenor product rather than convolution, thus it's out of the scope

in our analysis of linear CNN model.

2.3 Observations for Knowledge Graph Embedding Models Based on CNN

2.3.1 Limitations of CNN in Knowledge Graph

Observation 1 *The embedding space of Convolutional Neural Network for knowledge graph is more random than other fields.*

Convolutional Neural Network (CNN) has been widely used in various fields and is proved to be powerful dealing with various features, i.e. visual features, textual features and wave features. CNN is based on convolutional operations over fixed data, which extracts spatial information among dimensions of input. All of the input data in these fields are fixed and meaningful, like images, sentences, and waves. There are originally spatial or sequence information among the input data. For example, nearby pixels in images contains spatial information of objects and edges, nearby words in sentence make up phrases or grammar patterns. Frameworks in these fields learn the weights of CNN model according to fixed input data, and make predictions or classifications. However, knowledge graph embedding is a harder topic that both embedding of input and weights are unknown, and the only information for the input is triples. The embedding space of input knowledge data is totally random and there is no spatial information among dimensions in feature space. Though the results achieved by knowledge graph approaches based on CNN are impressive, i.e. [21], the idea and theory are doubted unintuitive [10].

Observation 2 *The way to generate input tensor of neural network needs to be improved.*

Knowledge graph embedding approaches based on CNN got trouble with generating

the matrix as input to CNN. [72] and [73] concatenate the embeding vectors of (s, r, o) to a $3 \times d_e$ matrix. [21] concatenate the embedding vectors alternately. All of the approaches are trying to make the CNN model of knowledge graph closer to CNN for images. But the reshape and concatenate ideas could not solve the concern that there is no spatial information among dimensions of embeddings as they are totally random.

Observation 3 *The convoluted representations are local and shallow.*

Knowledge graph approaches based on CNN usually use a small convolution filter and the network is shallow (mostly one layer). [72] claimed to use $3 \times k$ convolution filter, but stated $k = 1$ in the experiments part. That means the model is essentially a weighted sum of each dimension of (s, r, o) rather than convolution neural network. It has been proven both with [72] and our implementation that the performance of such concatenation idea will decrease as convolution filter width k increases. Alternating concatenation idea like [21] uses a small 3×3 convolution filer and we found smaller size still works. A small convolution filer with only one layer means there is little local computations among dimensions of input data, and that's why approaches have to reshape or alternate input data as stopgaps to enforce dimension interaction.

Based on the above three observations, we believe that the approaches based on classic convolutional neural network are not same with the deep CNN in other fields theoretically, and these approaches are actually local linear weighted sum of embeddings. We argue that a similar but simpler 1-D CNN with convolutional kernel width of 1 could achieve close or even better results, which means the convolution operation among dimensions are redundant.

2.3.2 1-1 and 1-N Scoring

Observation 4 *1-N scoring doesn't speed up the training time compared with 1-1 scoring.*

2.3.3 Negative Sampling and Regularization: Technique Updates in Knowledge Graph Embedding

Observation 5 *A few techniques to improve Non-Neural Network approaches don't work in Neural Network approaches*

There are only positive triplets in knowledge graph dataset. Negative triplets are constructed from positive triples. Approaches usually randomly corrupt the head/tail entity of a positive triplet and replace it by other entities from entity list. The Bernoulli negative sampling is firstly proposed by [121] to reduce false negative labels. The head entity will be given more chance be replaced if the positive triple is one-to-many, and the tail entity will be given more chance be replaced if the positive triple is many-to-one. This technique is utilized and proved to be effective in many approaches. However, this technique is not able to improve the results for knowledge graph embedding based on neural network. We tried 5 times in our approaches and the difference is too small.

Another technique that fails to work for approaches based on neural network is regularization on embeddings. The parameters of knowledge graph embedding models are randomly initialized embedding E and model weights W while the output is just a score $f(W, E)$ in range $[0, 1]$ for each triplet. There will be many trivial solutions if there is not enough regularization (constraint). Most approaches before the trend of neural network in knowledge graph embedding add $l-1$ or $l-2$ regularization on the embedding matrix of entities and relations, i.e. [13] [121] [58] [135]. The results will be unstable or even meaningless without such constrain. However, regularization on embeddings doesn't work on knowledge graph embedding models based neural networks. The results will be much worse if the constrain on embedding is added.

There are currently two popular ways to avoid the over-fitting problem in knowledge graph embeding with neural network. On way is adding regularization loss for weights of neural network to the loss function. Another way is adding batch normal-

ization layers and dropout layers, which has been proved to be powerful substations of regularization. Dropout acts as essential role in recent tenser factorization method [10], in which 3-D tensor requires strong regularization to be represented by three tensors.

2.3.4 Label Smoothing Regularization

Observation 6 *Label smoothing in currently approaches is not correct theoretically, while the wrong formula performs better than the "correct" one.*

Label smoothing regularization is firstly used in Knowledge graph embedding field by [21] to make the model robust. The predictions for all candidate entities are looked as multi-class classification problem. According to the released code, the label smoothing over the prediction vector $\boldsymbol{y}$ of (s, r) pair is:

$$\boldsymbol{y}_{ls} = (1 - \alpha)\boldsymbol{y} + 1/K \tag{2.3-1}$$

where each element y_i of $\boldsymbol{y}$ is the prediction of one (s, r, o_i) triple, $o_i \in \mathcal{E}$ is one tail entity among all entity candidates. α is the label smoothing parameter and K is the number of candidate entities. There are two mistakes in this label smoothing. Firstly, the label smoothing for one-hot encoding $\boldsymbol{y}$ should be

$$\boldsymbol{y}_{ls} = (1 - \alpha)\boldsymbol{y} + \alpha/K \tag{2.3-2}$$

in which the sum of smoothed labels should still equals 1. A label smoothing parameter of $\alpha = 0.0$ (no label smoothing) adds $1/K$ to all predictions according to Eq. 2.3-1. The conclusions made in [21] that label smoothing of 0.1 is better than 0.0 and best choice of α need to be rechecked. Many approaches that follow [21] or its released code made the same mistake, i.e. [9] and [10]. The performance of these approaches drops if this mistake is fixed on authors' released codes. Secondly, even

if Eq. 2.3-1 is fixed with Eq. 2.3-2, the label smoothing has a concern theoretically. The score of each candidate triple (s, r, o_i) is treated as the prediction of one class, and there are K classes in total. There are one or more candidate triples that are positive for each (s, r) pair, and the prediction is a multi-label classification. Eq.2.3-2 is derived for one-hot label, which means that there is only one label equals to 1 in K classes and other labels are 0. Both Eq. 2.3-1 and 2.3-2 have concerns in theory for knowledge graph embedding. We expect some approaches will derive correct label smoothing formula for knowledge graph embedding in future.

In the experiments, we found that the performance of model with Eq. 2.3-1 is better than same model without Eq. 2.3-1 or with Eq. 2.3-2 in both published approaches and our proposed model. We have two possible explanations. Firstly, even if Eq. 2.3-1 is not theoretically correct, the partially correct label smoothing is still beneficial for knowledge graph embedding. And Eq. 2.3-2 also has theoretical issue. Secondly, the data of knowledge graph is disjoint with other fields. Each image in visual filed and text in NLP field in the training data has specify label for the class, i.e. is a dog or not, is positive or not, while knowledge graph triplets have much more uncertainty and have different negative generation methods. The negative triplets are generated by replacing head or tail entity in positive triplet with random (1-1 scoring) or all other (1-N) scoring entities. The positive triples in the validation and test set are negative triples in the training set, and the whole dataset is only a small portion of read world knowledge graph. Many negative triples generated during training step are actually false negative ones. Eq. 2.3-1 adds $1/K$ to all negative labels, while Eq. 2.3-2 adds α/K. Considering that the label smoothing parameter is usually a small value, i.e. 0.1, Eq. 2.3-1 assigns $1/\alpha$ larger values for negative labels, which is more suitable for the case that there are many false negative labels.

2.3.5 Evaluation

Observation 7 *Models that fails to work or overfitting may achieve good results with current popular evaluation.*

Given a head entity - relation pair (s, r) or tail entity - relation pair (o, r), link prediction aims to rank the candidate entities so that the ground-truth tail entity or head entity is ranked in topper position. Some approaches with 1-1 scoring place the score of the ground truth triple (positive data) at the first position of the prediction list or before negative triples [72] [73] [105], then they sort the prediction list by scores and output the rank of ground truth triple. This type of evaluation is not accurate if predictions of multiple entities are with a same best scores, for example 1.0. Assume there is a simple model that predict all candidate triples with score of 1, then the ground truth triple will always rank first because it's already there before sorting. This situation occurs if the model is over-fitting, especially when the last layer of prediction is sigmoid or the constraints are not enough. Approaches based on 1-N partially solve this concern but the output is still not accurate for situation of multiple ones. Denote the ground truth value as p_{gt} and prediction vector as $\boldsymbol{p}$. The solution is replacing sorting predictions by output $rank = count(p_i \geq p_{gt}),\ p_i \in \boldsymbol{p}$. This evaluation is strict evaluation that output the worst rank if multiple best scores occurs. If there are 10 predictions of 1.0 among all candidate entity predictions, the classic evaluation will output rank of 1 while this strict evaluation will output rank of 10. Prediction with all ones will get last rank instead of first rank. We recommend to use this strict evaluation for validation (parameter selection) as well as testing. Robust model with will perform better than over-fitting ones.

2.3.6 Pre-trained Embeddings

Observation 8 *The performance of approaches with pre-trained embeddings is not enough to prove the advancement of their models.*

Some approaches are based on pretrained entity and relation embeddings generated from another approach. For example, ConvKB[72] relies embedding of TransE as input, while CapsuleE[73] relies ConvKB as input. The TransE used in [72] is a variant of TransE named STransE [74], which already performs well on Mean Rank and Hits@10, while doesn't perform well on Hits@1 and MRR. ConvKB slightly improves mean reciprocal rank (MRR) of STransE from 0.226 to 0.248 and Hits@10 from 50.1% to 52.5% on WN18RR dataset. The limitation of STransE that Hits@1 is much worse than other algorithms is not solved. The CNN model in ConvKB only fine-tunes the pretrained embeddings with a small learning rate 0.0005, and it works much worse if it's randomly initialized.

2.4 Conclusion

In this chapter we presents eight observations for existing knowledge graph embedding approaches, especially for approaches based on CNN. These observations contain both theoretical and practical analysis on the issues of knowledge graph embedding models. And we summarize and present a few conclusions that may benefit future model development.

CHAPTER 3

UNSUPERVISED DOMAIN ADAPTATION FOR MULTIMEDIA FEATURE TRANSFORMATION

In this chapter we research on domain adaptation, which is a subclass of transfer learning, and propose a novel unsupervised transfer learning algorithm for muti-media features. Heterogeneous domain adaptation (HDA), which aims to adapt information across domains with different input feature spaces, has attracted a lot of attention recently. However, many existing HDA approaches rely on labeled data in the target domain, which is either scarce or even absent in many tasks. In this chapter, we propose a novel unsupervised heterogeneous domain adaptation approach to bridge the representation gap between the source and target domains. The proposed method learns a sparse feature transformation function based on the data in both the source and target domains and a small number of existing parallel instances. The learning problem is formulated as a sparsity regularized optimization problem and an ADMM algorithm is developed to solve it. We conduct experiments on several real-world domain adaptation datasets and the experimental results validate the advantages of the proposed method over existing unsupervised heterogeneous domain adaptation approaches.

3.1 Introduction

For supervised learning, labels of instances play a key role for classification model training with majority of approaches. However, for many real-world problems, it is

difficult or expensive to collect a sufficient amount of labeled data. Domain adaptation (DA), which aims to transfer label knowledge from a source domain to a target domain, hence has attracted a lot of attention recently in computer vision [78] and many other fields [131].

Much existing DA work assumes a homogeneous cross-domain feature space, which hinders the applicability of domain adaptation in many real world scenarios where a source domain with different input feature space can be readily available for the prediction task in a given target domain. For example, for image classification in a target domain, a source domain can have labeled images across the same set of classes but with different resolutions, which leads to different dimensions of codebook and hence feature space; labeled text data that describe a set of object classes can be a natural heterogeneous source domain for images from the same set of object classes. Heterogeneous domain adaptation (HDA) techniques, which tackle domain adaptation problems with different cross-domain feature spaces, hence are in high demand.

The major challenge for HDA lies in bridging the disjoint cross-domain features spaces. Supervised HDA methods tackle this problem by building cross-domain connections based on the existence of a small set of labeled instances in the target domain. They utilize the labeled data to learn a feature mapping function from one domain to another [51, 37, 143], or map the feature spaces of both domains into a common subspace [25, 103]. The performance of such methods however is highly restricted by the scarcity of the labeled instances in the target domain. A number of semi-supervised HDA approaches hence further exploit the unlabeled instances in the target domain to alleviate this restriction and improve the learning of feature transformation or classifiers [113, 129, 132, 138]. Some semi-supervised HDA methods even utilize parallel unlabeled instances to learn cross-domain representations [81, 131]. These methods however still depend on the existence of labeled target instances. A few unsupervised

HDA approaches overcome this dependence limitation on labeled target data by learning a common latent correlation subspace based only on parallel instances [34, 139]. However, these existing methods typically require a large number of parallel instances to achieve reasonable performance.

In this chapter, we propose a novel feature transformation method to tackle unsupervised heterogeneous domain adaptation by assuming the existence of a small number of parallel instances. The method uses a linear function to transform the source domain features into the target domain features to match the parallel instances, while minimizing the cross domain distribution divergence by aligning the transformed source domain covariance matrix with the target domain covariance matrix. Under the assumption that only a small number of source domain features are needed to induce a target domain feature, we further exploit two types of sparsity inducing norms to regularize the linear transformation model. We formulate this unsupervised HDA problem as a minimization problem over a sparsity regularized quartic function and develop an alternating direction method of multipliers (ADMM) to solve it efficiently. Experiments are conducted on a few heterogeneous domain adaptation datasets for image classification. The experimental results show that the proposed method outperforms existing unsupervised heterogeneous domain adaptation approaches and achieves promising results even when there are only a very small number of parallel instances.

3.2 Related Work

In this section, we briefly review the related groups of DA approaches, including unsupervised domain adaptation methods, (semi-)supervised HDA methods, and unsupervised HDA methods.

3.2.1 Unsupervised Domain Adaptation

Unsupervised DA aims to exploit the labeled data in a source domain to assist a target domain that has no labeled instances at all. Many unsupervised DA techniques have been developed in computer vision field. The work in [30] addresses cross-domain object recognition problems. It constructs and computes the geodesic flow kernel of infinite subspaces between the source and target domains to overcome the domain shift problem. [27] propose a visual domain adaptation approach for image classification that uses a linear mapping to align subspaces across domains. [62] introduce a transfer joint matching model that considers both feature matching and instance reweighting for cross-domain digit classification and object recognition. Recently, [104] propose a simple but effective DA approach based on correlation alignment, which learns a feature transformation by aligning the covariance matrices of the source and target domains. The approach has been applied on cross-domain object recognition problems. [14] propose to combine feature extraction with domain adaptation. It learns and coordinates both private and shared subspaces with a deep learning model. Recently [16] propose to extract invariant feature representations and estimate unbiased instance weights for minimizing the cross-domain distribution discrepancy. These methods though share some similarities with our proposed approach in bridging the cross-domain feature gaps with feature transformation and alignment, they are limited to homogeneous domain adaptation problems and do not handle disjoint cross-domain feature spaces.

3.2.2 (Semi-) Supervised HDA

Most existing HDA methods require the availability of a small number of labeled instances in the target domain. Depending on whether unlabeled target domain instances are utilized, these methods can be divided into two groups: supervised and semi-supervised methods.

Supervised HDA methods exploit the labeled target domain instances in addition to the source domain data to bridge the feature representation gap. For example, [51] learn an asymmetric and nonlinear feature transformation matrix for cross-domain image classification by solving an ARC-t problem with the help of the labeled data. [25] propose a heterogeneous feature augmentation method that transforms the data in both domains into a common subspace and then augments the projected data with original features. [37] propose a max-margin domain transformation method, which combines the learning of an asymmetric cross-domain transformation function and the learning the classification parameters in max-margin framework. [143] construct a sparse and class-invariant feature transformation matrix to map the weight vector of classifiers. Recently, [103] propose to use the shared label distributions across domains as pivots for learning a sparse feature transformation in a supervised HDA setting.

Semi-supervised HDA methods further exploit unlabeled target domain instances to help the adaptation. [129] propose to learn a discriminative common feature space for cross-view action recognition by minimizing canonical correlations of interclass instances and maximizing intraclass instances. [132] develop a semi-supervised kernel matching framework that simultaneously maps the target domain instance into the source domain instances and learns a prediction function on the labeled source instances. [113] propose a cross-domain landmark selection method to learn representative landmarks from cross-domain data. [138] propose a semi-supervised domain adaptation method which learns a subspace that reduces the underlying cross-domain difference and preserves the local structures of domains. More recently, [134] propose to learn a discriminative correlation subspace and the target domain classifier simultaneously with a unified objective, which achieves state-of-the-art results. The *requirement* of labeled target instances however remains to be a *limitation* for such (semi-)supervised methods.

3.2.3 Unsupervised HDA

Unsupervised HDA methods do not require any labeled data from the target domain and mainly use unlabeled instances to bridge the heterogeneous cross-domain feature spaces. Although a lot of approaches have been developed for unsupervised domain adaptation [123], unsupervised heterogeneous domain adaptation has received far little attention due to its difficulty. [34] propose a canonical correlation analysis (CCA) method which learns a common semantic representation between the text and image domains. This method can naturally bridge the representation gap across heterogeneous domains with parallel data. Recently, [139] propose a novel unsupervised HDA framework that exploits unlabeled cross-domain data pairs to derive a feature transformation model for cross-domain recognition. Similar to CCA, it utilizes the derived correlation subspace as a joint representation for associating data across domains, and advances reduced kernel techniques for kernel CCA (KCCA) for producing nonlinear correlation subspace. It also incorporates the domain adaptation ability into classifier design by employing a SVM with a correlation regularizer. This method however can only exploit the unlabeled cross-domain data pairs (i.e., parallel instances) which can be limited in many domains, while ignoring the large set of unlabeled nonparallel instances in each domain. Our proposed unsupervised HDA approach in this chapter can overcome such a drawback by exploiting all existing data in both domains in an unsupervised manner.

3.3 Unsupervised HDA with Sparse Feature Transformation

In this section, we present a novel sparse feature transformation method for unsupervised heterogeneous domain adaptation (SFT-HDA). It induces a feature transformation by aligning the distributions of the transformed source features and target features in an unsupervised manner.

3.3.1 Problem Setting

We assume D_s and D_t are the source and target domains respectively with different feature spaces. There are n_s labeled instances (X_s, Y_s) in the source domain D_s, where $X_s \in \mathbb{R}^{n_s \times d_s}$ is the feature matrix and $Y_s \in \{0,1\}^{n_s \times L}$ is the label matrix over L classes with a single 1 indicating its class label in each row. In the target domain D_t, we only have n_t unlabeled instances $X_t \in \mathbb{R}^{n_t \times d_t}$ and need to predict their unknown label matrix $Y_t \in \{0,1\}^{n_t \times L}$ in the same label space as in the source domain. Unlike vast HDA approaches that utilize the labeled instances in the target domain to adapt the heterogeneous domains, we consider a harder unsupervised scenario where there is no labeled data in the target domain. Instead, we assume there are a small number of n_p unlabeled parallel instances, i.e., (X_s^0, X_t^0) with $X_s^0 \in \mathbb{R}^{n_p \times d_s}$ and $X_t^0 \in \mathbb{R}^{n_p \times d_t}$. The unlabeled parallel instances have feature representations in both the source and target domain feature spaces to build cross-domain connections. Compared with the expensive labeled target domain instances, the acquisition of a small number of unlabeled parallel instances can be more convenient – they can be readily available in many applications; e.g., images taken by two cameras on the same set of objects.

Under this unsupervised domain adaptation setting, our proposed feature transformation framework can be demonstrated in Figure 3.1. We will present the details below.

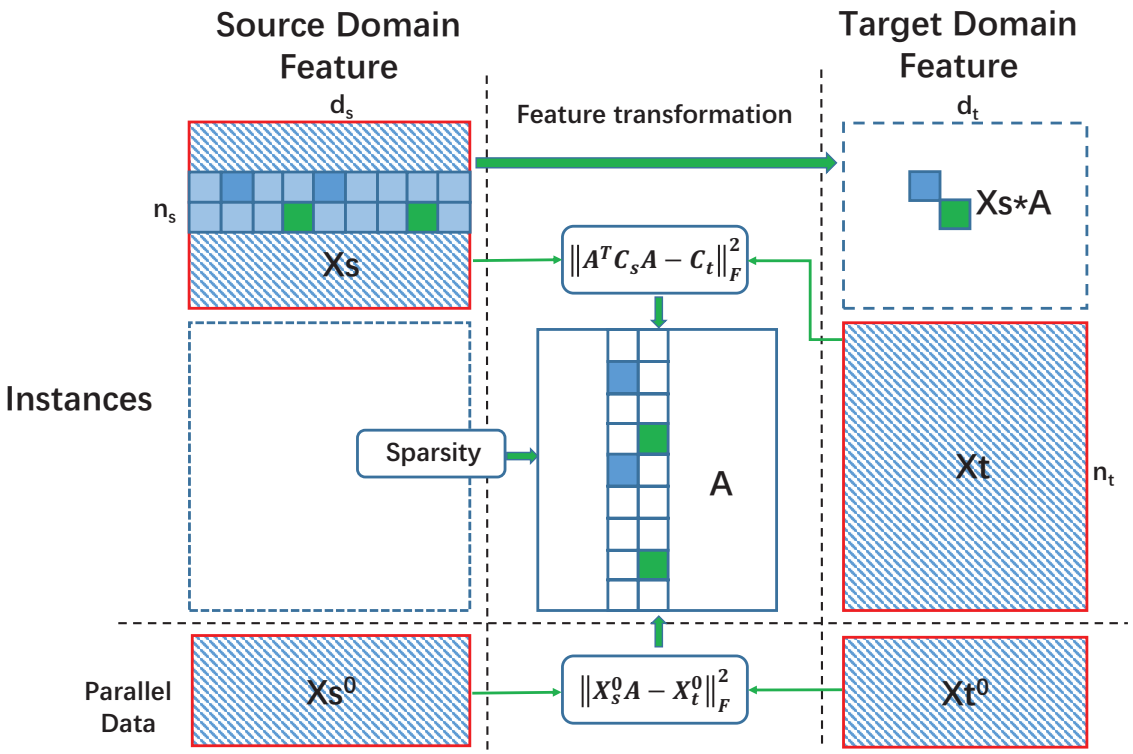


Figure 3.1: Unsupervised sparse feature transformation. The input data are indicated with blue stripes and red edges.

3.3.2 Feature Transformation Model for HDA

To exploit the information in the source domain for our target prediction task, the major challenge is to bridge the heterogeneous cross-domain feature spaces. Although projecting data from both domains into a common subspace has been a typically technique adopted in the literature for DA tasks, such a third space induces transformation loss for both source and target domains. Hence we propose to directly transfer the source domain features into the target domain feature space without seeking a middle ground representation. In particular, we consider a linear transformation function, $f : \mathcal{X}_s \rightarrow \mathcal{X}_t$, that maps the source domain features into the target domain features via a transformation matrix $A \in \mathbb{R}^{d_s \times d_t}$. On the parallel data, we expect the transformed data $f(X_s^0) = X_s^0 A$ can be a good approximation to its counterpart, X_t^0, in the target domain. By minimizing the squared approximation loss, this leads to the following transformation learning problem:

$$\min_{A} \quad \left\| X_s^0 A - X_t^0 \right\|_F^2 \tag{3.3-1}$$

where $\| \cdot \|_F$ denotes the Frobenius matrix norm. This learning formulation however entirely relies on the parallel instances, while ignoring the large amount of non-parallel data in the two domains. When the number of parallel instances is small, this learning mechanism can hardly induce well generalizable transformation functions – there can be distribution divergence between the transformed source feature space and the original target domain feature space.

To bridge the cross-domain divergence, we propose to align the transformed feature distribution with the original target domain feature distribution by adopting a second moment matching strategy [104]. Specifically, we minimize the distance between the second-order statistics, i.e., covariance, of the transformed source features

and target features and formulate the distribution alignment of HDA as:

$$\min_{A} \quad \left\| A^{\top} C_s A - C_t \right\|_F^2 \tag{3.3-2}$$

where the source feature covariance matrix $C_s \in \mathbb{R}^{d_s \times d_s}$ is computed from the non-parallel source domain data X_s, and the target feature covariance matrix $C_t \in \mathbb{R}^{d_t \times d_t}$ is computed from the non-parallel target domain data X_t. The covariance matrix in the transformed source feature space is computed with the linear transformation matrix A as $A^{\top} C_s A$. With the large amount of non-parallel data in the source and target domains, we expect the empirical covariance matrix matching will reduce the cross-domain feature distribution divergence and facilitate cross-domain information transfer.

To ensure both a meaningful feature transformation and a minimal cross-domain distribution divergence, we finally combine the loss functions in both Eq.(3.3-1) and Eq.(3.3-2) and formulate a heterogeneous feature transformation model as below by exploiting both the non-parallel data and the parallel-data in the two domains:

$$\min_{A} \quad \left\| A^{\top} C_s A - C_t \right\|_F^2 + \alpha \left\| X_s^0 A - X_t^0 \right\|_F^2 \tag{3.3-3}$$

where α is a trade-off parameter to balance the two losses.

3.3.3 Sparse Feature Transformation

The cross-domain feature transformation above can be interpreted as constructing each target domain feature as a linear combination of the source domain features. With a large number of source domain features, an unregularized full linear transformation however can easily cause overfitting, encode noise, or capture spontaneous cross-domain feature relations. We hence propose to enforce sparsity on our linear feature transformation model by employing the following form of mixed norm of sparsity

regularizers [50]:

$$\|A\|_{p,q} = \left[\sum_{j=1}^{d_t}\left[\sum_{i=1}^{d_s}|A_{i,j}|^p\right]^{q/p}\right]^{1/q} \tag{3.3-4}$$

With different (p, q) values, this mixed norm can result in different type of regularizers. This leads to the following sparse feature transformation model for HDA:

$$\min_A \frac{1}{2}\left\|A^\top C_s A - C_t\right\|_F^2 + \frac{\alpha}{2}\left\|X_s^0 A - X_t^0\right\|_F^2 + \frac{\gamma}{q}\|A\|_{p,q}^q \tag{3.3-5}$$

where γ is a trade-off parameter for the sparsity regularizer. By integrating the three components – distribution alignment of the source and target domains, transformation error minimization of the parallel data, and the sparsity regularization, we expect to learn a robust and generalizable feature transformation matrix A that can effectively bridge the cross-domain representation gap and facilitate information adaptation from the source domain to the target domain.

In this work, we consider two types of norms with (p=1,q=1) and (p=1,q=2) respectively. First, to filter noise and avoid spontaneous cross-domain feature correlations, we consider enforcing an overall sparsity regularization over the linear coefficients in the transformation matrix A. This can be achieved by using an entrywise ℓ_1 norm regularizer with p=1 and q=1:

$$\|A\|_{1,1} = \sum_{i=1}^{d_s}\sum_{j=1}^{d_t}|A_{i,j}| \tag{3.3-6}$$

Second, typically only a small fraction of the source domain features are related to a target domain feature. For instance, for domain adaptation across two types of image feature spaces, a target domain local feature that describes one part/position of image may only related to a small number of source domain features over several related parts/positions. With this motivation, we consider employing an individual

sparsity inducing $\ell_{1,2}$ norm regularizer below:

$$\|A\|_{1,2} = \left[\sum_{j=1}^{d_t}\left[\sum_{i=1}^{d_s}|A_{i,j}|\right]^2\right]^{1/2} \tag{3.3-7}$$

This $\ell_{1,2}$ norm can enforce individual sparsity on each column of the transformation matrix A separately [116], and hence relate each target domain feature to only a few source domain features with the non-zero entries of the corresponding column of matrix A.

3.3.4 Learning Algorithm

The unsupervised sparse feature transformation learning problem formulated in Eq.(3.3-5) is a quartic program with a non-smooth sparsity regularizer. It is difficult to tackle due to the existence of the quartic term. We propose to solve it using an alternating direction method of multipliers (ADMM), which breaks a complex optimization problem into a few simpler subproblems, and solves the simpler subproblems separately [15].

We first rewrite Eq.(3.3-5) into the following equivalent formulation by introducing an additional matrix B and an equality constraint

$$\min_{A,\,B}\ \frac{1}{2}\left\|A^\top C_s B - C_t\right\|_F^2 + \frac{\alpha}{2}\left\|X_s^0 B - X_t^0\right\|_F^2 + \frac{\gamma}{q}\|B\|_{p,q}^q \qquad \text{s.t. } A = B \tag{3.3-8}$$

The re-expressed problem is a quadratic minimization problem with sparsity regularizer in terms of A and B separately, subjecting to the equality constraint. The augmented Lagrangian function for this problem is

$$\begin{aligned} L_\rho(A,B,\Lambda) =& \frac{1}{2}\left\|A^\top C_s B - C_t\right\|_F^2 + \frac{\alpha}{2}\left\|X_s^0 B - X_t^0\right\|_F^2 + \\ & \frac{\gamma}{q}\|B\|_{p,q}^q + \mathrm{tr}(\Lambda^\top(A-B)) + \frac{\rho}{2}\|A-B\|_F^2 \end{aligned} \tag{3.3-9}$$

where Λ is the dual variable matrix associated with the equality constraint and ρ is the penalty parameter for the constraint. In each iteration of the ADMM algorithm, we then minimize this augmented Lagrangian over the primal variable matrices A and B separately, while updating the dual variable matrix. Specifically, in the $(k+1)$-th iteration, given the $(A^{(k)}, B^{(k)}, \Lambda^{(k)})$ from the previous iteration, we perform the following three steps.

(1) Minimization over B. Given the current fixed $A^{(k)}$ and $\Lambda^{(k)}$, B can be updated by minimizing the augmented Lagrangian:

$$B^{(k+1)} := \arg\min_{B} \; L_\rho(A^{(k)}, B, \Lambda^{(k)}) := \arg\min_{B} \; \ell(B) + \frac{\gamma}{q}\|B\|_{p,q}^q \tag{3.3-10}$$

where $\ell(B)$ is a smooth function such that

$$\ell(B) = \frac{1}{2}\left\|A^{(k)\top}C_sB - C_t\right\|_F^2 + \frac{\alpha}{2}\left\|X_s^0B - X_t^0\right\|_F^2 - \mathrm{tr}(\Lambda^{(k)\top}B) + \frac{\rho}{2}\left\|A^{(k)} - B\right\|_F^2$$

This minimization problem is a convex quadratic programming with a non-smooth sparsity regularizer. We solve it using a fast proximal gradient descent method with a quadratic convergence rate [11], which tackles Eq.(3.3-10) by solving a sequence of intermediate problems iteratively with proximity operators. In the t-th iteration, the intermediate problem at the current point $Q^{(t)}$ is in the following form:

$$\mathcal{P}_\eta(Q^{(t)}) = \arg\min_{B}\left\{\frac{1}{2}\|B - \widehat{Q}^{(t)}\| + \frac{\gamma}{q\eta}\|B\|_{p,q}^q\right\} \tag{3.3-11}$$

where $\widehat{Q}^{(t)} = Q^{(t)} - \frac{1}{\eta}\nabla\ell(Q^{(t)})$ is derived from the gradient of $\ell(Q^{(t)})$ at the current point $Q^{(t)}$ and η is the Lipschitz constant of the gradient; we used $\eta = \sigma_{\max}\big(C_s^\top A^{(k)}A^{(k)\top}C_s + \alpha X_s^{0\top}X_s^0 + \rho I\big)$, where $\sigma_{\max}$ denotes the largest singular value of the given matrix. The nice property about this intermediate problem is that it

allows us to exploit closed-form solutions for the proximity operator $\mathcal{P}_\eta(Q^{(t)})$ with either the ℓ_1-norm regularizer ($p = 1$ and $q = 1$) or the $\ell_{1,2}$-norm regularizer ($p = 1$ and $q = 2$). According to [50], we have a closed-form solution $\mathcal{P}_\eta(Q^{(t)}) = \tilde{Q}$ for the proximity operation such that for all (i, j),

$$\tilde{Q}_{i,j} = \begin{cases} \text{sign}(\widehat{Q}_{i,j}^{(t)}) \left(|\widehat{Q}_{i,j}^{(t)}| - \frac{\gamma}{\eta} \right)_+ & \text{If } p = 1, q = 1 \text{ } (\ell_1\text{-norm}); \\ \text{sign}(\widehat{Q}_{i,j}^{(t)}) \left(|\widehat{Q}_{i,j}^{(t)}| - \frac{\gamma \sum_{r=1}^{m_j} \overrightarrow{Q}_{r,j}}{(\eta + \gamma m_j) \|\widehat{Q}_{:j}^{(t)}\|_2} \right)_+ & \text{If } p = 1, q = 2 \text{ } (\ell_{1,2}\text{-norm}); \end{cases} \tag{3.3-12}$$

where $(\cdot)_+ = \max(0, \cdot)$, $\overrightarrow{Q}_{:j}$ denotes a reordered j-th column $|\widehat{Q}_{:j}^{(t)}|$ with a descending order of the entries, and m_j is the number such that

$$\overrightarrow{Q}_{m_j+1,j} \leq \frac{\gamma}{\eta} \sum_{r=1}^{m_j+1} \left(\overrightarrow{Q}_{r,j} - \overrightarrow{Q}_{m_j+1,j} \right), \quad \text{and} \quad \overrightarrow{Q}_{m_j,j} > \frac{\gamma}{\eta} \sum_{r=1}^{m_j} \left(\overrightarrow{Q}_{r,j} - \overrightarrow{Q}_{m_j,j} \right). \tag{3.3-13}$$

With the proximal operators, the proximal gradient descent method can easily handle the non-smooth mixed-norms and produce desired sparse solutions.

(2) Minimization over A. With the current values of $B^{(k+1)}$ and $\Lambda^{(k)}$ being fixed, we update A by minimizing the augmented Lagrangian objective:

$$A^{(k+1)} := \arg\min_A \; L_\rho(A, B^{(k+1)}, \Lambda^{(k)}) \tag{3.3-14}$$

This is a simple quadratic minimization problem, which yields a closed-form solution:

$$A^{(k+1)} = \left(C_s B^{(k+1)} B^{(k+1)\top} C_s^\top + \rho I \right)^{-1} \left(C_s B^{(k+1)} C_t^\top - \Lambda^{(k)} + \rho B^{(k+1)} \right) \tag{3.3-15}$$

(3) Update of the dual variable matrix Λ. Following the standard ADMM method, we update the dual matrix Λ by

$$\Lambda^{(k+1)} := \Lambda^{(k)} + \rho(A^{(k+1)} - B^{(k+1)}) \tag{3.3-16}$$

Given the input data and parameter settings, the algorithm first initializes A, B and Λ before the iterative updates. We initialize A and B by simply setting them as the closed-form solution of the ℓ_2-norm regularized parallel data transformation:

$$\begin{aligned} A^{(1)} = B^{(1)} &= \arg\min_{B} \ \|X_s^0 B - X_t^0\|_F^2 + \lambda \|B\|_F^2 \\ &= (X_s^{0\top} X_s^0 + \lambda I_{d_s})^{-1} (X_s^{0^\top} X_t^0) \end{aligned} \tag{3.3-17}$$

where I_{d_s} denotes an identity matrix of size $d_s \times d_s$; λ is the regularization trade-off parameter and can be used to avoid the numerical problem of matrix inversion in the closed-form solution. We expect this initialization can provide a more informative starting point than random initialization. For the dual matrix Λ, we initialize it with all zero values.

The iterative three step updates of primal and dual variable matrices of the ADMM algorithm aim to minimize the augmented Lagrangian function, which will eventually push A to be close to the sparse B to recover the equality constraint $A = B$. Hence the algorithm eventually solves Eq.(3.3-8). It has been shown in [38] that even in the presence of non-convex objective, the ADMM algorithm is able to reach the set of stationary solutions for the linearly constrained problem in the form of Eq. (3.3-8). In our experiments, we adopt the following stopping criterion for the iterative updates: We stop the iteration loop whenever the distance between $A^{(k+1)}$ and $B^{(k+1)}$ is less than a very small positive constant ϵ or the maximum iterations is reached.

3.3.5 Cross-Domain Classification with Feature Transformation

After obtaining a cross-domain feature transformation matrix A, we can transform the labeled source domain data X_s into the target domain feature space as X_sA. Then we can train a multi-class classification model over the labeled data (X_sA, Y_s) and use it to predict the class categories of the unlabeled target domain instances X_t. In our experiments, we used one-vs-all SVM as the classification model.

3.4 Experiments

3.4.1 Datasets and Settings

We conducted experiments on three datasets, *UCI Multiple Features* [5], *Wikipedia* [84], and *Office-Caltech* [30]. The *UCI multiple features dataset* contains 2000 images of 10 handwritten digits from '0' to '9', with 200 images per-class. For each image, there are six types of features. We dropped two types of features which have very small dimensions, and used the remaining four types of features: Fourier coefficient (fou), profile correlations (fac), Karhunen-Love coefficients (kar) and pixel averages (pix). By using one feature type as the source domain and another as the target domain, we formed 12 HDA tasks. For each task, 100 instances are pre-selected as the parallel instances. We then randomly sampled 20 instances per-category as the labeled source instances and the rest are used as the target instances. The *Wikipedia dataset* contains 2,866 multimedia documents over 29 categories, and each document consists of one paragraph of text and one related image. The images are represented as 128-dim bag-of-word SIFT features. Latent Dirichlet Allocation (LDA) is used to extract 10-dim text features from the document set. Following [139], we considered five categories: art and architecture, biology, literature, sport, and warfare. In total 200 instances are used as parallel instances. Then 100 instances are selected for each category: 20 instances per-category are used as the labeled source instances and the rest instances

are used as the target domain instances. The *Office-Caltech dataset* contains 10 classes of images from four domains: Amazon (A), DSLR (D), Webcam(W) and Caltech-256(C). We excluded the DSLR domain as there are very few instances per class. In addition to the 800-dimensional SURF features, we extracted 4096-dimensional CNN features (VGG19) [98]. We select one domain from the three domains (A, C, W) as the source domain with one feature type (e.g., SURF), and select another domain as the target domain with a different feature type (e.g., VGG19). Hence in total we have 6 HDA tasks from SURF feature to VGG19 and another 6 HDA tasks from VGG19 feature to SURF. We selected 50 instances from both the source and target domains as the unlabeled parallel data. Then we randomly selected 20 instances (10 for Webcam) per-category as the labeled source domain instances and used the other source instances as unlabeled source instances. The instances in target domain are used as unlabeled target instances.

3.4.2 Comparison Methods

There is not much work on unsupervised HDA. We compared to two CCA based unsupervised HDA methods. Moreover, we also compared to a number of variants of the proposed model by only considering parts of the three components in Eq.(3.3-5). All the comparison methods used in the experiments are listed below.

- **linear CCA:** It uses the linear canonical correlation analysis [34] to learn a common cross-domain representation with the unlabeled parallel instances.
- **Rd KCCA:** This is a Reduced Kernel CCA method, which is an unsupervised HDA method from [139].
- **SFT-noCov:** A variant of the proposed SFT-HDA method that drops the covariance alignment component.
- **SFT-noPara:** A variant of the proposed SFT-HDA that drops the parallel

data.

- **SFT-noSparse:** A variant of the proposed SFT-HDA method that drops the sparse regularization term.
- **SFT$_{1,1}$** and **SFT$_{1,2}$:** Our proposed SFT-HDA approach with the ℓ_1-norm and $\ell_{1,2}$-norm sparisity regularizers respectively.

For both linear CCA and Rd KCCA, we conducted experiments following the work in [139]. and set the correlation coefficient as 0.5. But we set the size of reduced set as 30, which leads to better performance than their original setting in our experiments.

For each method, a linear Support Vector Machine (SVM) is trained on the transformed labeled source instances, and tested on the target domain instances. For linear CCA and Rd KCCA, the target instances are also projected to the learned subspace. The hyper-parameter C of the SVM is selected with 5-fold cross-validation on the transformed labeled source instances. We also tried the linear CCA and Rd KCCA with the correlation-transfer SVM (CTSVM) proposed in [139], the improvement of accuracy is about 2% and it has little influence on the conclusions. To provide a fair comparison, we hence reported the SVM classification results for all comparison methods.

3.4.3 Parameter Selection

There are three hyper parameters in our approach: ρ, α and γ. However, ρ is only related to the ADMM optimization algorithm and it just needs to be set to a reasonable large value to guarantee the recovery of the equality constraint. We used $\rho = 10$ for the first and second experiments, but used $\rho = 1000$ in the third experiment to make the huge sparse feature transformation converge quicker. α is the trade-off parameter to balance the weights of the distribution alignment loss and the parallel mapping loss. We simply gave both losses equal weights and set $\alpha = 1$ in all experiments. As this is an unsupervised approach (no labeled data in the

Table 3.1: Average classification accuracy (%) over 20 runs on UCI Multiple Feature dataset.

Source	fou	fou	fou	fac	fac	fac
Target	fac	kar	pix	fou	kar	pix
linear CC	36.34	32.36	32.31	27.32	50.60	57.45
Rd KCCA	64.03	52.67	58.62	52.33	76.26	87.49
SFT-noCov	66.05	58.28	65.17	55.84	83.79	87.67
SFT-noPara	14.49	12.64	7.37	9.01	18.18	11.86
SFT-noSparse	71.16	52.87	65.37	59.53	85.00	92.96
$SFT_{1,1}$	**73.83**	56.50	**69.53**	**61.53**	**85.97**	93.35
$SFT_{1,2}$	71.06	**60.44**	66.79	61.17	85.50	**93.53**
Source	kar	kar	kar	pix	pix	pix
Target	fou	fac	pix	fou	fac	kar
linear CC	29.72	57.37	74.00	27.14	51.58	66.78
Rd KCCA	53.62	84.85	86.43	53.33	89.65	81.66
SFT-noCov	62.16	81.86	88.86	58.07	83.88	87.40
SFT-noPara	11.54	18.27	17.48	6.99	17.56	16.45
SFT-noSparse	62.47	91.30	93.71	56.51	91.59	89.31
$SFT_{1,1}$	62.66	91.59	**93.82**	58.92	**92.14**	**89.69**
$SFT_{1,2}$	**63.40**	**91.86**	93.79	**60.01**	92.01	88.13

target domain), the traditional hyper-parameter tuning method of cross-validation is not really applicable. Nevertheless, we used 5-fold cross-validation of linear SVM to select the value of γ from $[0.01, 0.1, 1]$ on the transformed labeled source instances $X_s A$. Moreover, the regularization parameter λ in the closed-form initialization of A and B is also set to a reasonable large value – we simply used the same value as ρ, while $\epsilon = 10^{-4}$ is used for detecting the convergence of the training algorithm.

3.4.4 Classification Results

Digits Classification

We first tested all the comparison methods on the 12 HDA tasks formed on the UCI Multiple Features dataset for digits classification. The average multi-class classification accuracy results over 20 runs are reported in Table 1 – each run is with a different random source and target instance partition. We can see SFT-noPara has very poor performance here. Though this strategy works well on DA tasks [104],

Table 3.2: Average classification accuracy (%) over 20 runs on Wikipedia dataset.

HDA task	Image → Text		
# parallel instances	100	150	200
linear CCA	58.23±2.50	49.60±3.61	40.80±2.27
Rd KCCA	47.67±4.83	64.58±2.06	67.40±2.55
SFT-noCov	75.70±2.11	81.47±1.44	84.35±1.58
SFT-noPara	22.10±1.99	22.10±1.99	22.10±1.99
SFT-noSparse	61.02±2.28	69.53±2.78	78.88±2.02
$SFT_{1,1}$	71.47±2.74	79.03±2.11	83.78±1.37
$SFT_{1,2}$	**76.17±2.81**	**83.92±1.56**	**88.05±1.44**
HDA task	Text → Image		
# parallel instances	100	150	200
linear CCA	27.48±0.72	20.88±0.43	26.88±0.48
Rd KCCA	35.92±1.95	38.85±1.23	**46.33±0.80**
SFT-noCov	41.92±0.43	**41.08±0.66**	42.98±0.46
SFT-noPara	20.25±1.18	20.25±1.18	20.25±1.18
SFT-noSparse	**43.35±0.91**	40.33±0.78	43.55±0.66
$SFT_{1,1}$	42.83±0.67	40.23±0.68	42.80±0.55
$SFT_{1,2}$	42.42±0.77	40.58±0.76	42.85±0.55

HDA is apparently a more challenging task. This set of HDA experiments suggest that the parallel data transformation component provides a major contribution to our SFT-HDA model. But nevertheless, the covariance matching and sparsity regularization components can further improve the performance, as the two full versions of the proposed SFT-HDA ($SFT_{1,1}$ and $SFT_{1,2}$) cover the best results across all the 12 HDA tasks. Compared with the baseline variants, the accuracy results of $SFT_{1,1}$ and $SFT_{1,2}$ are in general better than SFT without either the sparsity regularizer or the covariance alignment component. Moreover, our proposed full SFT-HDA methods substantially outperform the linear CCA and Rd KCCA methods, which depend on the parallel data. These results verified the efficacy of our proposed model.

Multimedia Classification

On the multimedia *Wikipedia dataset*, we have two HDA tasks, one performs adaptation from image to text and the other adapts from text to image. On this dataset, instead of using a fixed number of parallel instances, we conducted evaluations with

different numbers of parallel instances, i.e., with n_p varies from 100 to 200. The mean values and standard deviations of the multi-class classification accuracy results over 20 runs for the 2 HDA tasks are reported in Table 2. We can see for Image→Text HDA task, $\text{SFT}_{1,2}$ achieves a high accuracy of 76.17% even if there is only 100 parallel instances and it outperforms linear CCA and Rd KCCA by about 18% and 29% respectively. The performance of Rd KCCA is not good when there is not much parallel data available. Linear CCA performs worse as n_p increases to 200, which might be caused by the small dimension of the subspace and the high variety of the covariance with more instances. With the increasing of the number of parallel instances, the performance of our proposed approaches increases dramatically, which again shows the importance of parallel data. We can also see that with $\ell_{1,2}$-norm $\text{SFT}_{1,2}$ produces the best results on Image→Text and outperforms the ℓ_1-norm variant $\text{SFT}_{1,1}$ with notable margins. This suggests each text feature can be explained by a small fraction of image features. For the Text→Image HDA task, the sparsity regularizers for our approach however are not effective, and the impact of increasing parallel instance number is negligible. The reason is that the feature dimension of the text domain is quite small – only 10 features. It is hence not beneficial to have a sparse mapping to the image features or increase the number of parallel instances. Overall, our proposed full approaches again outperform both linear CCA and Rd KCCA.

Cross-domain Image Classification

The experimental results on 12 HDA tasks of the *Office-Caltech* dataset are reported in Table 3. The 4096-dim deep features are challenging. Rd KCCA fails to work on this dataset because it relies on k-means to cluster the instances into several splits with similar sizes and k-means fails to work on centralized deep features. The sparsity regularizers in our model are also affected by the deep features. The performance of SFT-noSparse, $\text{SFT}_{1,1}$ and $\text{SFT}_{1,2}$ are quite close. Nevertheless, our

Table 3.3: Average classification accuracy (%) over 20 runs on Office-Caltech dataset.

HDA task	SURF → VGG					
Source	A	A	W	W	C	C
Target	W	C	A	C	A	W
linear CCA	33.96	43.09	40.14	40.36	44.63	28.96
SFT-noCov	57.31	63.04	65.46	63.89	64.68	49.73
SFT-noPara	14.55	8.14	2.37	7.03	19.48	13.53
SFT-noSparse	**72.69**	70.05	**78.57**	**72.30**	72.64	**50.04**
$SFT_{1,1}$	72.59	69.99	78.06	**72.30**	**73.58**	48.33
$SFT_{1,2}$	71.57	**70.26**	76.66	70.90	70.79	46.55
HDA task	VGG → SURF					
Source	A	A	W	W	C	C
Target	W	C	A	C	A	W
linear CCA	29.41	29.91	24.28	24.81	31.15	25.61
SFT-noCov	33.51	31.10	37.21	**33.36**	43.47	43.82
SFT-noPara	9.67	11.64	7.84	9.55	18.86	16.41
SFT-noSparse	**37.12**	**34.51**	37.26	32.03	**48.87**	53.37
$SFT_{1,1}$	36.71	34.32	38.45	32.05	48.19	**54.86**
$SFT_{1,2}$	35.61	33.85	**38.56**	33.10	46.58	52.84

proposed approaches still outperform the linear CCA method, while our approach is the first to achieve promising results under unsupervised HDA setting for deep features.

Parameter Sensitivity Analysis

We conducted sensitivity analysis for the trade-off parameters α and γ with the HDA tasks on the Wikipedia dataset. We used 150 parallel instances, $\rho = 10$ and $\lambda = 10$. For the proposed methods, we conducted experiments first with $\gamma = 1$ and $\alpha \in [0.1, 0.5, 1, 2, 10]$, and then with $\alpha = 1$ and $\gamma \in [0.01, 0.05, 0.1, 0.5, 1]$. The average accuracy and standard deviation of 20 rounds for each HDA task are reported in Figure 3.2. From the two sub-figures on the left side, we can see that with a fixed γ value, $\alpha = 1$ leads to test performance that is among the best for both HDA tasks, which suggests that it is reasonable to give equal weights to the distribution alignment loss and the parallel mapping loss. The parameter γ controls the sparsity regularization term. The best value choice for γ depends on the properties of cross-

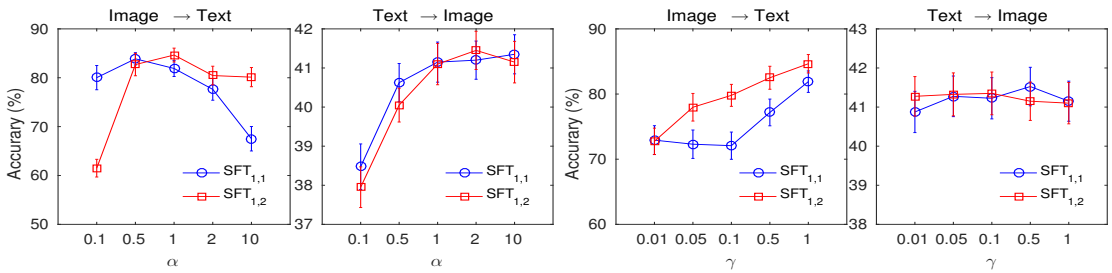


Figure 3.2: Performance of $SFT_{1,1}$ and $SFT_{1,2}$ with respect to trade-off parameters α and γ for two HDA tasks on Wikipedia dataset

domain features. As shown in the two sub-figures on the right side, γ should be set to a relative large value for mapping 128-dimensional image features to 10-dimensional text features. We also tested a much larger γ value than the range in the figure, but found the performance could be unstable and drop down rapidly, while selecting γ from $[0.01, 0.1, 1]$ in the previous experiments is reasonable for both the mapping from high to low dimensional feature space and the reverse mapping.

3.5 Conclusions

In this chapter, we propose a novel sparse feature transformation method for unsupervised heterogeneous domain adaptation. The method transforms the source domain features into the target domain feature space by matching the parallel instances and aligning the empirical second moments of the transformed source feature distribution and target domain feature distribution. To encode the assumption that only a small fraction of source domain features are related to a target domain feature and increase the robustness of transformation, we further exploit two types of sparsity inducing norms to regularize the linear transformation model. We develop an ADMM based optimization algorithm to solve the induced problem and conduct experiments for heterogeneous cross-domain classification. The experimental results demonstrate the benefits of our proposed approach.

CHAPTER 4

USER INTERESTS LEARNING: SESSION-BASED NEWS RECOMMENDATION FROM TEMPORAL USER COMMENTING DYNAMICS

In this chapter we propose a novel session-based news recommendation framework that model user's interest with Neural Network. With the increase in volume of daily online news items, it is more and more difficult for readers to identify news articles relevant to their interests. Thus, effective recommendation systems are critical for an effective user news consumption experience. Existing news recommendation methods usually rely on the news click history to model user interest. However, there are other signals about user behaviors, such as user commenting activity, which have not been used before. We propose a recommendation algorithm that predicts articles a user may be interested in, given her historical sequential commenting behavior on news articles. We show that following this sequential user behavior the news recommendation problem falls into in the class of session-based recommendation. The techniques in this class seek to model users' sequential and temporal behaviors. While we seek to follow the general directions in this space, we face unique challenges specific to news in modeling temporal dynamics, e.g., users' interests shift over time, users comment irregularly on articles, and articles are perishable items with limited lifespans. We propose a recency-regularized neural attentive framework for session-based news recommendation. The proposed method is able to capture the temporal

dynamics of both users and news articles, while maintaining interpretability. We design a lag-aware attention and a recency regularization to model the time effect of news articles and comments. We conduct extensive empirical studies on 3 real-world news datasets to demonstrate the effectiveness of our method.

4.1 Introduction

Many news media outlets allow users to comment on the news stories published on their websites (75% of U.S. online media offered this feature in 2008 [31]). User comments have evolved into a standard feature of online news and are considered one of the popular form of public online participation [122]. In this work, we aim to predict the article of interest for users. Comments are used as the proxy for an article of interest. We tackle the problem from a session-based news recommendation perspective.

Session-based recommendation is a scenario where implicit feedback (e.g., browsing, comments) is collected within a session from anonymous users [93]. The sequence of their implicit feedback determines the recommendation actions. Unlike session-based recommendation in e-commerce or movie settings, news articles and their readers exhibit unique temporal dynamics: the expected lifetime of news articles is short and their impact is typically bounded by their *immediacy*– their closeness to an emerging event– and readers' interest change over time. Hence, news recommendation faces challenges specific to both session-based recommendation and temporal dynamics modeling.

The classic way to predict users' interest in news recommendation is based on clicks, where user's interest are estimated by predicting articles to click, which could inaccurately reflect user's interest. Many clicks are accidental or for a quick glance. A user may be attracted by the title and lose interest in the news article soon after the click. This click action is not distinguishable from a detailed reading (strong

interest) in news recommendation. Also, there is no good way to distinguish a wrong or quick glance click from a detailed reading. The comments on the most popular news websites are usually of good quality. There are typically automatic moderation, filtering systems, and human monitoring on major news websites. Furthermore, the text of comments provide richer information for filtering and pre-processing than clicks for recommendation approaches. Thus, we hypothesize that a user's commenting action is a strong signal about news consumption interest.

We also need to consider that a user's interest changes over time, commenting activity is unevenly distributed over time, and the user may not comment on every article she reads; and that the news value of an article to a user decays over time. Thus, we have to abstract out user-news article-comment temporal dynamics into new temporal variables, such as the time interval from a historical commenting action until the time when the recommendation is provided, and the age of published article at the time we compute a recommendation. We call the former *lag*, and the latter *recency*.

A number of deep learning inspired methods have been specifically developed for news recommendation [79, 71]. They adapt popular general session-based recommendation approaches [108, 54]. Several other efforts solve the problem using traditional recommendation approaches [118, 55]. The latter class of works are out of the scope of this study.

In this work, we propose a method to capture the unique temporal user commenting dynamics with the following capabilities: (1) it uses RNN-based models to capture a user's sequential behavior; (2) it includes an attention mechanism to model a user's uneven commenting actions and another to model the importance of a reader's historical actions; and (3) it includes a recency-based regularizor to penalize the prediction scores of fresh articles (smaller recency) less, and to penalize older articles (larger recency) more. We make the following contributions in this work:

- We propose user commenting activity and its associated temporal dimensions

as a new basis for news recommendation.

- We propose an interpretable attentive neural network framework that captures temporal dynamics observed in news recommendation.
- We perform an extensive empirical study with a large dataset of news articles from three news outlets. The performance gain of our proposed model over the baselines is at least 40%.
- We provide in-depth analysis of the proposed method.

4.2 Related Work

In this section, we present related studies in two areas: session-based recommendation methods, and news recommendation methods. We omit traditional recommendation methods, e.g., collaborative filtering approaches[102, 48], matrix factorization [49, 70, 86], item-based collaborative filtering [60, 92], as they are not suitable for our problem; they do not consider the effect of temporal variables and ignore the connections between neighboring user actions.

4.2.1 Sequential Recommendation Methods

Non Deep Learning Approaches. Given the limitations of collaborative filtering approaches, researchers have looked into ways to model the sequential behavior of users while browsing, commenting, or purchasing items online. Methods based on Markov Chains are typical for sequential recommendation [148, 17]. Shani et al. [94] give one of the earliest attempts in this space, using a Markov Decision Process. Rendle et al. [87] factorizes the tensor constructed by the partially observed personalized probabilistic transition matrices, and incorporates the strengths of both Markov Chain methods and Matrix Factorization methods. Wang et al. [120] propose a two-layer hierarchical model to learn the aggregated representation of items from the last

transaction as well as user's representation, and considers both user's sequential behavior and user's general interest. Markov Chain based methods however are limited by the Markov assumption, which assumes that the current state is only dependent on the previous state. Under this assumption, it is difficult to capture a user's evolving interest. One other family of methods proposes to model temporal related factors in recommendation. Koren et al. [47] present a collaborative filtering based approach to capture temporal dynamics in the Netflix recommendation problem. They propose a time-aware collaborative filtering factor model by modeling latent factors of users and items as functions of time. Kararzog et al. [43] introduce a collaborative filtering based tensor factorization approach to factorize the multi-dimensional user-item-context tensor data. Although they consider temporal factors, they do not model the sequential behavior of users. Session-based KNN approaches [39, 63] are proposed as strong baselines compared to deep learning approaches. Garg et al.[29] achieve significant improvement over neighborhood-based method for session-based recommendation by utilizing sequential and temporal information from sessions. They design three decay/importance factors about positions of sessions and items. However, this general method does not focus on sequential information and user's interest, which play essential roles in news recommendation.

Deep Learning Approaches. Deep learning models have been applied successfully in sequential recommendation, a.k.a., *session-based recommendation*, given their advantages in modelling long-term and short-term aspects in user's sequential behavior online [63]. A Gated Recurrent Unit model with ranking-based loss function and session-parallel mini-batch training process is the first successful attempt of applying RNN to session-based recommendation [35]. It was latter improved by adding data augmentation and dropout on the input [108]. Li et al. [54] propose a hybrid attentive RNN model with a global recommender to capture user's general interest, and a local recommender to model user's current interest. Similarly, STAMP [61] is an atten-

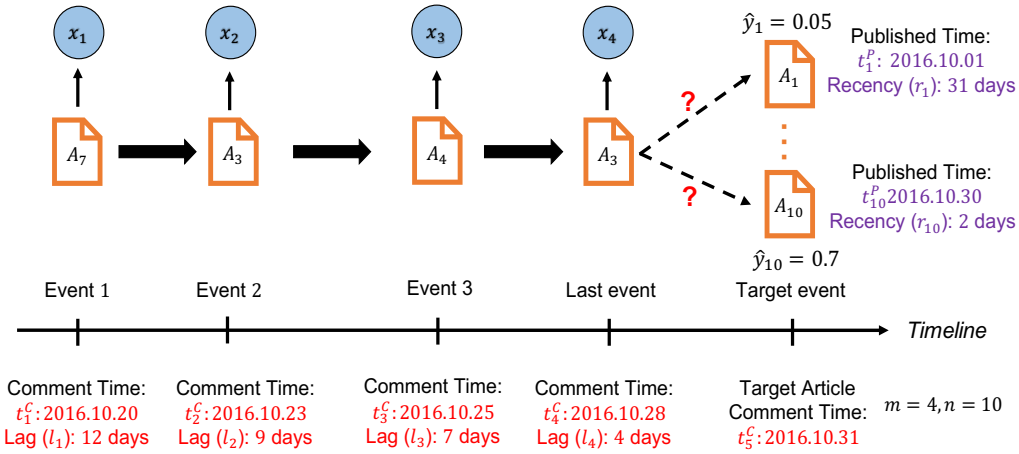


Figure 4.1: An example of a session with 4 historical events and 10 news articles in news recommendation ($m = 4, n = 10$).

tive multilayer perceptron neural model to capture the long-term memory from the aggregation of user's sequential behaviors. It captures the short-memory from user's last action. Recurrent approaches have also been applied in combination with collaborative filtering approaches by modeling the evolution of user rating and item rating via RNNs [22, 126]. Wang et al. [119] applies collaborative neighborhood information to session-based recommendations with hybrid inner and outer memory encoder modules. Although those methods can model user's sequential behavior, they do not exploit any temporal variables, like the time interval between user's actions. Zhu et al. [144] propose a variant of LSTM [36] by considering the effect of time intervals. The proposed model aims to better capture both user's short-term and long-term interests. Beutel et al. [12] also propose a method to utilize contextual data like time intervals, page, and software client for YouTube video recommendation. Several other RNN based methods are developed to model sequential data by utilizing contextual data in similar applications, for example, multivariate time series forecasting [52], and predicting user's retention time [41]. Since their applications are different, these works are out of the scope of the study in this paper. Nowadays, graph neural network has attracted a lot of attention. Wu et al. [128] models session sequences

as graph structured data and captures complex item transitions. Xu et al. [133] enhances self-attention network of session-based recommendation by combining graph neural network and self-attention model. Song et al. [101] extends graph-attention network by modeling dynamic interests and user influences. Repeat recommendation in session based recommendation is explored by Ren et al. [85]. They proposes a repeat-explore mechanism with encoder-decoder structure that automatically learns repeat and explore modes.

These methods do not cope well with the dynamic changes (fast outdated and evolving user interest) in the news environment [142]. The general recommendation systems do not handle well item value decay (e.g., the sharp relevance decay of a news article to the daily news) and shifts in a user's preferences [71], specific to news.

4.2.2 News Recommendation Methods

Deep learning has been successfully applied to news recommendation. For instance, one effort [79] uses an RNN method with a global encoder very similar to that proposed in [54] (described above) to recommend news articles. Its input consists of the traditional elements: the content of articles and reader browsing history. Moreira et al. [71] propose a two-step algorithm to provide session-based news recommendation by first learning the representation of news articles using their categories, and then applying a regular RNN model to serve as the predictor. The approach is similar to that of Tan et al. [108], except for the procedure of learning the embeddings of news articles. Zheng et al. [142] present a reinforcement learning approach for personalized news recommendation by utilizing news, user, and context features. We do not compare with them as we aim to provide recommendation in anonymous sessions and user information is hence not available. Wang et al. [118] conduct news recommendation by learning both reader embedding and news article embedding using convolutional neural network and knowledge graph, while Wu et al. [127] exploit het-

erogeneous user behavior via CNN networks and attentive learning framework. These two works solve the problem for news aggregators (e.g., Bing News) and use external knowledge, such as search query logs, none of which are available to news outlets. News outlets however have the user commenting activity, which the aggregators do not have.

In our work, we conduct session-based news recommendation based on users' historical commenting activities at news outlets. We utilize temporal variables like lag of historical events and recency, and incorporate them into an interpretable recency regularized attentive neural method to capture users' evolving interest and uneven commenting activity over time as well as the short lifespan of news article. To our knowledge, none of the above works have these characteristics.

4.3 Problem Definition

Notations: We introduce the key concepts and define the session-based news recommendation problem in this section. We will use bold lowercase letters for vectors (e.g., $\boldsymbol{h}_1$), and normal lowercase letters for scalars (e.g., w_1).

An *event* is defined as a reader's action of commenting on a news article, and a *session* is defined as a sequence of events. Let $\mathcal{A} = \{A_j | j = 1, \cdots, n\}$ be a set of n distinct news articles (i.e., $|\mathcal{A}|$ = n). t_j^P denotes the publication time of news article A_j. Let $[x_1, \cdots, x_m, x_{m+1}]$ denote a session (S) of historical events ordered by time, where x_i is the index of the news article commented in event i. We use t_i^C to denote the commenting time on the same article in event i. The *last event* and the *target event* correspond to event m and event m+1 in S, respectively. The *lag* l, which is the time interval between the historical event and the target event, is calculated as

$$l_i = t_{m+1}^C - t_i^C + 1,$$

for event i. The *recency* r of article A_j, which is the "age" of article A_j at the time when *target event* occurs (t^C_{m+1}), is calculated as

$$r_j = t^C_{m+1} - t^P_j + 1$$

The problem: We are given the sequence of historical events in an anonymous session, the pre-computed lags for historical events and recency for each article in $\mathcal{A}$. There is no other information for each user besides the comments he left. The recommendation task is to predict which article a user will comment in the target event. □

$\boldsymbol{y} \in \mathbb{R}^n$ denotes the target label, where $y_j = 1$ if A_j is the target article, and 0 otherwise. The prediction is denoted by $\hat{\boldsymbol{y}}$, where $\hat{y}_j$ is the probability that A_j is the target article. Figure 4.1 gives an example of a session with 4 historical events and 10 articles. In Figure 4.1, all articles are marked in orange and their corresponding representations are marked in blue. Lag and commenting time of news articles in historical events are marked in red. Published time and recency for articles are marked in purple. In this example, the session contains a sequence of user's commenting activities on A_7, A_3, A_4 and A_3 again. The prediction score for A_1 and A_{10} are 0.05 and 0.7, respectively. A_{10} is very likely because A_{10} is a fresher article (with smaller recency).

4.4 The Proposed Methods

We describe our proposed recency regularized attentive neural model in this section. We describe our methods to capture user-article-comment temporal variables, such as, user's time-varying interest, user's irregular commenting activity, and limited lifespan of an article.

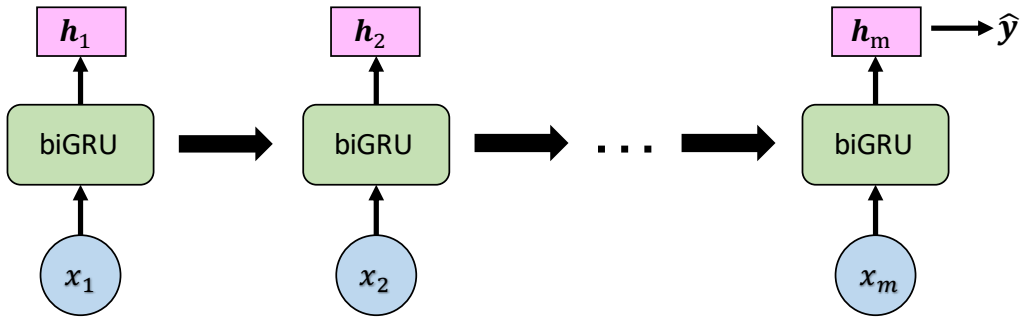


Figure 4.2: An illustration of a regular Recurrent Neural Network with bidirectional GRU.

4.4.1 Sequential Behavior Modeling

In this section, we give a brief background on the basic approach of modeling user's time-varying interest with RNN. As argued above, the traditional collaborative filtering methods are not feasible candidates in our temporally dynamic setting. They however work well in stationary settings. In this work, we use RNN to model the user's sequential behavior while consuming news. In particular, the information from previous user actions is propagated through the hidden unit of previous RNN cell to the current RNN cell. The idea is illustrated in Figure 4.2, where all inputs (user's actions) are marked in blue, RNN cells are marked in green and the hidden states are marked in magenta. This basic architecture is able to capture the evolution of user's interests over time.

We use Gated Recurrent Unit (GRU) [19] in our design, as GRU is able to learn when and by what weight we need to update the hidden unit in the recurrent cell. Bidirectional extensions further enhance recurrent models by considering input sequences from both past and future during training. Given its advantages of sequence modeling, (bidirectional) GRU is the preferred building block for solving general session-based recommendation problems [35, 108]. For ease of representation, we express the bidirectional GRU (biGRU) model as,

$$[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m] = biGRU([x_1, \cdots, x_m]) \tag{4.4-1}$$

where x_i, the index of the commented article in the i^{th} event, is the input of the biGRU cell i, and $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ is the corresponding hidden unit of the biGRU cell i, where d_h is the hidden size. In particular, the hidden unit $\boldsymbol{h}_i$ is the sum of the forward hidden unit $\overrightarrow{\boldsymbol{h}_i}$ and the backward hidden unit $\overleftarrow{\boldsymbol{h}_i}$. The process of sequence modeling is presented in Figure 4.2 . The chain structure of RNN is effective in capturing the time-evolving interest of a user.

Given this plain RNN architecture, one can make prediction using the last hidden unit, i.e., the output of the last RNN cell $\boldsymbol{h}_m$ in the RNN sequence. The prediction is accomplished with a linear transformation on the hidden output

$$\hat{\boldsymbol{y}} = softmax(\Theta \boldsymbol{h}_m + \boldsymbol{\gamma}) \tag{4.4-2}$$

where $\Theta \in \mathbb{R}^{n \times d_h}$ and $\boldsymbol{\gamma} \in \mathbb{R}^n$ are the parameters we need to learn. Figure 4.2 illustrates the predictive modeling with plain RNN. This design resemblances the architectures used in [35, 108].

4.4.2 Neural Attentive Framework

Although sequence data can be naturally handled by recurrent models, a user's general interest is not adequately captured. Inspired by the successful application of attention mechanisms in other domains, e.g., machine translation [7, 64, 115], we assume that *all* historical user's actions (behaviors) are important for a comprehensive depiction of a user's general interest over time. Nevertheless, we need to put different emphasis on her actions over time as some are less informative than others. We thus propose a general attentive neural model for our problem. Similar to the biGRU model introduced in Section 4.4.1, our approach here is to firstly model the sequence

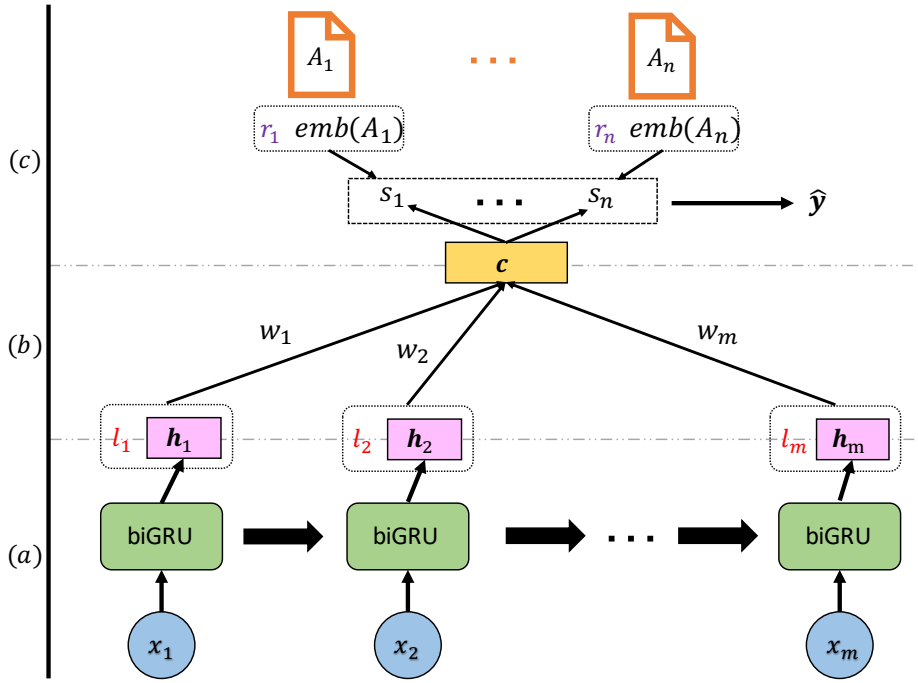


Figure 4.3: The architecture of recency regularized attentive neural model has three layers: layer (a) is the sequence modeling, layer (b) is the attention mechanism, and layer (c) is the recency regularized decoding.

data using biGRU. This is illustrated in Figure 4.3(a). However, instead of relying on the hidden unit for the last cell ($\boldsymbol{h}_m$) to make prediction, the attentive model is capable of learning the attention weight of each hidden unit and integrate them. Figure 4.3 (b) depicts this process. The attention weight for hidden unit $\boldsymbol{h}_i$ is denoted by w_i. And w_i is calculated via a function of hidden units and associated temporal variables, like *lag*. It is formulated as

$$w_i = \frac{\exp\left(attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda)\right)}{\sum_{j=1}^{m} \exp\left(attn(\boldsymbol{h}_j, \boldsymbol{h}_m, l_j; \Lambda)\right)} \tag{4.4-3}$$

where Λ is a set of parameters (that need to be learned), $attn(\cdot)$ is the function for calculating the attention score, and the attention weight is the normalized attention score such that $\sum_{i=1}^{m} w_i = 1$. We present more details about their realizations in the following sections. Given the attention weights, the hidden units from all time steps can be weight aggregated into a context vector $\boldsymbol{c} \in \mathbb{R}^{d_h}$. It is expressed as

$$\boldsymbol{c} = \sum_{i=1}^{m} w_i \boldsymbol{h}_i \tag{4.4-4}$$

Then, we can use the resulted context vector for output prediction. Under this framework, we propose two different designs of attention mechanism by exploiting the temporal variables extracted from the temporal characteristics in news recommendation.

Lag-Aware Attention

Here, we describe our approach in modeling the temporal dynamics of a user's irregular commenting activity. The influence of user's commenting behavior on the final prediction is made not only by the content and sequences of comments, but also the time of commenting events. The dynamic values of same commenting contents varies according to time. Intuitively, given two events that occur at the same sequential position in two different sessions, the older event (i.e., with larger lag from the target event) of the two should be given lesser importance in the prediction task than the more recent event (i.e., with smaller lag from the target event). Based on this intuition, the relevance of a historical event need not be decided by its position in the sequence of a session, but it needs to be quantified by the lag between its commenting time and the commenting time of the target event. An example of lag is illustrated in Figure 4.1 and explained in Section 4.3. In particular, we propose to model the lag-aware attention as

$$attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda) = a * l_i + b \tag{4.4-5}$$

where $\Lambda = \{a, b | a \in \mathbb{R}, b \in \mathbb{R}\}$ is the set of parameters to learn. As we expect that a user's current interest is more affected by her recent actions (small lag) and less affected by her older actions (large lag), the learned a needs to be negative, so

that w is monotonically decreasing with l. We do not add any box constraints to a in our optimization algorithm as we want to prove this hypothesis by learning from the data. (We will show empirically that this hypothesis is strongly supported by the data. The empirical evidence is presented in Section 4.5.7.) The design for lag-aware attention is formulated as

$$w_i = \frac{\exp\left(a * l_i + b\right)}{\sum_{j=1}^{m} \exp\left(a * l_j + b\right)} \tag{4.4-6}$$

We notice from Equation (4.4-6) that the lag of the last event l_m does not affect the attention score. The gap between lags, i.e., $l_i - l_m$, however matters. In this way, we can guarantee that the last event is sufficiently relevant to the target event, while the contributions of other historical events are still weighted by their lags to the last event. Although b is omitted in Equation (4.4-6), we keep it in Equation (4.4-5) as it can increase the numerical stability by avoiding the explosion in the learning process. We also try modeling the attention weight via $w_i = sigmoid(a * l_i + b)$. But this model performed worse in our experiments, because the attention weights are not normalized. We omit it in the rest of the paper.

History-Aware Attention

In addition to the lag-aware attention, we propose another realization of the attention mechanism in which the history-aware attention weight for event i is decided by the joint effort of itself and the last event. Similar designs have been successfully applied elsewhere [64, 54]. The attention function is formulated as:

$$attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda) = \boldsymbol{u}^T tanh(V_h \boldsymbol{h}_i + V_l \boldsymbol{h}_m) \tag{4.4-7}$$

where $\Lambda = \{\boldsymbol{u}, V_h, V_l | \boldsymbol{u} \in \mathbb{R}_a^d, V_h \in \mathbb{R}^{d_a \times d_h}, V_l \in \mathbb{R}^{d_a \times d_h}\}$ is the set of parameters to be learned. We set d_a the same as the hidden size d_h for ease of hyperparameter tuning.

Table 4.1: Datasets.

	#events	#articles	#distinct users	#aug. train sess	#train sess	#val sess	#test sess	avg length
DM	198,272	1,577	30,673	122,625	47,212	4,784	4,773	3.49
Fox	338,380	1,040	28,561	213,738	64,164	8,017	8,018	4.22
WSJ	74,389	1,544	6,353	44,448	15,016	1,864	1,866	3.97

The attention weight is calculated according to Equation (4.4-3).

4.4.3 Recency Regularized Decoder

In this section, we present how we capture the temporal dynamics related to news articles: users prefer to read and comment on recent articles given the short lifespan of news articles.

Efficient Output Decoding

A potential issue in the output layer of recurrent models is the low efficiency of parameter learning in terms of both time complexity and space complexity. As we can see from Equation (4.4-2), the number of scalar parameters in Θ is $n \times d_h$, which can easily become unmanageable with a large set of news articles. Similar issues are observed in other applications [69, 108]. To address this problem, we propose a generalized inner product for output prediction. We calculate the prediction score via the generalized inner product of article embedding and the context vector. The formula is as follows:

$$s_j = \langle \boldsymbol{c}\Omega, emb(A_j) \rangle \tag{4.4-8}$$

where $s_j \in \mathbb{R}$ is the prediction score, $\Omega \in \mathbb{R}^{d_h \times d_e}$ is the projection matrix for the context vector, and $emb(A_j) \in \mathbb{R}^{d_e}$ is the embedding vector for A_j, and d_e is embedding size. In this way, the number of scalar parameters is reduced from $n * d_h$ to $d_e * d_h$ as $n \geq d_e$.

Recency Regularization

News articles are typically short lived with a rapid decline in audience interest– as the old saying goes, "today's news is wrapping tomorrow's fish and chips". Without considering the recency of news articles, a user is equally likely to comment on multiple articles with similar contents. However, a more recent article is more likely to receive a comment than an older article is. In other words, the recency of an article needs to be taken into consideration for prediction. An example of the recency (r) is illustrated in Figure 4.1 and explained in Section 4.3. The recency is set to 0 if the article has not been published yet. In particular, the recency regularizor is formulated as

$$reg(r_j) = \begin{cases} sigmoid(\alpha)^{r_j}, \text{ if } r_j \neq 0 \\ 0, \text{ if } r_j = 0 \text{ (not published yet)} \end{cases} \tag{4.4-9}$$

where $\alpha \in \mathbb{R}$ is the parameter to learn, and $reg(\cdot)$ is the regularization function. To ensure that the base of this exponential function falls into the interval $(0, 1)$, we model it via $sigmoid(\alpha)$ instead of learning it directly. With this approach, more penalties are placed on older articles (with large recency) – $sigmoid(\alpha)$ is small. The recency regularized output scoring function is given by

$$s_j = reg(r_j) * \langle \boldsymbol{c}\Omega, emb(A_j) \rangle \tag{4.4-10}$$

Lastly, the prediction is computed with

$$\hat{\boldsymbol{y}} = softmax(\boldsymbol{s}) \tag{4.4-11}$$

where $\boldsymbol{s} \in \mathbb{R}^n$ is the vector concatenation of $\{s_j | j = 1, \cdots, n\}$. Figure 4.3 (c) depicts the entire process of output decoding.

4.4.4 Model Learning

Up to this point, we described our approach of incorporating the temporal variables into the designs of the attention and decoder. We also described the entire data flow of the proposed method from input to prediction. As the recommendation is essentially a multi-class classification problem, we use the negative log-likelihood loss function as the objective. This is given by

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\boldsymbol{y}^T \log(\hat{\boldsymbol{y}}) \tag{4.4-12}$$

This objective function is optimized using the adaptive moment estimation algorithm (Adam) [44].

Let RHAM denote our proposed recency regularized history-aware attention model and RLAM denote the recency regularized lag-aware attention model. Their non-regularized variants are denoted by HAM - history-aware attention model and LAM - lag-aware attention model, respectively.

4.5 Experimental Results

In this section, we present the details of our experimental settings and results. Our main goal is to demonstrate the effectiveness of our proposed approaches against a set of representative baselines. We will also give an in-depth analysis of our proposed methods.

4.5.1 Datasets

Our data consist of news articles and their user comments from three major news outlets: Daily Mail (DM), Fox News (Fox) and Wall Street Journal (WSJ). We crawled the data from July 2015 to February 2017. We removed mediator comments, which are transformed from spam, hate and meaningless comments detected

by filtering systems, e.g. "System generated comment," or "This comment was removed by a moderator because it didn't abide by our community standards". We also removed the news articles with fewer than 5 comments. We break reader commenting sequences into sessions such that all neighboring events from the same session take place within 7 days. It is because we observe that 80% of neighboring events happen within a week regardless of news outlet. We exclude sessions of length 1 and remove the tail events from sessions with more than 10 events. We sort the sessions in ascending order of the comment time of their first event. We use the first 80% of the sessions for training, then uniformly sample half from the remaining sessions for validation, and use the final 10% of sessions for testing. To effectively utilize the label information in the training data, we augment it according to the procedure in [108]. Table 4.1 provides the statistics of the data used in our empirical studies.

4.5.2 Baseline Methods

We compare our method against the following methods:

POP: It always recommends the most popular articles in the training set [63].

Item-KNN: It recommends the most similar article to the article commented in the last event. The similarity is defined as the co-occurrences of two articles in the training set divided by the square root of the product of the number of sessions in which each articles occurred. Regularization is also included to avoid high similarities of rarely commented articles [20, 60].

A2V-KNN: It is similar to Item-KNN. The difference is that similarity is defined as the cosine of article embedding vectors (A2V), which is calculated as the aggregation of word embeddings from pre-trained Google news corpus weighted by the normalized frequencies of words in the article.

STAN: STAN is the state-of-the-art neighborhood-based approach for session-based recommendations. It improves session-based KNN approach by introducing 3 types

of decay factors according to the timestamp and sequence of sessions and items [29]. We follows its original setting of computing similarity based on co-occurrence of items and 3 factors.

GRU4Rec+: GRU4Rec is an GRU model for session-based recommendation [35], which employs ranking-loss functions in a session-parallel mini-batch training process. GRU4Rec+ improves GRU4Rec model by considering temporal changes and augmenting the training data [108].

NARM: It is a RNN based neural attentive model which integrates a local encoder and a global encoder to jointly model user's sequential behavior and capture the user's main purpose [54].

STAMP: It is a multilayer perceptron based neural attentive model that captures both users' long-term and short-term memory for users' general and current interests [61].

tLSTM: tLSTM is Time-LSTM that models users' sequential actions. It explicitly models the effect of time interval between user's neighbor actions by time gates [144].

CHAMELEON: CHAMELEON [71] is a news session-based recommendation system. It has a metadata and text based article representation learning module and an LSTM-based next article recommendation module.

In summary, we select 8 session-based recommendation algorithms that can be used with our user news commenting datasets. Among them, POP, Item-KNN, A2V-KNN and STAN are non-deep learning methods; the rest are deep learning methods. They cover different types of neural network commonly used in recommendation, including RNN, LSTM, GRU, and MLP. As stated in Section 4.2.2, there are also a few session-based approaches for news recommendation. We implement CHAMELEON, a recent news recommendation algorithm based on RNN. We believe that our set of baselines is representative of the current spectrum of approaches in this field.

4.5.3 Experimental Setup

Evaluation Metric. We use the traditional evaluation metrics for recommendation systems:

- Recall@k: It is the proportion of testing sessions in which the desired target article is among the top k predicted articles in all testing sessions.
- MRR@k: It is the average of reciprocal ranks of the desired target articles in predictions of all testing sessions. The reciprocal rank is set to 0 if it is above k.

Hyperparameters. We tune the hyperparameters with the validation set on the best recall@k during model training and use them on the testing set. The tuned hyperparameters include: the input article embedding, which is either one-hot embedding or dense embedding learned within the model; d_h - the hidden unit size, $d_h \in \{100, 500, 1000\}$; the learning rate for Adam, which is set to one of $\{5e^{-2}, 1e^{-2}, 5e^{-3}, 1e^{-3}\}$; and the index of epoch with the best performance. When choosing dense embedding for input, we tune embedding size (d_e) from $\{100, 300, 500, 1000\}$. We use the bidirectional model with one hidden layer without dropout regularization.

4.5.4 Effectiveness Evaluation

To demonstrate the effectiveness of our proposed recency regularized methods, we compare them against 8 baselines described in Section 4.5.2 on data from three news outlets described in Section 4.5.1. We use Recall@k and MRR@k to evaluate performance. $k = 20$ is a common choice in studies about session-based recommendation [35, 108, 54, 61, 144]. $k = 5$ is a much harder setting than $k = 20$, which can be used to evaluate the lower bound performance. We evaluate all methods using both settings. Table 4.2 presents the outcome for $k = 20$ on the left-hand side and the

outcome for $k = 5$ on the right-hand side. The results of the best two methods are marked in bold. Using the results in Table 4.2, we make the following observations:

(1) Our recency regularized methods (RLAM and RHAM) consistently outperform all other methods by at least 40% (in some cases by more than 100%) across all experimental settings on all datasets in terms of both Recall and MRR (marked in bold).

(2) Both RLAM and RHAM achieve at least 0.197 in terms of MRR on all datasets with different k. The MRRs of the baselines are less than 0.2 in most experimental settings. This indicates that, on average, the predictions of our methods fall within the top-5 more often than those of the baselines.

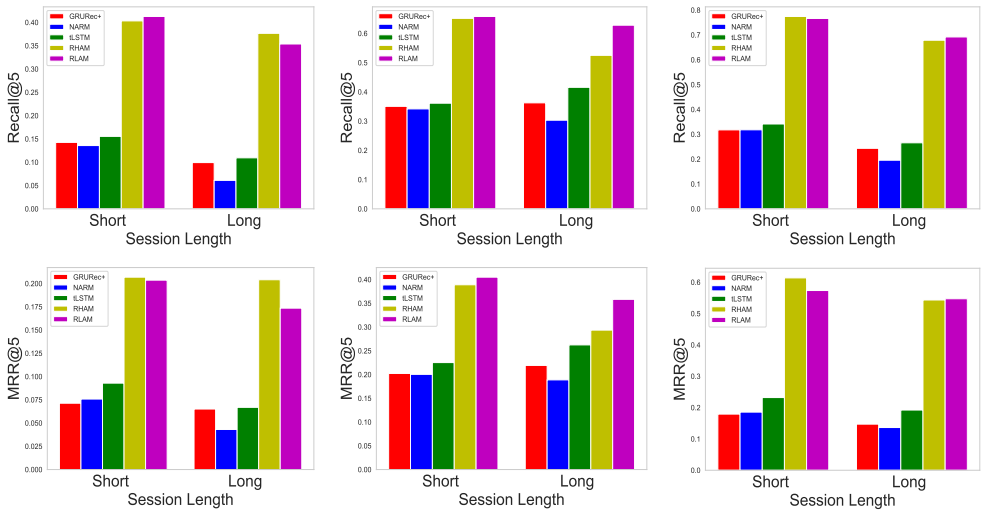
(3) Among all deep learning baselines, GRURec+ performs the best for $k = 20$, whereas tLSTM performs the best for $k = 5$. $k = 5$ is a more challenging criterion. It is likely that the explicit modeling of time factor provides tLSTM with some advantage over the other algorithms. It appears to be able to identify more relevant user actions (which helps when k is small) than the other deep learning algorithms, and penalize more irrelevant user actions, most of which are old actions. Overall however, the results of the RNN based methods are close to each other.

(4) STAMP reports better performance than RNN based methods on general session-based recommendation [61]. However, it performs worse in our experimental results. It also requires about 100 epoches of training to converge in our experiments, while other RNN based methods need around 30 epoches to converge. Given these observations and the fact that STAMP is not a RNN based approach, the drop in its performance on news recommendation may be explained by its inability to model a user's sequential behavior. This however is where the RNN based methods excel.

(5) Though CHAMELEON is a RNN based model designed for news recommendation, it is not the best performing algorithm among the RNN-based baselines for $k = 5$. It performs worse for $k = 20$. This may be due to the lack of metadata (e.g.,

categories) as input for article representation learning.

(6) The non deep learning methods achieve inferior results to deep learning approaches in our problem. Although these methods are able to model the similarities between news articles, they could not leverage sequential signals in the data as effective as deep learning approaches, and they fail to capture user-article-comment connections.



(a) WSJ (b) DM (c) Fox

Figure 4.4: Recall@5 and MRR@5 on different session lengths.

4.5.5 Performance on Different Session Lengths

According to the experimental results in the previous section, RNN based methods generally surpass the other methods. We report a study here about the performance of RNN based methods on sessions with different lengths. We divide the testing sessions into two groups: short sessions, whose lengths are up to 4, and long sessions, whose lengths are greater than 4. We choose 4 because is the average session length (Table 4.1). The results are evaluated using Recall@5 and MRR@5 on identical settings to the experimental study reported in the previous section. We present them in Figure 4.4. We observe that: (1) our proposed methods perform consistently better than

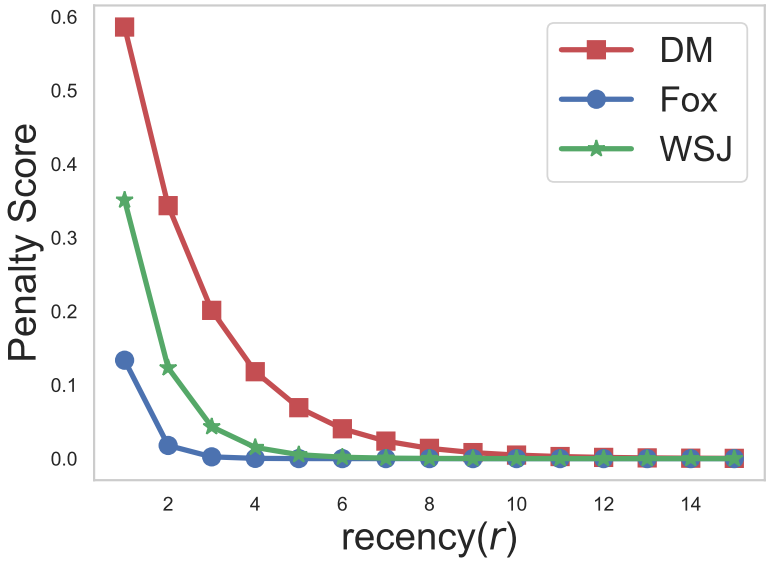


Figure 4.5: Effect of the recency.

the other RNN based methods in both session groups. This shows that our methods are not as sensitive to the session lengths as the baselines, and their advantage over baselines is not dependent on the choice of session length. (2) Overall, all methods perform better on shorter sessions than on longer sessions. The intuition is that user's current interest is more likely influenced by her most recent actions. The accuracy decreases for longer sessions because the prediction is negatively influenced by the noise from the older historical user actions.

Table 4.2: Effectiveness evaluation using Recall@k and MRR@k, $k \in \{20, 5\}$. The results of two best methods are marked in bold.

Method		$k = 20$		
		Daily Mail	Fox	WSJ
		Recall / MRR	Recall / MRR	Recall / MRR
Non DL	POP	0.000 / 0.000	0.019 / 0.002	0.000 / 0.000
	A2V-KNN	0.031 / 0.013	0.090 / 0.034	0.050 / 0.010
	Item-KNN	0.436 / 0.111	0.352 / 0.105	0.206 / 0.062
	STAN	0.471 / 0.119	0.358 / 0.096	0.233 / 0.066
DL	GRU4Rec+	0.571 / 0.227	0.477 / 0.193	0.290 / 0.086
	NARM	0.555 / 0.222	0.450 / 0.193	0.220 / 0.079
	tLSTM	0.533 / 0.246	0.422 / 0.236	0.247 / 0.098
	STAMP	0.508 / 0.106	0.361 / 0.073	0.228 / 0.060
	CHAMELEON	0.570 / 0.230	0.466 / 0.197	0.276 / 0.096
Ours	RLAM	**0.867 / 0.426**	**0.824 / 0.579**	**0.645 / 0.224**
	RHAM	**0.894 / 0.410**	**0.857 / 0.615**	**0.674 / 0.238**
Method		$k = 5$		
		Daily Mail	Fox	WSJ
		Recall / MRR	Recall / MRR	Recall / MRR
Non DL	POP	0.000 / 0.000	0.014 / 0.002	0.000 / 0.000
	A2V-KNN	0.013 / 0.011	0.085 / 0.033	0.015 / 0.007
	Item-KNN	0.202 / 0.088	0.182 / 0.088	0.108 / 0.052
	STAN	0.203 / 0.093	0.162 / 0.078	0.101 / 0.053
DL	GRU4Rec+	0.351 / 0.203	0.304 / 0.173	0.133 / 0.070
	NARM	0.339 / 0.200	0.296 / 0.177	0.120 / 0.069
	tLSTM	0.365 / 0.228	0.328 / 0.225	0.146 / 0.087
	STAMP	0.215 / 0.100	0.142 / 0.059	0.104 / 0.048
	CHAMELEON	0.344 / 0.207	0.317 / 0.181	0.149 / 0.084
Ours	RLAM	**0.656 / 0.402**	**0.753 / 0.569**	**0.400 / 0.197**
	RHAM	**0.643 / 0.383**	**0.757 / 0.601**	**0.398 / 0.206**

Table 4.3: Recall@5 and MRR@5 on three news outlets.

		Daily Mail		Fox		WSJ	
k	Method	Recall	MRR	Recall	MRR	Recall	MRR
5	LAM	0.352	0.202	0.301	0.166	0.133	0.072
	HAM	0.360	0.209	0.309	0.175	0.143	0.080
	Joint	0.364	0.216	0.310	0.188	0.137	0.071
	RLAM	**0.656**	**0.402**	**0.753**	**0.569**	**0.400**	**0.197**
	RHAM	**0.643**	**0.383**	**0.757**	**0.601**	**0.398**	**0.206**
10	LAM	0.472	0.219	0.427	0.185	0.220	0.085
	HAM	0.480	0.226	0.405	0.188	0.207	0.089
	Joint	0.476	0.231	0.414	0.190	0.205	0.080
	RLAM	**0.788**	**0.420**	**0.818**	**0.579**	**0.549**	**0.217**
	RHAM	**0.801**	**0.404**	**0.849**	**0.614**	**0.563**	**0.230**
20	LAM	0.566	0.225	0.533	0.193	0.287	0.089
	HAM	0.577	0.232	0.468	0.193	0.278	0.094
	Joint	0.571	0.216	0.488	0.207	0.271	0.085
	RLAM	**0.867**	**0.426**	**0.824**	**0.579**	**0.645**	**0.224**
	RHAM	**0.894**	**0.410**	**0.857**	**0.615**	**0.674**	**0.238**

4.5.6 Effect of Recency Regularizor

In this section, we study how recency affects the penalty on prediction scores. We plot the penalty score $reg(r)$ against recency r from 1 to 15 using the learned α from the best RLAM model used in our experiments. The calculation of $reg(r)$ is explained in Equation (4.4-9). Figure 4.5 clearly shows that recency is a key factor deciding the value of articles across all outlets. The predictions for fresh articles (with less recency) are less penalized, while penalty is larger for older articles. Moreover, we observe that the effect of recency to the penalty is quite different on different outlets. Specifically, the effect of the recency to the penalty is decided by the base $sigmoid(\alpha)$ in Equation (4.4-9). In our results, the bases calculated from the learned α are 0.586, 0.134 and 0.351 for Daily Mail, Fox and WSJ, respectively. It shows that recency is more important to the value of articles in Fox than in the other outlets. This appears to be due to increased daily reporting on the U.S. presidential election in 2016 at Fox.

Previously, we presented the effectiveness of our proposed recency regularized neural models, but without knowing the contribution of each module. Here, we conduct an ablation study to explore the contribution of plain attentive neural models. In particular, we repeat the same experimental setting in Section 4.5.4. The results are presented in Table 4.3. From the results, we conclude that: (1) After removing the recency regularizor from RLAM and RHAM, the performance of the non-regularized methods (LAM and HAM) drops across all experimental settings. This observation demonstrates the importance of modeling the dynamics of recency in our problem. (2) Although HAM and LAM are not as good as recency regularized models, they however are still comparable with other deep learning approaches (if we compare the results from Tables 4.3 and 4.2). They are also interpretable given their attention mechanism. We specifically discuss the effect of lag-aware attention proposed in LAM in Section 4.5.7. Regarding the attention in HAM, one can also visualize their weights for further investigation. Exploration on similar attention is discussed in

Table 4.4: Weight of lag in lag-aware attention.

outlet	Daily Mail	Fox	WSJ
a	-0.745	-0.526	-0.388

other studies [64, 54]. We do not discuss it further in this paper. (3) In addition, we also investigate the effect of a hybrid attention model which integrates HAM and LAM by concatenating their context vectors. We call this approach Joint and present its performance in Table 4.3. Analyzing the results in the table we note that the joint model is not always better than its two modules due to its heavier modeling. Hence, we do not experiment further with this approach in this work.

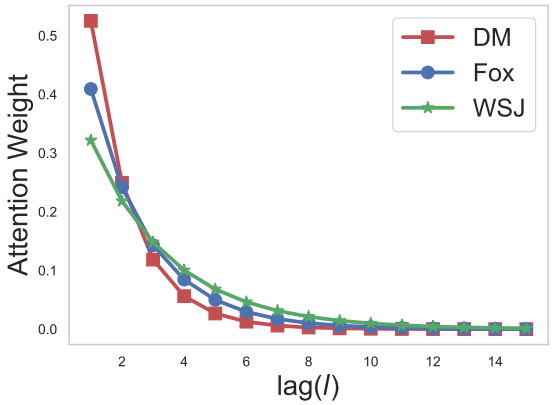


Figure 4.6: Effect of the Lag

4.5.7 Interpretable Lag-Aware Attention

In this section, we study the effect of lag l to Lag-Aware Attention weight w. We firstly want to demonstrate the soundness of our hypothesis that *user's recent actions are more relevant to her current interest*. In other words, the parameter a in Equation (4.4-6) learned from the data should be negative, such that the attention weight is negatively correlated with the lag. For instance, recent actions with small lags have larger attention weights. We present the learned a from three datasets in Table 4.4. They are all negative and they are learned without adding any box constraints. We

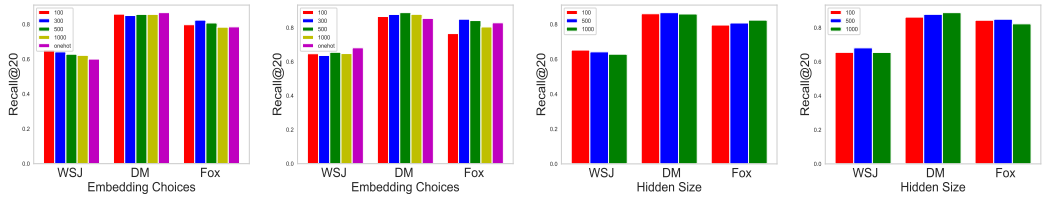


Figure 4.7: Effect of embedding choices and hidden size on RLAM and RHAM.

contend that this asserts the soundness of our hypothesis.

To understand the effect of lag-aware attention weight better, we create a synthetic session with event lags ranging from 1 to 15. We plot the attention weight w against lags l with a learned from the best LAM models (Figure 4.6). We observe from the figure that more recent events are associated with larger weights in all 3 outlets.

4.5.8 Effect of Hyperparameters

In this section, we discuss the effect of hyperparameters (embedding choices and hidden sizes) on our proposed RHAM and RLAM. For ease of representation, we only report their results in Recall@20. For other hyperparameters, we tune their optimal choices before this experiment. In particular, we initialize our models with an one-layer bidirectional GRU model, as deeper architectures perform worse. The optimal learning rate is 0.001 when using ADAM since larger learning rates may cause divergence. The embedding is onehot embedding and its size is chosen from $\{100, 300, 500, 1000\}$. Hidden size is chosen from $\{100, 500, 1000\}$. Figure 4.7 summarizes the outcome of this experiment. We observe that the optimal choices vary across datasets. For instance, small-scale datasets like WSJ require fewer parameters (smaller embedding and hidden sizes) to learn in the model, while larger datasets like DM require more parameters. For WSJ, the best embedding size is 100 and the best hidden size is 100. The best setting for RLAM on Daily Mail are onehot embedding

and a hidden size of 500.

4.6 Conclusion

We tackle the problem of predicting the news article a user will comment on given her historical sequential behavior from an anonymous session. We treat the problem as an instance of the session-based news recommendation problem. Unlike the general session-based recommendation problem, we observe unique temporal dynamics in reader's commenting activities and news articles in our problem. To capture them, we propose a recency regularized neural attentive framework that seeks to model the temporal variables observed from the time-varying interests of a user, like irregular commenting user habit and short-lived relevance of news articles. We demonstrate the effectiveness of our approach, in particular that of the recency regularized model, in comparison with state-of-the-art methods on real-world data from three major news outlets. In addition, we show that our approach is interpretable in terms of understanding (i) the effect of temporal variables in the prediction outcome and (ii) their varied behavior across news outlets.

CHAPTER 5

FEATURES DESIGN AND LEARNING FOR NEWS ARTICLE: CANNOT PREDICT COMMENT VOLUME OF A NEWS ARTICLE BEFORE USERS READ IT

In this chapter, we perform extensive empirical studies about news outlets and news categories, and we design and learn various features for news commenting prediction task. Many news outlets allow users to contribute comments on topics about daily world events. News articles are the seeds that spring users' interest to contribute content, i.e., comments. An article may attract an apathetic user engagement (several tens of comments) or a spontaneous fervent user engagement (thousands of comments). In this paper, we study the problem of predicting the total number of user comments a news article will receive. Our main insight is that the early dynamics of user comments contribute the most to an accurate prediction, while news article specific factors have surprisingly little influence. This appears to be an interesting and understudied phenomenon: collective social behavior at a news outlet shapes user response and may even downplay the content of an article. We compile and analyze a large number of features, both old and novel from literature. The features span a broad spectrum of facets including news article and comment contents, temporal dynamics, sentiment/linguistic features, and user behaviors. We show that the early arrival rate of comments is the best indicator of the eventual number of comments. We conduct an in-depth analysis of this feature across several dimensions, such as

news outlets and news article categories. We show that the relationship between the early rate and the final number of comments as well as the prediction accuracy vary considerably across news outlets and news article categories (e.g., politics, sports, or health).

5.1 Introduction

Commenting on news is a common form of participation in contemporary news consumption, and it is one of the most common forms of citizen engagement online [26]. A key indicator of user participation in daily news events is the volume of user comments reacting to a news article [82, 147]. Several works propose methods to predict it [111, 8]. They use a large number of features, which can be broadly categorized into article content, meta-article (e.g., outlet or category), temporal (e.g., date and time of publication), and semantic (e.g., named entities). They model the prediction problem as a classification problem. For example, they determine if an article will receive a "high" or "low" volume of comments. One of their key findings is that *(i) predictors based on article content features alone are the best performers and (ii) one may even achieve high accuracy with such predictors.*

The social science community, in particular the Communication community, argues that quality discourse emerges only when many users participate in commenting on a news article and when there is interactivity among users, i.e., users comment/reply to prior comments [45]. The general questions pursued in this space aim to understand the factors affecting participation and interactivity in the comment section of an article [2, 122]. Some studies (via face-to-face interviews) show that factors from previously posted user comments affect the involvement of new users and ultimately increase users' willingness to engage in online news discussions [66, 145]. *They conclude that a large fraction of the comments an article receives– up to 50%– do not respond to the journalistic value of a news article, but rather to a previously*

posted user comment [99, 91].

Our work in this paper is motivated by the apparent disagreement between the findings from different communities. We aim to understand the factors– ranging from article content to observed dynamics of user comments – on predicting the eventual comment volume an article receives. One may notice a problem here. On the one hand, one would like to predict the comment volume before an article's publication. On the other hand, one has access to user comments only after the article has been online for some time. Nonetheless, the number of eventual total comments an article will get remains relevant. Thus, we relax the problem by formulating it as follows:

Problem: Given a news article A and its first α user comments, predict N_A, the eventual number of comments A receives.

We can draw a parallel to the problem of predicting the distance traveled by a ball (news article), say in soccer. The distance depends on the ball itself and the person who kicks it (news outlet and author), but it also depends on factors, like launch angle and exit speed, unrelated to the ball. Those are only known shortly after the ball was kicked and traveled a short distance. Similarly, we expect the first α comments to give us the missing information necessary to predict the eventual number of comments the article receives.

One may notice that N_A is not well defined: theoretically, it may continue to grow endlessly with time. In practice, however, this does not happen for news articles. We monitored each article for 3 months. The articles accumulated 99.84% of their overall comment volumes within a week and 99.97% within a month. *Consequently, hereafter N_A is the number of comments accumulated in the first week by news article A*, which empirically is almost identical to the true total number of comments that A receives in practice.

The magnitude of α and its relation to N_A may trivialize the problem, say, "look at the first $\alpha = 1{,}000$ user comments and predict if the article will receive $N_A = 1{,}050$."

We study the dynamics among the very first few comments and aim to predict if N_A will reach 1,050. We empirically test $\alpha = 5$, 10, 15, 20, and 50. The accuracy of prediction increases by about 9% from $\alpha = 5$ to $\alpha = 10$, and by less than 2% from $\alpha = 10$ to $\alpha = 50$. We set $\alpha = 10$ in all our empirical studies. Thus, we aim to predict the eventual number of comments an article will receive based on the observations among the first 10 comments. It takes 63 minutes on average for the 10^{th} comment to arrive since the posting of the first comment.

After evaluating and comparing the prediction performance of various models on 19K articles and over 9M comments from 6 news outlets, we show that signals gathered from the early dynamics of user comments largely influence the ability to predict the eventual number of comments on a news article, while the contribution from article features is small. This finding is consistent with the conclusion from the social science community. We study the user comment features and identify the feature of "the arrival rate of *early* comments" (`rate`), which is defined as the number of comments per minute, as a key missing link in accurately predicting the comment volume.

We study `rate` across six major U.S. and U.K. news outlets. We notice that the performance of the rate-based model varies across news outlets. It is highly accurate for news articles published by Wall Street Journal, but less accurate for Fox News and the Guardian. With regard to the analysis of `rate` by categories, we note that the characteristic of the rate model differs across categories. For example, "Politics" is particularly sensitive to the rate of early comments. This is common across all news outlets. "Health," on the other hand, is less sensitive to the rate of early comments.

We also consider the relationship between news outlets and categories. We study the characteristic of `rate` model for each outlet-category pair. We find that the rate model performs the best at Wall Street Journal in most of the categories, and the eventual user activity is more affected by the initial engagement in political areas

across all outlets.

We believe that our findings are of interest to social scientists because they reveal the relationship between the early user commenting behavior and the total comment volume of a news article, across news outlets and news categories. *This appears to be a trait unique to news readership communities. Features based on early arrival pattern in other social communities, such as Twitter and Facebook, on related prediction tasks have limited predictive power* [6, 124]. Network topology features are more effective in those tasks; most of those features are not applicable to news readership communities as they lack an underlying network.

We make the following contributions in this paper:

- We postulate that one cannot predict the comment volume of an article unless one considers (early) user commenting activity.
- We identify the importance of (early arrival) `rate` in the task of predicting the comment volume of a news article.
- We perform extensive empirical studies by news outlets and news categories, and show additional novel insights.

5.2 Related Work

We review several lines of research about user generated content in news domain and social networks.

News Domain. Mining and analyzing the content produced by users in news media are popular research directions. Some of the explored problems include examining the relationships between news comment topicality, temporality, sentiment, and quality [23]; analyzing the sentiment of comments and headlines of news article [24]; news propagation [107]; personalized recommendation of news stories [96]; topic clustering of news articles [1]; and modeling and predicting comment volume

[112, 8, 89, 109]. The prediction of comment volume is treated as a (binary) classification problem (e.g., "High" "Low" volume) [111] and regression classification problem [8] in previous studies. [109] uses a simple linear regression model with early user activity during a short observation period after publication to predict the comment volume of articles. Besides, [3] describes few current models of the growth of comment threads.

Social Networking Platforms. Understanding user behavior in social networking platforms, e.g. Twitter and Facebook, has attracted large interest. Some studies aim to understand user conversations and their evolution over time [117], while others study the commenting and comment rating behavior [97]. A number of works address problems related to the prediction of reply volume. They employ a variety of features, such as bag of words [137], the arrival patterns of early comments [6], network specific, like "followship," and historical behavior in retweet [4]. The prediction problem itself is modeled in a variety of ways. Some model it as a binary classification problem [4, 6]. Other formulations include regression [114], multi-label classification [124], cascades size prediction [18, 46], self-exciting point process [68], or Hawkes process modeling [140, 88]. The popularity prediction and modeling on social media is also a fruitful research [56, 67, 59].

Our Work. While our work shares some commonalities with these lines of work, it also distinguishes from them in several important ways. The main difference with the work in the news domain is that we focus on the analysis of (early) user commenting activity and its importance on the prediction of comment volume of a news article. The study of using user comments in early stage to predict the final comment volume has been analyzed in [109], which only evaluates a simple linear prediction model with limited factors from articles and comments. While our work in this paper explores more features related to article and early user commenting activity (both new and old). Moreover, we consider multiple machine learning techniques, both linear and

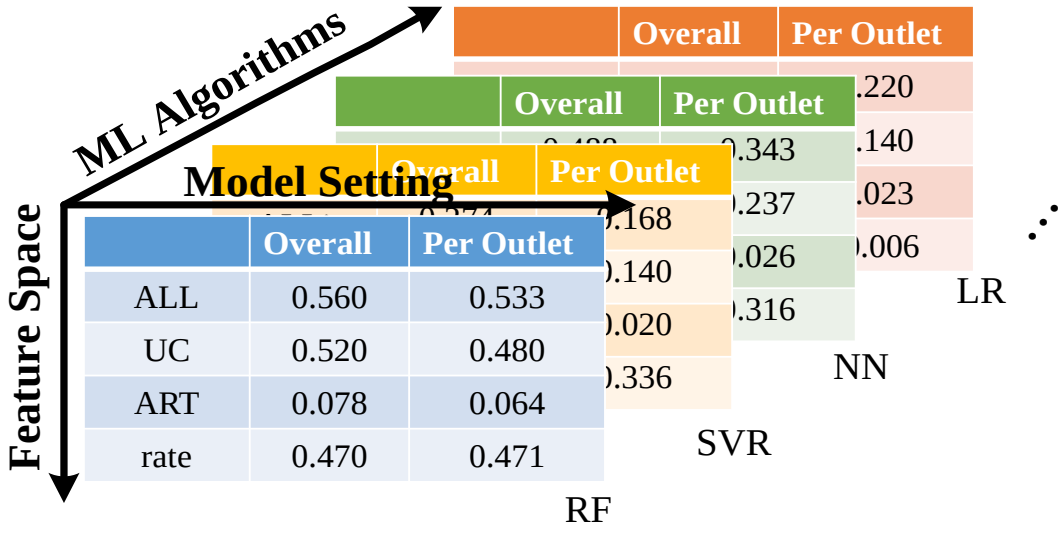
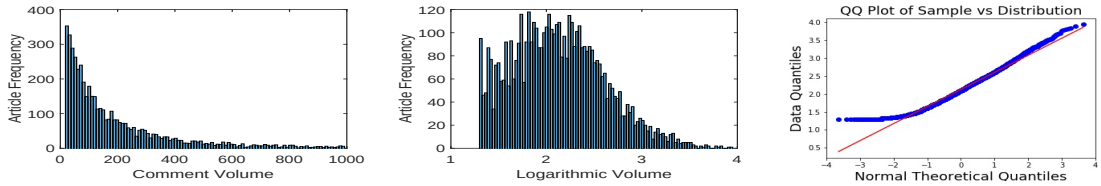
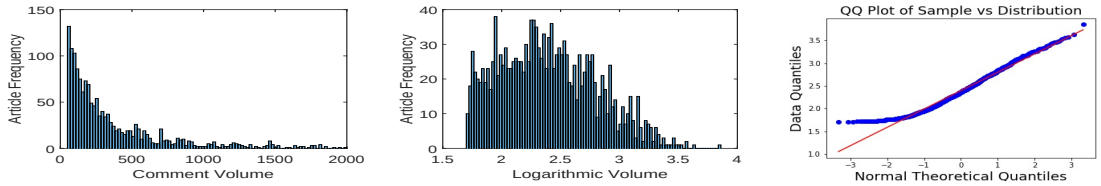


Figure 5.1: Methodology illustration.

nonlinear. According to our analysis, we show that (1) the prediction problem is difficult and, thus, nonlinear models are better suited to solve the prediction problem; and (2) the proposed new feature `rate` remains its dominant power across machine learning techniques. The key distinction with the work in social networks is that the social communities at news outlets are *not* networked. The works in social networks make heavy use of the network topology and the community around a user, e.g., followers and friends. These are not applicable in our setting. While arrival patterns of user posts are considered in previous works, they are not as consequential in their respective prediction tasks as `rate` is in ours. For instance, arrival patterns as defined in [6] contribute less than 4.4% to the overall performance, compared to 90% on average for `rate`. The family of features "growth rate" [124], which includes a feature similar to `rate`, has a much weaker predictive power than that of their other features. Since their `rate`s are inconsequential, these works do not pursue any in depth studies of their `rates`. We present a study of `rate` along several dimensions, such as news outlet and news category.



(a) Daily Mail



(b) the Guardian

Figure 5.2: The distribution of comment volume and logarithmic volume per news outlet. In the first column of graphs, the articles with more than 1,000/2,000 comments are discarded to make the graphs visible. For each outlet, article frequency on the y-axis of the first two graphs is the number of articles, the third graph provide the Q-Q plot of the logarithmic volume.

5.3 Methodology

Our goal is to understand the feature subset most important for predicting the comment volume of a news article. The prediction of the eventual comment volume is a regression problem. Figure 5.1 summarizes our methodology. We conduct our study along three dimensions: (i) Feature Space, (ii) Model Setting, and (iii) Machine Learning (ML) Algorithms. In (i), we analyze the entire feature space (denoted as ALL), the user comment only (UC) and news article only (ART) features, as well as `rate` (which is a single feature in UC) alone. In (ii), we consider two settings: *global* and *local*. The global dataset has the news articles from all the news outlets. The local dataset has the news articles grouped by news outlet. In (iii), we use 4 representative ML algorithms for regression: Random Forest (RF), Support Vector Regression (SVR), Neural Network (NN), and Linear Regression (LR). In the figure, a slice represents an instance from the cross product of (i), (ii), and (iii).

5.4 Data

We collected news articles with comments from Oct. 2015 to Feb. 2017 from the following six news outlets: Washington Post, Daily Mail, Wall Street Journal, Fox News, the Guardian, and New York Times. We crawled the topics of the collected articles from Google News and monitored their duration there as well. The dataset has over 19K articles with comments and 9M comments (including replies). We monitored each article for 3 months. We observed that on average each article accumulates 99.84% of its overall comment volume within a week. Recall that in this paper N_A is the number of comments accumulated in the first week by news article A. This is the number we try to predict.

Figure 5.2 illustrates the distribution of news articles at these outlets in our dataset (we give 2 outlets due to space constraints). We observe a heavy-tailed distribution in each outlet when we plot the distribution by number of comments (the first graph per outlet). If we plot the comment volume in the log scale, the distributions are (or close to be) bell-shaped. It seems that the volume distributions are nearly log-normal. To test this hypothesis, we provide the Q-Q plot of the logarithmic volume as the third column of graphs in Figure 5.2 for each outlet. We can see that most of the points stay on or very close to the straight line, except for some head and tail data points. It shows that the distribution of user comment volume over news articles is well approximated by the log-normal distribution.

Table 5.1 describes the number of articles in each news outlet. We give the mean and standard deviation (STD) of comment volume and logarithmic volume, displayed in the last two columns of Table 5.1. Considering the log-normal distribution of comment volume, we will work on the prediction of comment volume in log scale.

Table 5.1: Data summary. Aw.C = Articles with Comments.

Outlets	**Aw.C**	**Mean Vol. (STD)**	**Mean Log Vol. (STD)**
Washington Post	6,470	364.8 (942.2)	1.88 (0.76)
Daily Mail	6,046	264.1 (560.4)	1.99 (0.62)
Wall Street Journal	2,516	189.4 (346.5)	1.74 (0.7)
Fox News	1,739	1,896.5 (3790.2)	2.47 (0.94)
the Guardian	1,697	504.4 (716.8)	2.46 (0.45)
New York Times	965	481.4 (530.9)	2.38 (0.6)
Overall	19,433	465.8 (1400.6)	2.02 (0.74)

5.5 Predicting Comment Volume

In this section, we show that factors drawn from (early) user commenting activity are the keys to accurately predict the comment volume a news article receives. We describe the feature set and the experimental setting, and report on the prediction performance in this section. We follow the methodology described above. Finally, we demonstrate that `rate` is the dominant feature.

5.5.1 Features

Table 5.2: Features utilized in prediction experiments. We organize them into 5 groups. Here are first 2 groups.

Feature	**Description**
Topic features	
topic*	Topic of article.
Article features	
month	Published month of article (1-12).
day	Published day of the month (1-31).
hour	Published hour of the day (0-23).
wom	Week of the month (1-5).
dow	Day of the week (1-7).
author	Author of article.
art_length	Article content length.
art_question	Whether there is a '?' in article title.
art_exclaim	Whether there is a '!' in article title.
art_num_ne_loc	Number of location-type named entities in article content.
art_num_ne_per	Number of person-type named entities in article content.
art_num_ne_org	Number of org.-type named entities in article content.
art_num_ne_misc	Number of miscellaneous-type named entity in article content.
art_senti_score*	Sentiment score of article content.

Table 5.3: Features utilized in prediction experiments. We organize them into 5 groups. Here are last 3 groups..

Comment features	
rate*	Arriving rate of the first α comments.
fc_mid*	Time of first comment - 12am (in min.)
uniq_com	Number of unique commenters.
num_reply	Number of replies.
num_thread	Number of threads.
num_question	Number of '?'.
num_exclaim	Number of '!'.
num_words	Number of words.
complexity	Complexity of the first α comments.
has_url	Whether there is a link.
num_ne_com	Number of named entities.
depth*	Depth of the comment tree.
width*	Width of the comment tree.
avg_senti_score*	Average of sentiment scores of the first α comments.
num_likes	Aggregated number of likes.
num_dislikes	Aggregated number of dislikes.
News Factors	
continuity*	Time difference between article's publication and its topic's appearance.
aggression*	Fraction of aggressive words.
position	NA.
MISC features	
pub_resp	Time difference (in minutes) of first comment to article's publication.
inter_art*	Defined as $\frac{\lvert NE_{art} \bigcap NE_{com_i} \rvert}{\lvert NE_{art} \rvert}$
inter_com*	Defined as $\frac{\lvert NE_{art} \bigcap NE_{com_i} \rvert}{\lvert NE_{com_i} \rvert}$

Table 5.2 and Table 5.3 summaries the set of features. There are five groups of features: topic, article, comment, news factors, and misc features. We introduce 11 new features.

Topic features. We observe that some topics, such as Ebola Outbreak or Paris (terrorist attack), trigger more discussion than others. Therefore, we include these finer grain topics as one of the predictive features. The fine grain topics are rarely provided by the news outlets. We extract them from Google News along with their parent categories, e.g., Health and World. We collect 768 distinct topics in total.

Article features. All features in this group are related to news articles. They can be categorized into metadata and text features. The metadata features include *month*, *day*, *hour*, *wom* (week of the month), and *dow* (day of the week) of the publication. Previous work argues that the time of publication may affect the comment volume an article receives [112]. The rest of the features in this group are extracted from article title and content. The features *art_question* and *art_exclaim*, suggested in [6], show whether there are '?' and '!' in article title. The features *art_num_ne_loc*, *art_num_ne_per*, *art_num_ne_org*, and *art_num_ne_misc*, proposed in [111], provide the number of locations, persons, organizations, and miscellaneous named entities mentioned in the article content. We utilize Stanford NER to extract named entities.

The feature *art_senti_score* gives the sentiment score of article content. To calculate this sentiment score, we make use of an effective document representation and sentiment analysis model proposed in [136], which is a word-sentence-document level bi-directional GRU neural network with two levels of attention. We initialize the 100 dimensional word embeddings with pre-trained Glove word vectors. The model is trained on IMDB dataset (25K reviews with positive or negative rates) for 10 epochs. We use the output from the prediction layer of the deep model as sentiment score, which is in the range of [0, 1]. Articles with score close to 0 are predicted with overall negative sentiment, while articles with score close to 1 are predicted with overall

positive sentiment.

Comment features. The comment features are extracted from the first α comments of an article. The feature *rate* measures the number of comments per unit of time, which is computed as

$$rate = \frac{i}{t_i - t_1}$$

Here, $t_i - t_1$ is the elapsed time (in minutes) between the first and i-th comment (as in [124]); it is 63 minutes on average for the 10^{th} comment. The feature *fc_mid* is the absolute difference between the time of the first comment and midnight. The feature *uniq_com* gives the number of unique commenters in the first α comments, which is one of the indicators for the arrival pattern of the first α comments [6]. The features *num_reply* and *num_thread* give the number of replies and discussion threads, respectively.

The features *num_question*, *num_exclaim*, *num_words*, *complexity*, *has_url*, *num_ne_com*, and *avg_senti_score* study the text of comments. The meaning of these features are provided in Table 5.2. *Complexity* measures the cumulative entropy of terms within the first α comments [90]. It is given by:

$$complexity(c) = \tfrac{1}{|T(c)|} \sum_{t \in T(c)} tf(t,c)(\log|T(c)| - \log tf(t,c))$$

Here, $T(c)$ is the set of unique terms in comment c and $tf(t,c)$ is the frequency of each term $t \in T(c)$.

The feature *avg_senti_score* is the average of the sentiment scores of the first α comments. The sentiment score of a comment is given by the deep model in [136].

The features *depth* and *width* are extracted from the comment reply tree T_A of an article A. T_A is constructed as follows. An article A is the root of T_A. Comments that are not replies (responses) of any previous comments are the children of A (the article). The replies of a comment are its children nodes. The *depth* of the reply tree

T_A is the number of levels of T_A. If L denotes the levels of the reply tree T_A, the *width* is given by

$$WIDTH = \max_{j \in \mathbf{L}} \sum_{i=1}^{m_j} s_{ji},$$

where m_j is the number of sibling groups in level j, and s_{ji} is the count of nodes in the i-th sibling group in level j. A feature named *depth* appears in [18], but its definition and meaning are different from ours.

The features *num_likes* and *num_dislikes* count the aggregated number of likes and dislikes received by the first α comments. The consideration of number of likes is proposed in [6].

News factors. We implement a number of novel features based upon *news value theory*, which states that journalists and media users select news items depending on news factors such as continuity, negativity, and aggression [122, 145]. These dimensions were confirmed after extensive face-to-face interviews with users who commented on news stories online [146]. We create novel features to quantify many of the news factors. Some of them are encountered in other studies, e.g., climate change [80], but with different definitions. We quantify the factor *continuity* (if a news article continues issues that are already on the media agenda) [122] as the time difference (in minutes) between article's publication time and its topic's appearance in Google News. The intuition is that a user's interest to comment on an article diminishes the farther its publication time is from the time when the news event first broke in. We additionally consider the factors *negativity* and *aggression* both in the article text and user comments. To quantify *negativity*, we calculate the sentiment score of a piece of text (article or comment) by applying an effective document representation and sentiment analysis model proposed in [136]. A piece of text with sentiment score closer to 0 shows stronger negativity, while a text with score closer to 1 indicates stronger positivity. For the sentiment of article and comments, we propose features *art_senti_score* and *avg_senti_score*, which are already present in the group of article

and comment features, respectively. We use the lexicon LIWC [110] to quantify *aggression*. Given a piece of text, *aggression* is defined as the count of entries in the category "Hostile" together with the ones under "anger" that appear in the text. Additional news factors, such as *time of publication*, *uncertainty*, *length*, and *facticity*, are considered in the previous feature groups. They are underlined in the table. Following their definitions [146], *uncertainty* is measured by the count of question marks in a piece of text and *facticity* is a binary feature, which is 1 if the piece of text contains an URL, and 0 otherwise. The factor *position* of comment in the discussion thread is not applicable in our case, since we only analyze the first α comments.

MISC features. The feature *pub_resp* describes how fast users respond to an article, which is similar to some of the features in [6, 18]. Some works argue that the longer users delay their response to an article, the less overall user activity the article receives [6]. The features *inter_art* and *inter_com* quantify the ratio of overlap between the sets of named entities in an article and its first α comments. Let NE_{art} and NE_{com_i} be the sets of named entities that appear in an article and its first α comments, respectively. We define *inter_art* and *inter_com* as

$$inter\_art = \frac{|NE_{art} \bigcap NE_{com_i}|}{|NE_{art}|}$$
$$inter\_com = \frac{|NE_{art} \bigcap NE_{com_i}|}{|NE_{com_i}|}.$$

5.5.2 Experimental Setup

The experimental study employs the cross-validation methodology. We split the dataset into five folds randomly. The training set consists of articles in four folds. The articles in the remaining fold are used for testing. For a given set of features, we build a model based on the training set, and apply it on a disjoint testing set. The process is repeated five times, each time selecting a different fold for testing. We report the average performance.

Evaluation Metrics

We treat the task of predicting the comment volume of a news article as a regression problem. We evaluate each model in the experiments based on R^2 and the mean absolute error (MAE), which are defined as

$$R^2 = 1 - \frac{MSE}{Variance} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

We calculate the MAE instead of MAPE (mean absolute percentage error) since MAE is more robust to outliers (the long tail in the first graph per outlet in Figure 5.2). Target variable y_i in the calculation of R^2 and MAE is the logarithm of the number of comments because the distribution of comment volumes resembles lognormal distribution, as shown in Figure 5.2.

Hyperparameter Tuning and Setting

We consider the first $\alpha = 10$ user comments for each article when we compute the comment features. We reached $\alpha = 10$ after we studied the variation in prediction accuracy for $\alpha = 5$, 10, 15, 20, and 50, respectively. The accuracy of prediction increases by about 9% from $\alpha = 5$ to $\alpha = 10$, and by less than 2% from $\alpha = 10$ to $\alpha = 50$. Therefore, we set $\alpha = 10$ in all our empirical studies. We explore multiple machine learning (ML) algorithms for performance comparisons on the proposed comment volume prediction task as listed in Figure 5.1. The results indicate that the feature `rate` does consistently well across the board, thereby indicating it to be a strong algorithm independent feature that inherently captures the prediction task.

We give a brief overview of the hyperparameter setting for the three nonlinear ML algorithms in our methodology: Random Forest (RF), Support Vector Regression (SVR), and Neural Network (NN). We tune the number of trees (`ntrees`) for RF. We

Table 5.4: Comparison of R^2/MAE results on the overall dataset. ART is the baseline for each algorithm. The highlighted row (ART) gives the outcome of the baselines.

	RF	**SVR**	**NN**	**LR**
ALL	**0.560/0.282**	0.472/0.324	0.499/0.310	0.413/0.338
UC	0.520/0.294	0.479/0.303	0.502/0.301	0.400/0.342
1-5 ART	0.078/0.439	0.021/0.452	0.016/0.459	0.020/0.458
1-5 `rate`	0.470/0.316	0.465/0.311	0.459/0.323	0.370/0.354

use SVR with kernel 'rbf' and tune the hyperparameters `C` and ϵ. We choose Multi-layer Perceptron to implement the NN and tune the hidden layer sizes (`hsize`) and the initial learning rate (`lr`); the activation function for the hidden layer is set to be 'relu'. The choices for these hyperparameters are drawn from: $ntrees \in [50, 100, 200, 300]$, $C \in [0.1, 0.5, 1, 5, 10]$, $\epsilon \in [0.01, 0.05, 0.1, 0.5]$, $hsize \in [10, 20, 30, 50, 100, 200]$, and $lr \in [0.001, 0.005, 0.01, 0.05, 0.1]$.

5.5.3 Experimental Results

We report the performance for the four algorithms (i.e., Random Forest, Support Vector Regression, Neural Network, and Linear Regression) along the four sets of features (i.e., ALL, UC, ART, and `rate`), in the *global* setting in Table 5.4. We omit the outcome with the *local* setting because of the page limitation. But, the conclusion is very similar to the one drawn in the *global* setting.

User Factors Matter

In Table 5.4, the combined use of all features (ALL) along with Random Forest achieves the best accuracy. We observe that the R^2 value for ART in each testing scenario is near zero, suggesting that the article features alone are not useful signals for predicting the comment volume an article will receive. The prediction models yield much better results when the commenting behavior from early users are taken into consideration (e.g., compare UC row to ART row). *This proves that attempting to predict the eventual volume of user comments on the merits of a news article itself*

is a futile endeavour. The reason is that a large fraction of the users post comments are triggered by other users' comments instead of the content of the news article itself [99, 91]. The article features cannot account for the user commenting dynamics. Thus, it is necessary to look into the early user commenting behavior after the publication of a news article to improve our ability to approximate the eventual comment volume of the article.

Since the difference among MAEs across the four feature spaces for a specific algorithm is not as clear as R^2, we use R^2 to discuss additional issues about the prediction performance in the subsequent sections. We use Random Forest in the remaining experiments.

Non-linearity

Contrasting the performances of the linear algorithm (Linear Regression) and other nonlinear algorithms (Random Forest, Support Vector Regression, and Neural Network), the R^2 of Linear Regression is consistently worse no matter which feature space (except for ART) is considered. This suggests that Linear Regression alone cannot solve the task of predicting the eventual comment number in a news article. We need to look into more complex models to improve accuracy.

5.5.4 Dominant Feature Discovery

We perform feature ablation by removing one set of features at a time to understand the strengths of the feature families described in Table 5.2. We report the outcome for both the global and local settings. Table 5.5 summarizes the outcome of this study. We observe that the decrease in R^2 is no more than 0.055 when we include only the comment features (compare the columns ALL and UC). The performance drops dramatically if we remove the comment features (see column $ALL - UC$). This is another supporting evidence on our account that signals gathered from early user

Table 5.5: R^2 results for feature ablation and selection for both global and local settings with machine learning algorithm Random Forest. Acronyms: WSP: Washington Post, DM: Daily Mail, WSJ: Wall Street Journal, FN: Fox News, Gd: the Guardian, NYT: New York Times.

Model	**ALL**	**UC**	***ALL − UC***	***rate***	***ALL − {rate}***
WSP	0.533	0.480	0.147	0.471	0.152
DM	0.541	0.498	0.170	0.477	0.193
WSJ	0.737	0.726	0.327	0.651	0.359
FN	0.449	0.408	0.142	0.378	0.134
Gd	0.468	0.428	0.168	0.416	0.170
NYT	0.631	0.612	0.213	0.484	0.280
Overall	0.560	0.520	0.185	0.470	0.209

comments largely influence the ability to predict comment volume.

We also study the importance of the individual features by applying the stepwise forward feature selection method [32]. This study shows that `rate` (from the group of comment features) is the most useful predictive variable in the prediction of comment volume. To further understand its importance, we redo the experiments with leaving out `rate`. The last column in Table 5.5 displays the outcome. Compared with the results in the column ALL, there is a dramatic drop in R^2, from 0.560 to 0.209, in the global setting. The decrease ranges between 0.298 - 0.381 across the outlets. This further illustrates the importance of `rate` in the prediction task at hand.

The outcome of feature ablation and selection is consistent with the results in Table 5.4. We also draw the same conclusion from the other algorithms: `rate` is the dominant single feature in the prediction task.

5.6 Rate Analysis

In this section, we focus on the prediction models trained only with `rate`, and investigate their characteristics across news outlets and news categories. We use Random Forests in the experiments reported in this section.

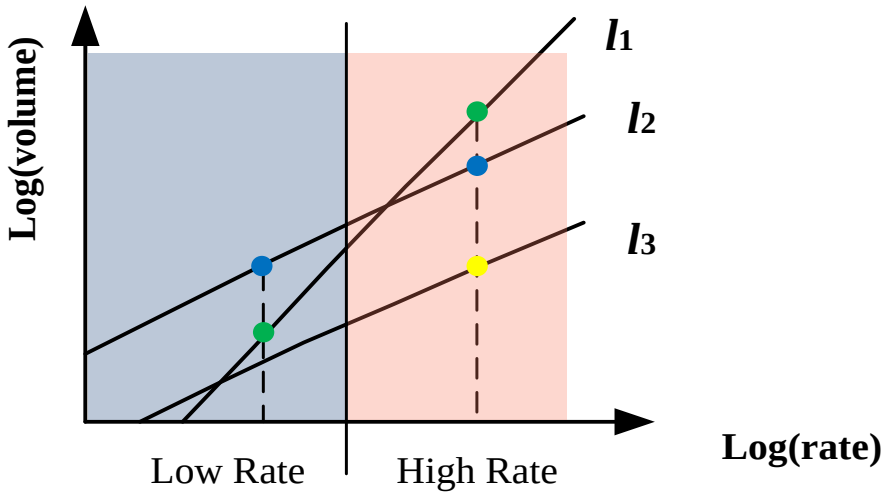
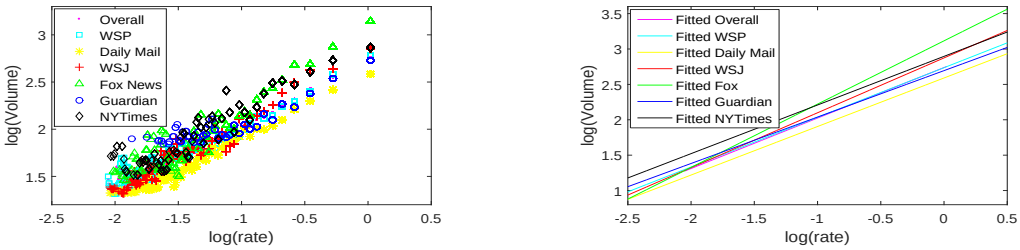


Figure 5.3: Comparison of regression lines.



(a) Rate and Volume in the log scale (b) Fitted Lines

Figure 5.4: The plot of the prediction model. If we plot the points based on the value of rate and logarithmic volume, points are too dense around the origin. Therefore, we draw the graph in the log scale for rate and volume.

5.6.1 Study of Rate across Outlets

We build rate models for both the global and local settings with Random Forest, considering the observed `rate`'s among the first $\alpha = 10$ user comments, and use them to predict the eventual comment volume of a news article. We also repeat the studies of rate models built on other values of α (the results are provided in Appendix), and the results are consistent with the 10 comment threshold.

Rate Modeling

Figure 5.4a shows the shape of the prediction models. Points in each dataset are fitted by a linear regression line, as shown in Figure 5.4b. We use the cartoon example in Figure 5.3 to describe the chief points we seek to convey in this study about the rate models. There are three regression lines in Figure 5.3. *l1* is the fitted regression line of points in outlet 1, *l2* is for outlet 2, and *l3* for outlet 3. We distinguish two interesting cases: (1) the regression lines cross each other, as in *l1* and *l2*; and (2) the regression lines are parallel, as in *l2* and *l3*.

Consider the lines *l1* and *l2*. The rate area can be split into two parts: low rate and high rate, based on their intersection. If we carefully compared the points on the two lines, we gather that the user commenting behaviors in outlets 1 and 2 vary across areas. In the low rate area, users in both outlets show less interest at the beginning, reflected by the small values of rate, but the users in outlet 2 keep commenting more than those in outlet 1 as indicated by the larger eventual comment volume. However, in the high rate area, articles in outlet 1 attract more commenting activity than those in outlet 2, even though the commenting activity early on is the same.

We can draw another useful observation by studying the lines *l1* and *l2*: the same rate fluctuation leads to different variation in comment volume. Since the slope of *l1* is larger than that of *l2*, *l1* will grow faster. Therefore, we conclude that the comment volume in outlet 1 is more sensitive to the rate in the early commenting stream than the comment volume in outlet 2.

Consider the parallel lines *l2* and *l3*. This scenario suggests that the same initial rate leads to different commenting volume in outlets 2 and 3. More precisely, the comment volume of a news article from outlet 2 is larger than that of a news article from outlet 3.

Table 5.6: Statistics of the prediction model trained by *rate*. The values in column Slope (Intercept) Interval are the lower and upper confidence limits for 95% confidence intervals of Slope (Intercept). Column MoPV = Mean of Predicted Volume in the log scale. We reuse the acronyms for news outlets in Table 5.5.

Model	**Slope**	**Intercept**	**Slope Interval**	**Intercept Interval**	**MoPV**
WSP	0.758	2.740	[0.755, 0.762]	[2.766, 2.768]	2.163
DM	0.703	2.606	[0.701, 0.705]	[2.604, 2.608]	2.131
WSJ	0.841	2.885	[0.832, 0.849]	[2.875, 2.895]	2.111
FN	**0.963**	**3.201**	[0.953, 0.972]	[3.194, 3.209]	**2.577**
Gd	0.656	2.728	[0.649, 0.663]	[2.723, 2.732]	2.396
NYT	0.707	2.935	[0.695, 0.719]	[2.924, 2.945]	2.454
Overall	0.777	2.767	[0.776, 0.778]	[2.766, 2.768]	2.224

Comparison across Outlets

We plot the shape of rate models in Figure 5.4 and provide the slopes and intercepts of regression lines in Table 5.6. We provide the lower and upper confidence limits for the 95% confidence intervals of slope and intercept of regression lines for each dataset. The bounds of slopes and intercepts show that their observed differences are due to actual differences between outlets rather than random chance. We also calculate the mean of the predicted logarithm of the comment volume for each dataset (column MoPV in Table 5.6). We present the results in the light of the discussion in the previous section. The plot shows that the users at Fox News are more active than those at the other news outlets— the logarithm of volume for Fox News is 2.577, which is close to the true value 2.47 in Table 5.1.

The regression line of Fox News (Figure 5.4b) behaves as line *l1* in Figure 5.3. Its high slope suggests that the total comment volume is very sensitive to the `rate` of the initial comments at this outlet. If the rate is small, the discussion dies out quickly. If the rate is large, the readers become very engaged. The comment volume of an article from the Guardian is the least sensitive to the `rate` of early user comments. Comparing the commenting activity at these outlets, the commenting activity at New York Times has the longest attention, because the commenting persists longer

Table 5.7: Article examples for outlets with the regression lines crossing each other (Fox News vs the Guardian) and in parallel (Daily Mail vs New York Times).

		Fox News	**the Guardian**
Pair 1	Article	FN_1	GD_1
	Rate	0.769	0.769
	N_A	2,768	705
Pair 2	Article	FN_2	GD_2
	Rate	0.092	0.092
	N_A	30	117
		Daily Mail	**New York Times**
Pair 3	Article	DM_1	NYT_1
	Rate	0.667	0.667
	N_A	227	407
Pair 4	Article	DM_2	NYT_2
	Rate	0.07	0.07
	N_A	50	108

and this results in higher volume. We think that the observed behavior might be caused by factors such as the quality or temporal relevance decay of articles, which are difficult to extract from the article content. New York Times behaves as *l2* and Daily Mail as *l3* in Figure 5.3. The commenting attention at Daily Mail seems to be lower. Interestingly, Wikipedia calls Daily News "middle-market tabloid," "that attempts to cater to readers who want some entertainment from their newspaper," so it might indicate a more fleeting character of their articles compared to other outlets.

Article Examples for Rate Models

We provide a few concrete examples to illustrate the rate-to-volume behavior across pair of news outlets in this section. We illustrate two scenarios: (i) pairs of news article for outlets whose regression lines cross each other (e.g., Fox News vs the Guardian) and (ii) pairs of news article for outlets whose regression lines are parallel (e.g., Daily Mail vs New York Times) in Table 5.7. For each of (i) and (ii) we showcase pairs of news articles from both the high and low rate zones.

Crossing regression lines. We have two pairs of news articles (FN_1, GD_1)[1,2] and (FN_2, GD_2)[3,4] from Fox News and the Guardian, respectively. The rates of (FN_1, GD_1) are both high at 0.769, while the rates of (FN_2, GD_2) are low at 0.092. According to our analysis of the rate-to-volume behavior, since the rates of FN_1 and GD_1 are located in the high rate area and the slope of the regression line for Fox News is larger than that of the Guardian, we expect N_A, the eventual number of comments, of FN_1 to be larger than that of GD_1: FN_1 receives more comments than GD_1, 2,768 versus 705 (in a week). See Pair 1 in Table 5.7. The effect is reversed for FN_2 and GD_2: FN_2 receives fewer comments than GD_2, 30 versus 117. See Pair 2 in Table 5.7.

Parallel regression lines. For the parallel case (Daily Mail vs New York Times), we also provide two pairs of news articles: (DM_1, NYT_1)[5,6] and (DM_2, NYT_2)[7,8]. The rates of (DM_1, NYT_1) are both high at 0.667, while the rates of (DM_2, NYT_2) are low at 0.07. Since the regression lines of these two outlets are parallel, we do not expect a reversal as in the previous case. Hence, we expect both the N_A of NYT_1 to be larger than that of DM_1 (407 versus 227, Pair 3 in the table), and the N_A of NYT_2 to be larger than that of DM_2 (108 versus 50, Pair 4 in the table).

5.6.2 Study of Rate across Categories

We now study the performance and characteristics of rate models by news category. We show that rate model behaves quite differently across the major news categories. The present results are from the rate models built on the first 10 user comments, but we have consistent observations from other values of α.

[1] www.foxnews.com/politics/2016/01/12/in-gop-response-haley-pans-obama-presidency-makes-case-for-new-direction.html
[2] theguardian.com/politics/2015/nov/26/labour-whip-email-vote-against-syria-airstrikes
[3] www.foxnews.com/opinion/2015/12/01/in-paris-obama-worships-at-altar-europes-real-religion-climate-change.html
[4] www.theguardian.com/science/2015/oct/28/us-approval-for-drug-that-turns-herpes-virus-against-cancer
[5] www.dailymail.co.uk/news/article-3296561/Syrian-anti-ISIS-activist-blogged-terrible-conditions-Raqqa-decapitated-Turkey-alongside-beheaded-corpse-frien
html
[6] www.nytimes.com/2015/11/12/us/politics/republicans-ted-cruz-marco-rubio.html
[7] www.dailymail.co.uk/sciencetech/article-3311075/Anomalies-thermal-scanning-Egypt-pyramids.html
[8] www.nytimes.com/2016/02/09/sports/basketball/knicks-fire-derek-fisher-as-coach.html

Categorizing Articles

To analyze `rate` in different news categories, we need to assign each article to its corresponding categories first. We observe that news outlets assign category labels to their articles. Our initial idea was to make use of these category labels. However, we soon noticed that labels are not consistent across news outlets. For example, the categories such as "U.S. Showbiz" in Daily Mail, "Local" and "National" in Washington Post, "Soccer" in the Guardian, and "Magazine" in New York Times are unique to these outlets. It is difficult to confidently align these categories over time in general, because news outlets periodically reorganize their news categories. We thus resort to the category labels in Google News, which are more stable over a longer period of time. We set the set of labels $C = \{$"Politics", "US", "World", "Sports", "Entertainment", "Technology", "Business", "Science", "Health"$\}$ in our analysis. We add "Politics" because it is a category that appears uniformly in all news outlets. We encounter many articles which are not explicitly assigned to any of these categories. We use their topics to determine their categories. We first categorize the topics of an article and then propagate the category labels to the article. We describe the process below.

Categorizing Topics. We follow the method of categorizing topics proposed in [141]. For a topic t, its probability of belonging to category $q \in C$ is

$$p(q|t) = \frac{p(q,t)}{p(t)} = \frac{|D_{q,t}|}{|D_t|} \propto |D_{q,t}|,$$

where D_t is the union of articles whose topic is t, $D_{q,t}$ denotes the subset of articles in D_t that are labeled with category q. For a specific topic t, D_t is constant. We select the top three categories for t if it belongs to over three ones. For example, the topic "Donald Trump" is assigned to the categories "US," "World," and "Politics."

Assigning News Categories to News Articles. Our crawler extracts topic information for each news article. Once we determine the news categories C_t of a

Table 5.8: Statistics of the datasets and the global models trained by *rate* in category domains. The values in column Slope (Intercept) shows the slope (intercept) of the *rate* model together with its lower and upper confidence limits for 95% confidence intervals.

Category	**# of Articles**	R^2	**Slope** (Interval)	**Intercept** (Interval)
Politics	8,491	0.442	0.729 ([0.727, 0.730])	2.818 ([2.816, 2.819])
US	6,202	0.457	0.667 ([0.665, 0.668])	2.719 ([2.717, 2.720])
World	2,548	0.490	0.700 ([0.696, 0.704])	2.735 ([2.732, 2.739])
Sports	1,839	0.484	0.570 ([0.566, 0.575])	2.448 ([2.444, 2.453])
Entertainment	1,828	0.535	0.645 ([0.640, 0.650])	2.549 ([2.545, 2.554])
Technology	469	0.528	0.633 ([0.620, 0.647])	2.567 ([2.553, 2.581])
Business	342	0.472	0.664 ([0.649, 0.679])	2.639 ([2.623, 2.654])
Science	235	0.468	0.652 ([0.626, 0.678])	2.626 ([2.599, 2.653])
Health	156	0.493	0.562 ([0.525, 0.600])	2.485 ([2.440, 2.530])

topic t, we place each article on topic t in each of the categories in C_t.

Comparison across Categories

We first partition the overall dataset across the news categories C, and then train a rate model in each category. The analysis follows the steps of the analysis across news outlets. Table 5.8 gives the article count, performance (R^2), and the characteristics of the rate models per news category.

The first observation is that R^2 differs very little across the news categories, hovering about 0.5. This is in stark contrast to the observations made about R^2 in news outlets. Comparing the linear regression in each category, "Politics" has the highest slope, indicating that the predicted comment volume for articles in this category is more sensitive to the early rate than for articles in other news categories. "Health"

Table 5.9: The distribution of article count by outlet and category. We reuse the acronyms for news outlets in Table 5.5.

	WSP	**DM**	**WSJ**	**FN**	**Gd**	**NYT**
Politics	3,374	2,077	1,212	940	408	480
US	2,257	1,548	966	283	655	493
World	635	878	362	188	351	134
Sports	403	1,060	112	62	154	102
Entertainment	199	1,242	53	80	215	39
Technology	80	195	88	42	48	16
Business	80	105	89	8	41	19
Science	33	100	33	31	29	9
Health	33	72	18	25	4	4

has the lowest comment volume and the prediction for its articles is less sensitive to the rate of early comments.

5.6.3 Interplay between Outlets and Categories

Given the different performance of the rate models across outlets and news categories, a natural follow up question is whether there is some mutual effect between outlets and categories over rate. For this study, we summarize the distribution of article sizes among outlets and categories in Table 5.9. Then, we repeat the prediction analysis for rate models. Table 5.10 displays the results per outlet and news categories.

Comparing Table 5.10 (local rate models) to Table 5.8 (global rate models), we observe that the local models at Washington Post achieve similar R^2 as the global one in "Politics" and "US." For "Sports," local models at Daily Mail perform similarly to the global one in Table 5.8. This is because either Washington Post or Daily Mail dominates the article size in these categories (see Table 5.9).

We notice that commenting activity in the "Politics" articles at Fox News is quite distinct from others: by far the largest slope and intercept. In other outlets, "Politics", "US," and "World" are comparable. This indicates that the total comment volume of an article from "Politics" is very sensitive to the `rate` of the initial com-

ments at Fox News.

User commenting preference per category within each outlet reveals additional properties about news outlets (Table 5.9). In Table 5.10, we notice that the value of intercept in "Politics" is the largest across all categories and outlets, except for Daily Mail where "World" has the largest intercept. Besides, the highest slope appears three times in "Politics", twice in "World", and once in "US". Considering that most of the "US" and "World" articles are related to politics, it is reasonable to conclude that the comment volume is *sensitive to rate* (suggested by high slope) and *higher* (reflected by large intercept) in political area at most outlets.

5.7 Conclusion

In this paper, we study the problem of predicting the total number of user comments a news article will receive. We compile and analyze a large set of features, which we group by topic, article, user comment, news factor, and miscellaneous. Our main insight is that the early dynamics of user comments contribute the most to an accurate prediction, while news article specific factors have surprisingly little influence. Furthermore, we show that the early arrival rate of comments is the best indicator of the eventual number of comments. We conduct an in-depth analysis of this feature across several dimensions, such as news outlets and news article categories. We show that the prediction of comment volume is very sensitive to the early user commenting activity in some news outlets (e.g., Fox News) and categories (e.g., Politics).

We believe that our findings shed new light on the unique characteristics of readership community compared to other online communities, e.g., those at Twitter or Facebook. This is particularly emphasized by the strong role of early user posting activity on the eventual comment volume for a news article. This has important implications in social media and user behavioral response process understanding, which are key components of the social-news media ecosystem. We believe that this insight

Table 5.10: The results of the local models trained by *rate* among outlets and categories. The three values in each cell are R^2, then slope, and intercept. Some categories are removed because of insufficient articles.

	WSP	**DM**	**WSJ**	**FN**	**Gd**	**NYT**
Politics	0.450	0.366	**0.657**	0.352	0.396	0.400
	0.701	0.671	0.783	**0.891**	0.657	0.647
	2.772	2.675	2.934	**3.202**	2.775	2.953
US	0.438	0.393	**0.628**	0.324	0.432	0.372
	0.652	0.652	0.772	0.690	0.744	0.628
	2.692	2.612	2.921	2.932	2.764	2.925
World	0.490	0.361	**0.587**	0.236	0.458	0.380
	0.714	0.707	0.706	0.663	0.634	0.640
	2.765	2.744	2.722	2.772	2.714	2.781
Sports	0.389	0.489	**0.565**		0.337	
	0.566	0.588	0.694	-	0.433	-
	2.478	2.426	2.722		2.481	

is also of value to news analytics, which may lead to better understanding of user participation motives and engagement in commenting news items online.

CHAPTER 6

FUTURE WORK

6.1 End-to-End Transfer Learning with Neural Network

In Chapter 2, we propose an unsupervised heterogeneous domain adaptation framework that could deal with features from various fields, including visual features, textual features and deep features. This framework has great potential to be upgrade to an end-to-end system.

In detail, the current approach adapts features which is the output of other feature generation models. The trend of deep learning in almost all field in computer sciences encourage a more complete framework that combine feature generation and feature transfer learning together as an end-to-end system. There are several advantages: Firstly, feature learning in different domains is able to influence the feature generation by propagation of neural network. The labels in one domain thus help the feature learning of another domain. Secondly, it would be more convenient to combine the framework of transfer learning with neural network. Neural network consists of multiple layers, we may design a few transformation layers that handle domain adaptation in different layers during feature generation. It is much better than the current model that only utilize features as input.

6.2 Linear and Non-linear Neural Network for Knowledge Graph Learning

In Chapter 5, we present several observations that point out the current issues of knowledge graph learning methods. We show that the current knowledge graph learning models based on CNN are local and shallow, and are essentially local weighted sum of entities and relation embedding. There should be more interactions between dimensions in embedding space while CNN only model linear ones. It would be promising if we could research on how to add non-linear interaction among dimensions of entities and relation.

Tensor factorization models on knowledge graph learning could use non-linear tensor multiplication operation to score triplets. The tensor multiplication is expressive, but only a small portion of the products are meaningful. Tensor factorization models rely on strong regularization to overcome over-fitting concern. A combination of linear and nonlinear models is an promising direction for future work. It would be able to model both weighted sum (linear operation) and multiplication (non-linear operation) among embedding space.

In summary, neural network is not a black box that could be applied directly for many task in various fields. Deep features learned by neural network are not always better than human-designed features. Properties and analysis of the data are most important when we design and utilize neural network. In this thesis, we design and learn features according to the properties of the data, and present a few applications of neural network on various tasks. We aims to solve the issues as the development of both neural network and tasks. There are always new issues to solve and new targets to achieve, we wish to develop more and better models and frameworks that fit the data in future.

BIBLIOGRAPHY

[1] Ahmet Aker, Emina Kurtic, AR Balamurali, Monica Paramita, Emma Barker, Mark Hepple, and Rob Gaizauskas. A graph-based approach to topic clustering for online comments to news. In *ECIR*, 2016.

[2] David L Altheide and Christopher J Schneider. *Qualitative media analysis*, volume 38. Sage, 2012.

[3] Pablo Aragón, Vicenç Gómez, David García, and Andreas Kaltenbrunner. Generative models of online discussion threads: state of the art and research challenges. *Journal of Internet Services and Applications*, 8(1):15, 2017.

[4] Yoav Artzi, Patrick Pantel, and Michael Gamon. Predicting responses to microblog posts. In *NAACL*, pages 602–606. ACL, 2012.

[5] A. Asuncion and D. Newman. UCI machine learning repository. `http://archive.ics.uci.edu/`, 2007.

[6] Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *WSDM*, pages 13–22, 2013.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Ali Balali, Masoud Asadpour, and Hesham Faili. A supervised method to predict the popularity of news articles. *Computación y Sistemas*, 21(4):703–716, 2017.

[9] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565. Springer, 2019.

[10] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

[11] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *J. on Imaging Sciences*, 2(1):183–202, 2009.

[12] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, 2013.

[14] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning (FTML)*, 3(1):1–122, 2011.

[16] Y. Cao, M. Long, and J. Wang. Unsupervised domain adaptation with distribution matching machines. In *AAAI*, 2018.

[17] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *KDD*, pages 714–722, 2012.

[18] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *WWW*, 2014.

[19] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *RecSys*, pages 293–296. ACM, 2010.

[21] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] Robin Devooght and Hugues Bersini. Collaborative filtering with recurrent neural networks. *arXiv preprint arXiv:1608.07400*, 2016.

[23] Nicholas Diakopoulos and Mor Naaman. Topicality, time, and sentiment in online news comments. In *CHI*, 2011.

[24] Julio Cesar Soares Dos Rieis, Fabrício Benevenuto de Souza, Pedro Olmo S Vaz de Melo, Raquel Oliveira Prates, Haewoon Kwak, and Jisun An. Breaking the news: First impressions matter on online news. In *ICWSM*, 2015.

[25] L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. In *ICML*, 2012.

[26] Martin Emmer, Gerhard Vowe, and Jens Wolling. *Bürger online: die Entwicklung der politischen Online-Kommunikation in Deutschland*. UVK Verlagsgesellschaft, 2011.

[27] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013.

[28] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[29] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Sequence and time aware neighborhood for session-based recommendations: Stan. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1069–1072, 2019.

[30] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

[31] Biving Group. The use of the internet by america's largest newspapers. `http://www.bivings.com/thelab/presentations/2008study.pdf`, 2008.

[32] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *JMLR*, 3(Mar):1157–1182, 2003.

[33] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[34] D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computat.*, 16(12): 2639–2664, 2004.

[35] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko. Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*, 2013.

[38] M. Hong, Z. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *J. on Optimizat.*, 26(1):337–364, 2016.

[39] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 306–310, 2017.

[40] Yantao Jia, Yuanzhuo Wang, Hailun Lin, Xiaolong Jin, and Xueqi Cheng. Locally adaptive translation for knowledge graph embedding. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[41] How Jing and Alexander J Smola. Neural survival recommender. In *WSDM*, pages 515–524. ACM, 2017.

[42] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[43] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] Spiro Kiousis. Interactivity: a concept explication. *New media & society*, 4(3): 355–383, 2002.

[46] Ryota Kobayashi and Renaud Lambiotte. Tideh: Time-dependent hawkes process for predicting retweet dynamics. In *ICWSM*, 2016.

[47] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[48] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

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

[50] M. Kowalski, M. Szafranski, and L. Ralaivola. Multiple indefinite kernel learning with mixed norm regularization. In *ICML*, 2009.

[51] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, 2011.

[52] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *ACM SIGIR*, pages 95–104. ACM, 2018.

[53] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.

[54] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428, 2017.

[55] Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach. In *IJCAI*, pages 3805–3811, 2018.

[56] Dongliang Liao, Jin Xu, Gongfu Li, Weijie Huang, Weiqing Liu, and Jing Li. Popularity prediction on online articles with deep fusion of temporal process and content features. In *AAAI*, 2019.

[57] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[58] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[59] Zehang Lin, Feitao Huang, Yukun Li, Zhenguo Yang, and Wenyin Liu. A layer-wise deep stacking model for social image popularity prediction. *WWW*, 22(4): 1639–1655, 2019.

[60] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1):76–80, 2003.

[61] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. Stamp: short-term attention/memory priority model for session-based recommendation. In *SIGKDD*, pages 1831–1839, 2018.

[62] M. Long, J. Wang, G. Ding, J. Sun, and P. S Yu. Transfer joint matching for unsupervised domain adaptation. In *CVPR*, 2014.

[63] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4-5):331–390, 2018.

[64] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[65] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11):39–41, 1995.

[66] Gilad Mishne, Natalie Glance, et al. Leave a reply: An analysis of weblog comments. In *Weblogging at WWW*, 2006.

[67] Swapnil Mishra. Bridging models for popularity prediction on social media. In *WSDM*, pages 810–811, 2019.

[68] Swapnil Mishra, Marian-Andrei Rizoiu, and Lexing Xie. Feature driven and point process approaches for popularity prediction. In *CIKM*, pages 1069–1078, 2016.

[69] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

[70] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[71] Gabriel de Souza P Moreira, Felipe Ferreira, and Adilson Marques da Cunha. News session-based recommendations using deep neural networks. *arXiv preprint arXiv:1808.00076*, 2018.

[72] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.

[73] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A capsule network-based embedding model for knowledge graph completion and search personalization. *arXiv preprint arXiv:1808.04122*, 2018.

[74] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*, 2016.

[75] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, 2011.

[76] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[77] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth Conference on Artificial Intelligence*, 2016.

[78] L. Niu, W. Li, and D. Xu. Multi-view domain generalization for visual recognition. In *ICCV*, 2015.

[79] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *SIGKDD*, pages 1933–1942, 2017.

[80] Alexandra Olteanu, Carlos Castillo, Nicholas Diakopoulos, and Karl Aberer. Comparing events coverage in online news and social media: The case of climate change. In *ICWSM*, pages 288–297, 2015.

[81] J. C Platt, K. Toutanova, and W. Yih. Translingual document representations from discriminative projections. In *EMNLP*, 2010.

[82] Fabian Prochazka, Patrick Weber, and Wolfgang Schweiger. Effects of civility and reasoning in user comments on perceived journalistic quality. *Journalism Studies*, 19(1):62–78, 2018.

[83] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007.

[84] N. Rasiwasia, J. C. Pereira, E. Coviello, G. Doyle, G. RG Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM MM*, 2010.

[85] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI*, volume 33, pages 4806–4813, 2019.

[86] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461, 2009.

[87] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

[88] Marian-Andrei Rizoiu, Swapnil Mishra, Quyu Kong, Mark Carman, and Lexing Xie. Sir-hawkes: Linking epidemic models and hawkes processes to model diffusions in finite populations. In *WWW*, pages 419–428, 2018.

[89] Georgios Rizos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Predicting news popularity by mining online discussions. In *WWW*, pages 737–742, 2016.

[90] Matthew Rowe and Harith Alani. Mining and comparing engagement dynamics across multiple social media platforms. In *WebSci*, pages 229–238, 2014.

[91] Carlos Ruiz, David Domingo, Josep Lluís Micó, Javier Díaz-Noci, Koldo Meso, and Pere Masip. Public sphere 2.0? the democratic qualities of citizen debates in online newspapers. *The International Journal of Press/Politics*, 16(4):463–487, 2011.

[92] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. *WWW*, 1:285–295, 2001.

[93] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[94] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *JMLR*, 6(Sep):1265–1295, 2005.

[95] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Association for the Advancement of Artificial Intelligence*, 2017.

[96] Erez Shmueli, Amit Kagian, Yehuda Koren, and Ronny Lempel. Care to comment?: recommendations for commenting on news stories. In *WWW*, pages 429–438, 2012.

[97] Stefan Siersdorfer, Sergiu Chelaru, Jose San Pedro, Ismail Sengor Altingovde, and Wolfgang Nejdl. Analyzing and mining comments and comment ratings on the social web. *TWEB*, 8(3):17, 2014.

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

[99] Jane B Singer. Separate spaces: Discourse about the 2007 scottish elections on a national newspaper web site. *The International Journal of Press/Politics*, 14 (4):477–496, 2009.

[100] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. 2013.

[101] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *WSDM*, pages 555–563. ACM, 2019.

[102] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[103] S. Sukhija, N. C Krishnan, and G. Singh. Supervised heterogeneous domain adaptation via random forests. In *IJCAI*, 2016.

[104] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. *arXiv preprint arXiv:1511.05547*, 2015.

[105] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[106] Ilya Sutskever, Joshua B. Tenenbaum, and Ruslan R Salakhutdinov. Modelling relational data using bayesian clustered tensor factorization. In *Advances in Neural Information Processing Systems*, 2009.

[107] Chenhao Tan, Adrien Friggeri, and Lada Adamic. Lost in propagation? unfolding news cycles from the source. In *ICWSM*, 2016.

[108] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22. ACM, 2016.

[109] Alexandru Tatar, Jérémie Leguay, Panayotis Antoniadis, Arnaud Limbourg, Marcelo Dias de Amorim, and Serge Fdida. Predicting the popularity of online articles based on user comments. In *WIMS*, 2011.

[110] Yla R Tausczik and James W Pennebaker. The psychological meaning of words: LIWC and computerized text analysis methods. *JLS*, 29(1):24–54, 2010.

[111] Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. Predicting the volume of comments on online news stories. In *CIKM*, pages 1765–1768, 2009.

[112] Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. News comments: Exploring, modeling, and online prediction. In *ECIR*, pages 191–203, 2010.

[113] T.H. Tsai, Y. Yeh, and Y. Wang. Learning cross-domain landmarks for heterogeneous domain adaptation. In *CVPR*, 2016.

[114] Oren Tsur and Ari Rappoport. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *WSDM*, pages 643–652, 2012.

[115] 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*, pages 5998–6008, 2017.

[116] B. Vatashsky and K. Crammer. Multi class learning with individual sparsity. In *IJCAI*, 2013.

[117] Chunyan Wang, Mao Ye, and Bernardo A Huberman. From user comments to on-line conversations. In *SIGKDD*, pages 244–252, 2012.

[118] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*, pages 1835–1844, 2018.

[119] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. A collaborative session-based recommendation approach with parallel memory modules. In *ACM SIGIR*, 2019.

[120] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. Learning hierarchical representation model for nextbasket recommendation. In *SIGIR*, pages 403–412, 2015.

[121] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Association for the Advancement of Artificial Intelligence*, 2014.

[122] Patrick Weber. Discussions in the comments section: Factors influencing participation and interactivity in online newspapers' reader comments. *New Media & Society*, 16(6):941–957, 2014.

[123] P. Wei, Y. Ke, and C.K. Goh. Deep nonlinear feature coding for unsupervised domain adaptation. In *IJCAI*, 2016.

[124] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Predicting successful memes using network and community structure. In *ICWSM*, 2014.

[125] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[126] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *WSDM*, pages 495–503. ACM, 2017.

[127] Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. Neural news recommendation with topic-aware news representation. In *ACL*, pages 1154–1159, 2019.

[128] 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*, volume 33, pages 346–353, 2019.

[129] X. Wu, H. Wang, C. Liu, and Y. Jia. Cross-view action recognition over heterogeneous feature spaces. In *ICCV*, 2013.

[130] Han Xiao, Minlie Huang, and Xiaoyan Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[131] M. Xiao and Y. Guo. A novel two-step method for cross language representation learning. In *NIPS*, 2013.

[132] M. Xiao and Y. Guo. Feature space independent semi-supervised domain adaptation via kernel matching. *PAMI*, 37(1):54–66, 2015.

[133] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *Proc. 28th Int. Joint Conf. Artif. Intell.(IJCAI)*, pages 3940–3946, 2019.

[134] Y. Yan, W. Li, M. KP Ng, M. Tan, H. Wu, H. Min, and Q. Wu. Learning discriminative correlation subspace for heterogeneous domain adaptation. In *IJCAI*, 2017.

[135] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[136] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489, 2016.

[137] Tae Yano and Noah A Smith. What's worthy of comment? content and comment volume in political blogs. In *ICWSM*, 2010.

[138] T. Yao, Y. Pan, C. Ngo, H. Li, and T. Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, 2015.

[139] Y. Yeh, C. Huang, and Y.F. Wang. Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *TIP*, 23(5):2009–2018, 2014.

[140] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *KDD*, pages 1513–1522, 2015.

[141] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *ECIR*. Springer, 2011.

[142] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *WWW*, pages 167–176, 2018.

[143] J.T. Zhou, I. W Tsang, S.J. Pan, and M. Tan. Heterogeneous domain adaptation for multiple classes. In *AISTATS*, 2014.

[144] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, pages 3602–3608, 2017.

[145] Marc Ziegele and Oliver Quiring. Conceptualizing online discussion value: A multidimensional framework for analyzing user comments on mass-media websites. *Annals of the International Communication Association*, 37(1):125–153, 2013.

[146] Marc Ziegele, Timo Breiner, and Oliver Quiring. What creates interactivity in online news discussions? an exploratory analysis of discussion factors in user comments on news items. *Journal of Communication*, 64(6):1111–1138, 2014.

[147] Marc Ziegele, Mathias Weber, Oliver Quiring, and Timo Breiner. The dynamics of online news discussions: effects of news articles and reader comments on users' involvement, willingness to participate, and the civility of their contributions. *Information, Communication & Society*, 21(10):1419–1435, 2018.

[148] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 580–588, 2001.

[149] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. In *Artificial Neural Networks*, pages 14–22. Springer, 2008.