博士論文

Doctor Thesis (Academic Year 2015)

# Category-enhanced Embedding Model for Massive Text Data

（カテゴリを用いた大規模テキストデータのための

分散表現獲得手法に関する研究）

The University of Tokyo

School of Engineering
Technology Management for Innovation

Junki Marui
丸井淳己

Supervisor: Prof. Ichiro Sakata

February 2016

**Abstract Of Doctor Thesis  Academic Year 2015**

# Category-enhanced Embedding Model for Massive Text Data

## Summary

This thesis addresses the effectiveness of using the category-enhanced embedding method for massive text data to understand entities.

In the era of Big data, many real-world entities are mapped to the Web or to databases, in the form of text, which creates a need to process them to extract useful knowledge. Most of the entities do not have structured data on the Web; therefore, we often need to process massive and various text data with the weakly structured labels, which are often represented as categories. Categories that we discuss here are not necessarily well structured, but they are groups which humans use for understanding entities and where entities share underlying properties. For example, e-commerce sites have categories for products and academic articles have keywords; they are labeled by humans for searchability. We call such categories "explicit categories". Categories in this thesis also include communities or user groups in social media; most of social media do not have explicit communities or user groups, however we can divide them into similar user groups using community detection algorithm. We call such categories "implicit categories" since we do not have explicit tags or labels for each category, but we know the algorithm how to extract groups where entities share underlying properties. Using these categories in addition to entity information, the proposed method can learn similarity among entities from categories and vice versa, and it can be used for analyzing the landscape of entities.

In this thesis, we propose an embedding method that learns similarity among categories and entities. Embedding methods *embed* an entity into a fixed dimensional vector, and recent studies in natural language processing (NLP) concentrate on this approach. One of the reasons this approach is popular is that the embeddings of infrequent words can be learned from frequent words. We extend this idea to categories as well as entities since some categories or entities appear infrequently compared to others. This is especially true with massive data, which often have various categories and entities, and thus traditional methods have difficulties to apply.

We propose the Category Vector models, which train embeddings of the categories, entities, and words simultaneously, and we compare the proposing models with the traditional model and the existing embedding models which do not take categories into account. We conduct category inference tasks on categories (explicit categories) of e-commerce sites in Japan and in the U.S., and the same inference tasks on communities (implicit categories) of a microblogging service. We also analyze the ways in which the Category Vector models capture the difference among categories, and we compare the word embeddings that capture the local context of words (i.e. word usage) and the learned embeddings of categories.

The proposing method can be widely applied for various massive data since the method has less limitations than top-down methods using taxonomy or ontology, which are often unavailable for various entities. Moreover, the proposing method can make use of humans' knowledge for entities compared to bottom-up approach such as bag-of-words; the proposing method can capture how similar categories or entities are from the massive dataset. The proposing method is scalable, therefore we can apply for massive data in the industries. We expect that the Category Vector models assist with decision making in industries by capturing the landscape of entities and categories.

## Keywords

Entity labeling, distributed representations, community analysis, neural network, natural language processing

School of Engineering
Technology Management for Innovation
The University of Tokyo

Junki Marui

# Contents

# List of Figures

# List of Tables

# Chapter 1

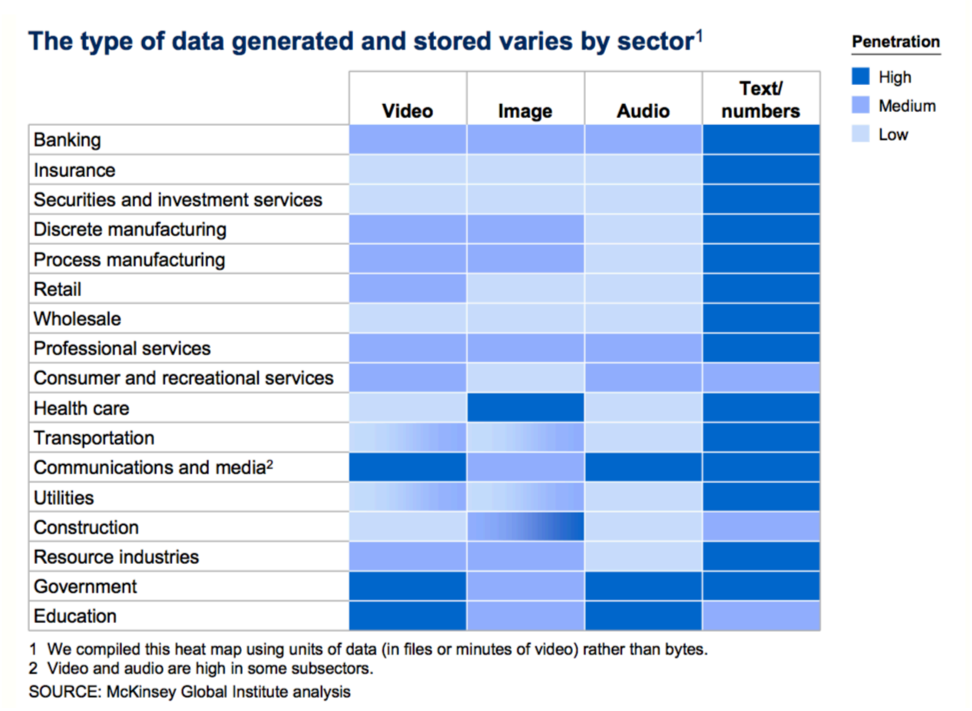
# Introduction

## 1.1 Background

Recent years have seen an exponential growth of massive data, which is often referred to as "Big data". In this chapter, first we describe what the term "Big data" refers to, and discuss the situations in which it appears. Second, we discuss "categories", which are often analysis units for humans and also are used in Big data analysis. Third, we explain the "embedding" methods, which are employed in natural language processing (NLP) for the scalability and applicability to massive data. Finally, we describe the purpose of this thesis: why we apply the embedding methods to handle categories in Big data.

### 1.1.1 Big data

The need to process massive amounts of data in order to explore certain tendencies and to extract knowledge is growing rapidly as the amount of data explodes. Many people refer to such large amounts of data as "Big data". McKinsey Global Institute reported that various domains can use Big data to create value through *(1)creating transparency, (2)enabling experimentation to discover needs, expose variability, and improve performance, (3) segmenting populations to customize actions, (4)replacing/supporting human decision making with automated algorithms, and (5)innovating new business models, products, and services* [59]. Among different types of data, text/numerical data are stored in various sectors, while video, images, and audio information is stored in specific domains. The report also lists techniques for analyzing Big data, including data mining, natural language processing, and network analysis, which can be applied to marketing applications such as branding campaigns, and modeling the purchase behaviors of customers. Japanese Ministry of Internal Affairs and Communications reported in the White



**The type of data generated and stored varies by sector[1]**

| | Video | Image | Audio | Text/ numbers |
|---|---|---|---|---|
| Banking | ▨ | ▨ | ▨ | ■ |
| Insurance | □ | □ | □ | ■ |
| Securities and investment services | □ | □ | □ | ■ |
| Discrete manufacturing | ▨ | ▨ | □ | ■ |
| Process manufacturing | ▨ | ▨ | □ | ■ |
| Retail | ▨ | □ | □ | ■ |
| Wholesale | □ | □ | □ | ■ |
| Professional services | ▨ | ▨ | ▨ | ■ |
| Consumer and recreational services | ▨ | □ | ▨ | ▨ |
| Health care | □ | ■ | □ | ■ |
| Transportation | □▨ | □▨ | □ | ■ |
| Communications and media[2] | ■ | ▨ | ■ | ■ |
| Utilities | □▨ | □▨ | □ | ■ |
| Construction | □ | ▨■ | □ | ▨ |
| Resource industries | ▨ | ▨ | □ | ■ |
| Government | ■ | ▨ | ■ | ■ |
| Education | ■ | ▨ | ■ | ▨ |

**Penetration**

| | |
|---|---|
| ■ | High |
| ▨ | Medium |
| □ | Low |

1 We compiled this heat map using units of data (in files or minutes of video) rather than bytes.
2 Video and audio are high in some subsectors.
SOURCE: McKinsey Global Institute analysis

Figure 1.1: The type of data generated and stored varies by sector, cited from the Big data report of McKinsey Global Institute [59]

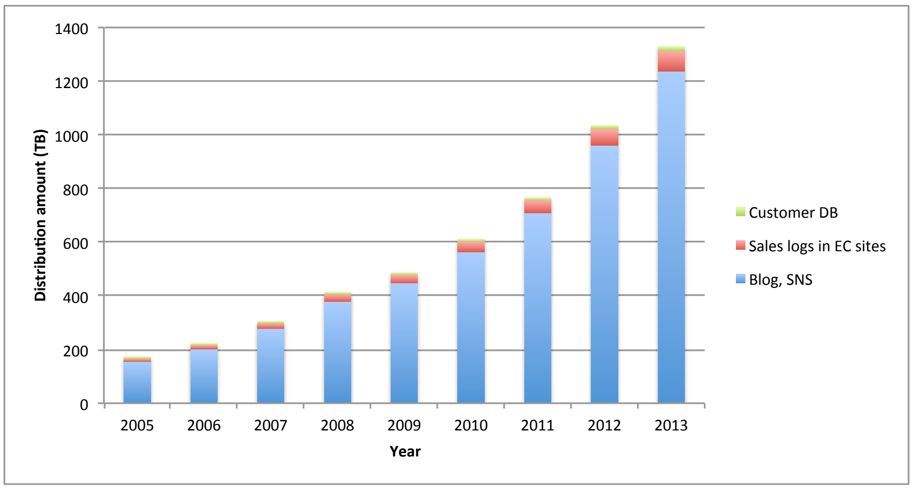


Figure 1.2: Development of Big data in the field of Blog, SNS, and EC sites. Data are retrieved from the White Paper, "Information and Communications in Japan".

Table 1.1: Top 4 usage of media by industry. Data are retrieved from the White Paper, “Information and Communications in Japan”.

| **Industry** | **Top 1** | | **Top 2** | | **Top 3** | | **Top 4** | |
|---|---|---|---|---|---|---|---|---|
| Commerce | POS data | 31.8% | Sales logs | 27.1% | Access logs | 22.1% | Blog, SNS | 21.3% |
| Finance/ Insurance | Access logs | 19.4% | Customer DB | 18.7% | Blog, SNS | 16.6% | Sales logs | 16.0% |
| Manufacture | Sales logs | 23.0% | Blog, SNS | 19.1% | Access logs | 18.6% | Customer DB | 18.0% |
| Energy/ Water supply | CTI Voice logs | 21.0% | Access logs | 20.6% | Sales logs | 18.5% | Blog, SNS | 18.0% |
| Real estate | Access logs | 23.9% | GPS data | 20.0% | Blog, SNS | 18.6% | POS data | 18.4% |
| Service | Blog, SNS | 19.6% | POS data | 19.4% | Access logs | 18.4% | Customer DB | 17.7% |
| Information/ Communication | Sales logs | 24.1% | Access logs | 18.4% | POS data | 18.2% | Customer DB | 17.8% |
| Construction | Access logs | 19.5% | Customer DB | 14.7% | Business Diaries | 14.7% | Blog, SNS | 12.6% |
| Transportation | Sales logs | 25.7% | Customer DB | 16.6% | Financial data | 15.3% | Access logs | 14.4% |

Paper, “Information and Communications in Japan” (2014) [平成 26 年版 情報通信白書][1] that the amount of the data has increased by 8.7 times during the 8 years (from 2005 to 2013). As we can see in Figure 1.2, data of articles in blogs and social networking services (SNS), sales logs in e-commerce (EC) sites, and customer databases (DB) are also growing; the amount has increased by 7.7 times during the same period. According to the White Paper, the amount of these data is less than 0.01% of the total distribution amount; however, they are used in various industries. Table 1.1 shows that many companies use sales logs, customer DB, and blog/SNS, regardless of the industry. These media tools are useful for marketing and advertisement purposes since they have various attributes for the customers and users, activities of users, and massive texts generated by users — namely, information regarding *various entities* in the real world. Here, real-world entities mean objects or users in services, for example, products or customers in a Web service. The information of such entities is electronically stored in databases. A large number of companies try to extract useful information for their business from massive information of such entities, which is one of the motivations of utilizing Big data.

Although no clear definition is established for the term Big data, we intend to mean not only the massive data, but the necessity for scalable algorithms and

[1]Retrieved from `http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h26/pdf/n3100000.pdf` on August 12th 2015

data management to extract knowledge. Boyd and Crawford defined Big data as *a cultural, technological, and scholarly phenomenon that rests on the interplay of technology, analysis and mythology* [9]. They claimed that Big data cannot be defined from the amount of data, but that it is the phenomenon that can change people's view of knowledge. Mayer-Schönberger and Cukier claimed that it is only the start to define Big data as massive data which cannot fit into the memory — Big data challenge the way we perceive the world, and make the society *shed some of its obsession for causality in exchange for simple correlations: not knowing why but only what* [61]. Before Big data existed, we tested a small number of hypotheses; however, using the Big data, we can discover the connections between things we have not considered. In addition, as Mayer-Schönberger and Cukier also pointed out, *Big data is messy, varies in quality*; therefore, *we lose in accuracy at the micro level we gain in insight at the macro level.* Thus, modern Big data approaches are often referred to as the way in which we *let data speak for itself*, not so much seeking to assume hypotheses, but aiming to improve the model from the larger set of data.

To know more about various real entities in the Web or the databases, therefore, the ability to handle massive data, and to extract their tendency, and establish their relationship is becoming increasingly important. Among massive data, our thesis focuses on the massive text data since various types of information from different industries is stored in a variety of ways in the text data. To process text data, such as documents, we generally use vector representations for the text data. In recent years, many researchers in the field of natural language processing (NLP) focus on the embedding method, which can improve the model from massive text data in an unsupervised way. Thus, it is considered to be a promising method for the era of Big data.

### 1.1.2 Development of embedding methods

It is common to handle texts, or *documents*, in the form of vector representations because it is easy to apply mathematical techniques to documents when we have document vectors. Traditionally, it has been common to assign a dimension for each word or sequence of words. The use of $n$-grams and bag-of-words (BOW), which are widely employed for NLP, requires such a strategy to deal with text data. An $n$-gram is a sequence of $n$ words, appearing in a text. If $n = 1, 2, 3$ (called *unigram*,*bigram*, and *trigram*, respectively), we can retrieve sequences shown in the following from a sentence.

- each zebra has its own unique pattern
- Unigram: {each, zebra, has, its, own, unique, pattern}

- Bigram: {each zebra, zebra has, has its, its own, own unique, unique pattern}
- Trigram: {each zebra has, zebra has its, has its own, its own unique, own unique pattern}

Language models use $n$-grams to predict the next word from a given sequence of words. $N$-gram model predicts better when $n$ is larger; however, $n$-grams become much more sparse with larger $n$. The main reason of the sparsity is Zipf's law. Zipf's law states that the frequency of the $k$-th frequent word is proportional to $1/k$ [105]. For example, the word *okapi* is much less frequent than *zebra*. Therefore, it is hard to estimate the probability of "each *okapi* has its own unique pattern", although it sounds natural. The combinatorial explosion caused from this property is also a problem. When we observe more data, we face new, infrequent words. We cannot handle all possible combinations; therefore, we have to assign probabilities to unseen sequences (which is called smoothing). We can use $(n-1)$-grams to estimate the probability of $n$-grams. This idea leads to back-off smoothing [46], and Kneser-Ney smoothing [50] , and yet $n$-grams with $n > 3$ are difficult to handle.

One of the promising approaches to avoid the sparsity of $n$-grams is the "embedding method". The embedding method projects a symbol to vectors or matrices. Most of the embedding methods in NLP assign real-valued vectors to words since words are key symbols.

Embeddings, which we will discuss in chapter 2 in details, are typically vector representations. The embedding method often assigns a real-valued vector representation for each word, of which dimension is 100 to 1000, and it projects some aspects of the *meanings* of words into vector representations. When we use the embedding method, we can handle words or several words as vectors, and we can calculate the similarities or dissimilarities from their vector representations. Therefore, we can avoid the combinatorial explosion without using external knowledge.

Also, the embedding method can ease the harmful influence of Zipf's law since an embedding of an infrequent word (e.g., *okapi*) can acquire the relationship to other words from the usage of the more frequent and similar word (e.g., *zebra*) when they are learned as similar words in a model.

Now, how can we train the model to obtain embeddings that can represent the word meanings? Bengio et al. took a neural network based approach: the neural probabilistic language model [3]. The model is based on a feed forward neural network, where each word has its embedding. Figure 1.3 shows the architecture of the neural probabilistic language model. The vectors of the context words (words seen in a certain word window) are concatenated in the input layer, and we project

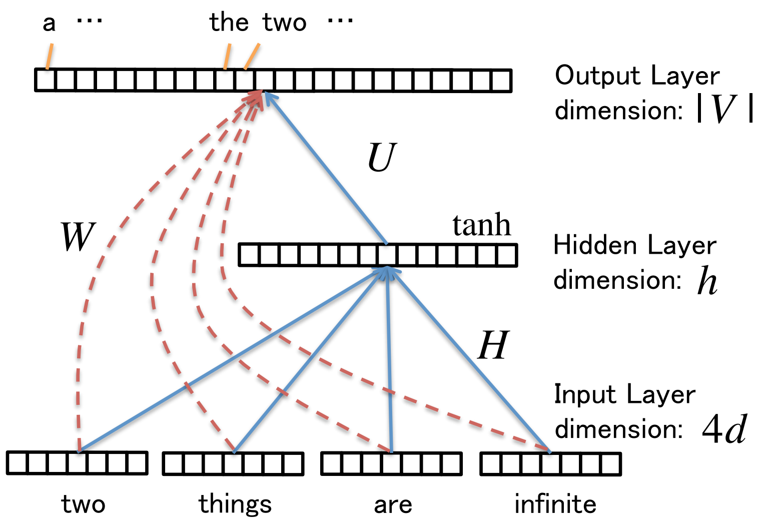

a ⋯
the two ⋯
Output Layer dimension: $|V|$
$U$
tanh
Hidden Layer dimension: $h$
$W$
$H$
Input Layer dimension: $4d$
two
things
are
infinite

Figure 1.3: Neural probabilistic language model by Bengio et al. [3]

the concatenated vector to the middle layer using the matrix $H$. We calculate the tanh of the vector in the middle layer, and using the matrix $U$, we project it to the output layer, where the values represent the unnormalized distribution for the word to be predicted. The model also has $W$, which projects the input vector directly into the output layer; however, the main architecture is similar to the feed forward neural network. In this model, vectors for words that share similar context words are learned as similar vectors. While $n$-gram models handled words completely separately, the neural network approach can augment training of the vectors from sequences of similar words. The experiment showed that the neural probabilistic language model performed better than the $n$-gram model in terms of predicting the next word from previous words.

One of the key characteristics of this approach is that the similarity between items are trained from preceding items through unsupervised learning. This approach is based on *distributional hypothesis*, which claims that we can extract the meanings (discussed in chapter 2 in detail). Another key characteristic is that there is no need to store all the fragments of sequences nor the set of items — we only need to store the vector representations. These characteristics are useful for Big data since the embedding approach does not require much additional memory or storage size as the data increase. When we receive new data, we only have to update the parameters (including embeddings) while the number of $n$-grams or the vector size of BOW increases; therefore, much research adopts this approach. Additionally, the embedding method, particularly when used with the neural network approach, often learns embeddings through online learning. Therefore, we can retrain embeddings from the learned models.

The above discussion addressed the effectiveness of adopting word embeddings. However, the embedding method also can be applied for entities that are accompanied with semantic relationships. Various entities are present in Big data, and the embedding method can ease the difficulties in the sparseness of entities as word embeddings did. Wang et al. embedded a knowledge graph of entities as well as a local context [92]. For a knowledge graph they used Freebase[2] [7], which is a large knowledge base contributed by community members. They combined the model learning the word meanings from the local contexts, and the model learning the word meanings from the knowledge base, which resulted in obtaining better representations. Yu and Dredze also took a similar approach, embedding the semantic resources from the Paragraph Database[3] [27] and WordNet as well as the local contexts [98]. Both studies aim to obtain embeddings for entities, which are better at capturing semantic relationships, when they have external knowledge.

However, in the era of Big data, we should often take a bottom-up strategy to handle various entities since taxonomies generally are not available and we typically handle domain-specific data. Although WordNet and Freebase are large datasets that provide semantic relationships among entities, they are far from complete in describing all the knowledge. Moreover, we want to handle various entities that are accompanied with diverse texts, rather than general terms. Therefore, we should use the information of entities to generate embeddings.

A problem here is that the information of entities is typically not structured as a knowledge graph. However, managing the entities with units or groups that we use for analysis is often sufficient. The decision must be made regarding which units or groups will be in the Big data analysis. This topic will be discussed in the next section.

### 1.1.3 Categories

We often focus representations for documents that are connected with entities. When we conduct clustering and classifying for analyses, we use representations of entities, often generated from text. However, we have many entities when we are dealing with Big data, and it is often the case that some of them contain little information compared to others, or that some of them appear in the dataset infrequently. We can use "categories" in these cases to gather various entities into certain groups in which entities are similar in some aspects.

Categories discussed here are groups where real-world entities are expected to share some properties. In the field of marketing, for example, demographic

[2]`https://www.freebase.com`; it is now read-only and they do not accept any contributions.
[3]`http://www.cis.upenn.edu/~ccb/ppdb/`

characteristics (age, marital status, family size) are traditionally used as categories. Engledow et al. pointed out similarities in demographic characteristics and purchase behaviors betweenthe United States and West Germany and indicated the existence of a homogeneous cross-cultural elite of consumers, which may have strategic importance for marketing [25]. Therefore, a motivation is to divide customers based on age, marital status, and other customer attributes. The way to divide customers is not necessarily via demographic characteristics. Using questionaires, Dawar and Parker divided customers into four clusters: brand-oriented, price-oriented, physical appearance-oriented, retailer reputation-oriented customers. They argued that this division is unrelated to the customers' background cultures [21]. In addition to customers, entities including products in e-commerce sites and articles in blogs or social media have groups which we can use for analyses; products have categories (e.g., fashion, PC), articles have tags or topics (e.g., politics, programming). These categories are given to navigate users to the needed information quickly, and they are also useful for optimization of the user experience. E-commerce sites use their product categories for search and recommendations, and blog services or social media use tags for recommendations and arrangements of their articles.

When we do not have explicit categories in advance in social media, we can use user groups in social networks for the unit of analysis. As Wasserman and Faust pointed out, traditional networks had boundaries such as classrooms or social clubs; however, recent networks have less well-defined boundaries [93]. Generally, users in social media also do not have explicit membership tags. Therefore, we should extract groups from a network. We often assume that connected users in social media are similar since people tend to be familiar with people who are similar to them, as the proverb "birds of a feather flock together" indicates. McPherson called such tendency "homophily", which is the phenomenon that people are more likely to associate and bond with others similar to them [62]. Also it is useful to observe activities of connected groups instead of simply similar users because we also expect the information diffusion in a network — so called *word of mouth*. There have been many studies about the relationship between information diffusion and a network. Rodrigues et al. investigated the diffusion of URLs in the social media. They showed that users consume information of others living near them and that there can be a tight relationship between the friendship link (mutual link) and the distance of users [80]. Bakshy et al. investigated the role of social networks in information diffusion and showed that the opportunity to obtain new information for users is increased by the posts of their acquaintances, while users tend to click URL links shared by close friends more than acquaintances [2]. These studies indicate that the information tends to diffuse in a connected manner between similar users, and we often call such groups *communities*. Then, how can we extract communities? Empirical studies have shown that many networks such as the Web, social networks, and biological networks have the community structure; therefore, we can extract the communities from a network structure. Several studies have proposed the algorithms of community extraction, and some of them are applicable to large networks.

Demographic characteristics, product categories, and communities discussed above are the units for the massive data analyses, and we call them simply "categories" here. We showed the overview of the relationship among categories, entities, and their text data in Figure 1.4. Categories discussed here generally share common properties or tendencies among entities. Sometimes they are manually given — shopping sites have categories over products (fashion, electronics, home appliances, etc.), news articles have topics (ubiquitous, greek economics, etc.), and some marketing techniques use demographic groups (15-24 male, 45-54 female, etc.). In this case, we have labels that describe the common properties within each group. Sometimes, categories are sometimes extracted from network or tendencies — clusters from entities' activity, and communities from network (social networks, citation network of academic papers, etc.). In this case, we do not have any explicit categories beforehand. However, we can find groups in which entities share certain properties or tendencies in some cases, as the notion of homophily indicates.

Now, why do we often need categories to grasp various entities? Although it is difficult to understand entities individually as the number of entities grows, we often use demographic groups / given categories / communities in the analysis. In the next section, we refer to Prototype theory, which indicates that the categorization is a fundamental function for humans.

By these categories, we can model the relationships among entities without using fine-grained taxonomy or structured data. . We often have categories regardless of the data size, and we can obtain embeddings for entities and categories, when we apply the embedding method for modeling both of them. Conversely, if we handle each category individually, we cannot deal with the changing trend and the large types of categories, each of which has few entities. For example, topics or keywords of articles in newspapers and academic papers have changed with the times. The topic "ubiquitous", which once referred to articles about small devices connecting to the wireless network, now relates to "wearable devices", referring to smart watches, and smart glasses, although they are labeled with different tags (as shown in Figure 1.5).

Another example is categories in e-commerce sites: the number of categories can be large (e.g. more than 10,000 in Rakuten Ichiba[4]), while the products are renewed periodically. If we take a traditional approach, topics in academic papers can be modeled as a set of entities and their text data, while we cannot use the similarity among topics for describing the relationship among entities in this case. If we use the embedding method, we can obtain embeddings of entities and categories simultaneously, and we can consider the local context of words, the similarity among entities, and the similarity among categories, all of which can improve the other embeddings. Thus, we can extract the similarity between the topic “ubiquitous” and the topic “wearable devices” from their embeddings, and we can obtain the embedding of a category that has few entities since the similarity or dissimilarity among entities can help the algorithm to learn the similarity among categories. Also, we can use the learned embeddings of categories to explore the similarity and the dissimilarity among them; it can benefit community analyses in social media. Therefore, it can be beneficial to take the embedding approach to model categories in order to handle various entities with labels, categories, and communities.

### 1.1.4 Prototype theory

Categories are not just groups that share arbitrary attributes, but they can be used for the unit of analysis. The necessity of using categories can be related to humans’ cognitive process of categorization.

Prototype theory, one of the influential hypotheses in cognitive science, claims that humans understand concepts from their prototypes, rather than rules, and classify things using learned prototypes. Homa et al. found the phenomenon that students learned prototypes when several dot patterns are shown to them [40]. They showed to the students dot patterns, which are distorted from the original patterns, and students were asked to classify them. After several days, the students were again asked to classify dot patterns, and they could classify them more accurately when the pattern were close to the prototypes, even though they had never seen the prototypes before.. Later Rosch et al. formulated the basic concept of prototype theory [83].

A prototype has typical or ideal features of its category, and it is sometimes a central member of the category, or a person’s best idea about a category. Rosch, who developed prototype theory, asked college students to rate how good each item listed is as an example of the category *furniture*, and she ranked the items [82]. A chair and a sofa were ranked as the best examples, a lamp was ranked number 31, and a telephone was ranked number 60, as the worst example. In this case, the

[4]www.rakuten.co.jp

prototype of the category *furniture* is a chair or a sofa, but the important point is that a graded categorization is universal in humans, even though the rank itself may vary in languages, cultures, or even in individuals. Prototype theory can be supported by several experiments: the experiment by Homa et al. showed that students learned prototypes of dot patterns showing distorted patterns [40, 41], and Minda et al. discovered that simulations of the basic prototype model fit well to the learning of visual categories through simulations of the basic prototype model [66].

Prototype theory claims that we categorize objects without defining the boundaries of concepts; we recognize concepts or categories based on the central idea of them. The central idea or object of the category is formed as we observe many objects or things. This process is robust since we do not need to reconstruct rules; we put the novel object distant from the prototype, and later we can form another category when we observe similarly novel things.

Prototype theory suggests that humans' cognitive process needs categories to grasp entities. The experiments mentioned above indicate that humans build up categories to distinguish even dot patterns. We can find a relationship between this cognitive process of humans and the usage of categories, which we apply to marketing and various analyses instead of investigating entities individually. Thus we can use categories in analyses of Big data, which have various entities.

One of the possible reasons we use categories is that categories enable robust analysis. The central idea of categories can reflect properties that most of the entities in the categories have in common. The notion of graded categories in prototype theory can also enhance the robustness for analyses since categories are often not fine-grained. Namely, we should consider the similarity among categories; some categories are similar when one category has several entities that are similar to the entities in the other category.

## 1.2 Purpose

The purpose of this thesis is to apply an embedding method for data with categories, and to explore its effectiveness on massive text data. As we discussed in this chapter, categories are groups that can be used for analyses on massive data, such as categories for products and communities for social media, and they do not necessarily have a hierarchy nor a taxonomy. We adopt the embedding method, which can learn both of the relationship among entities and categories from the dataset through unsupervised learning without any taxonomy. Since we train embeddings of words, entities, and categories at the same time from massive text

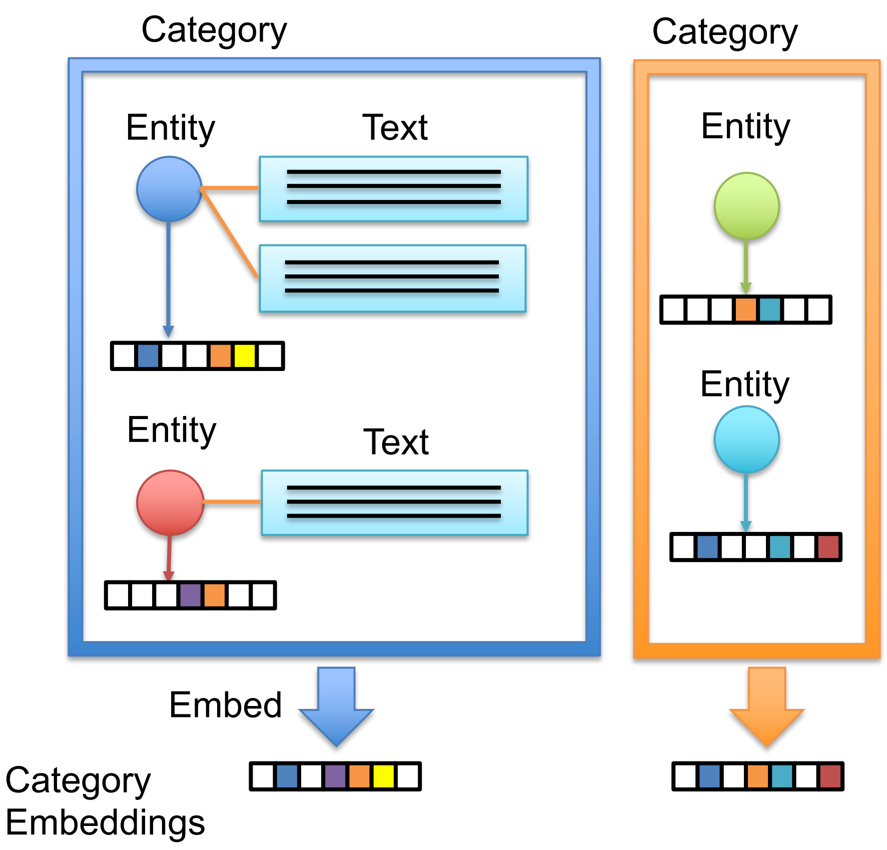


Figure 1.4: Overview of the embedding model, which considers categories.

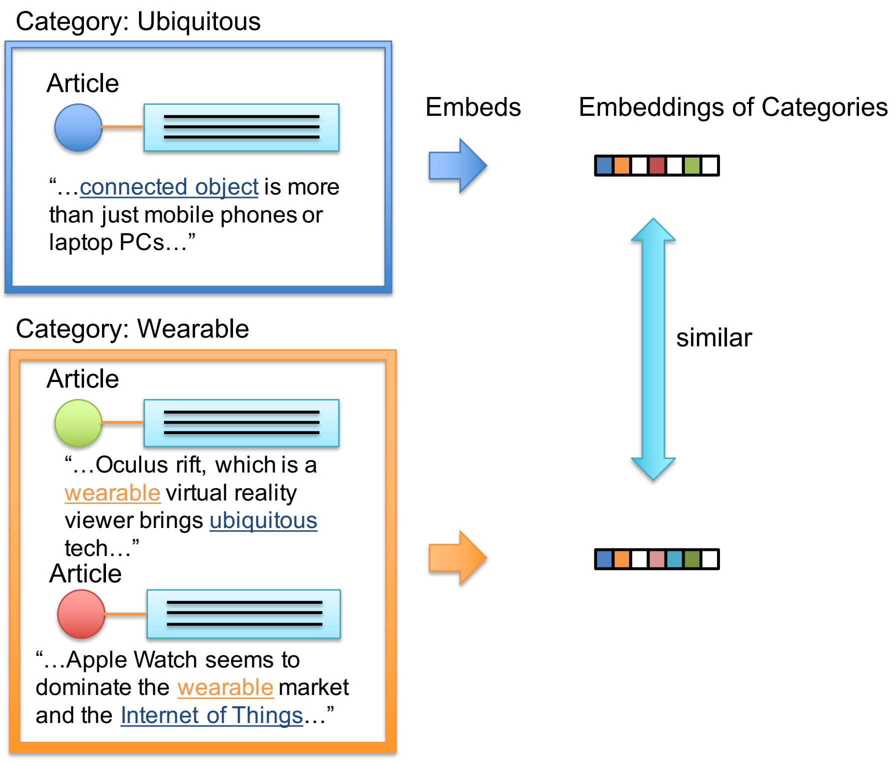

Category: Ubiquitous
Article
"…connected object is more than just mobile phones or laptop PCs…"
Category: Wearable
Article
"…Oculus rift, which is a wearable virtual reality viewer brings ubiquitous tech…"
Article
"…Apple Watch seems to dominate the wearable market and the Internet of Things…"
Embeds
Embeddings of Categories
similar

Figure 1.5: An example of a category.

data, the model can learn the similarity among entities from their text data and their categories, and also it can learn the similarity among categories from their members. Embeddings of entities belonging to the same category are learned to be similar embeddings. Therefore, we evaluate document representations or embeddings by the accuracy of experiments, where we infer the category of an entity. Finally, we compare the word embeddings among categories to determine how our model learned the relationship among them.

We focus on the embeddings of categories, rather than the categorization or classification methods, since we aim to obtain better representations by taking various categories into account. One might think that we can handle categories by adding them as attributes to the document representations (i.e., the *bag-of-words* model). With such representations, however, we cannot consider the similarities among categories and the diversity of the items in groups; an item can be similar to another item in a different category, or an item can be different from the other item in the same category. For example, the diversity of the items in the recommendation improves the performance [104]. Therefore, it is beneficial to consider the similarities of categories in the representations. We also aim to use the embeddings as inputs to various categorization, classification, and clustering tasks.

Since the embeddings are dense vectors, this method can avoid the bad effect from *the curse of dimensionality*, which various studies have indicated [42, 4].

## 1.3 Contributions

The contributions of this thesis are as follows:

- We proposed the new embedding method to generate better entity representations considering categories.
- We showed the effectiveness of the proposed method in the different datasets: e-commerce data and social media data.
- We proposed the method to analyze the difference in the local contexts in which the words are used among categories.

To the best of our knowledge, it is the first work that models entities with non-hierarchical categories learning embeddings of words, entities, and categories simultaneously. Since the method requires only the non-hierarchical categories, it can be widely applied for databases containing various entities (e.g. products, users, and text data). The obtained representations from our models can be used for understanding entities assisted by categories, which can help companies with their marketing and advertisement strategies.

## 1.4 Outline

In chapter 2, we introduce the related works. In chapter 3, we formulate the research question through discussion about categories and embedding methods. We introduce the embedding methods for categories in chapter 4, and show the effectiveness on the dataset in e-commerce sites. chapter 5 presents the application of the embedding methods to social media, and shows the effectiveness on community analysis. In chapter 6, we analyze users' contents on social media to explore the results of chapter 5 further. chapter 7 contains a discussion of the results of the embedding methods for categories. The conclusion of this thesis is provided in chapter 8.

# Chapter 2

# Related Work

Here we introduce the methods to represent documents including embedding methods, and we discuss previous works related to various text categorization tasks.

## 2.1 Document representations

First, we descibe traditional methods to represent documents, and point out the existing problems. Next, we describe embedding methods and explore how they can improve the problems.

### 2.1.1 Traditional approaches

The "bag-of-words" (BOW) model is one of the simplest ways to represent documents as vectors. In this model, a document is represented as a multiset (or *bag*) of its words, ignoring the word order. The vectors are generally very sparse, since this model assigns one dimension for each word. By representing documents as vectors, we can calculate the similarities among documents, taking cosines between document vectors.

To weight each word in a document vector, we usually use "TF-IDF" (term frequency - inverse document frequency) [85]. TF-IDF comes from the term frequency (TF) and the inverse document frequency (IDF). The TF is a measure of how frequently a word appears in a document. We typically use the count of a word divided by the length of a document as the TF of the word:

$$TF(w_i, d_j) = \frac{count(w_i)}{\sum_{w_k \in d_j} count(w_k)} \tag{2.1}$$

where $w_i$ is the $i$-th word of the document $d_j$. The IDF is a measure of how rare a word is in all the documents. It can be calculated from DF (document frequency), which is a measure of how often a word appears in all the documents. IDF is a logarithm of the reciprocal of DF:

$$IDF(w_i, D) = \log \frac{N}{|d \in D : w_i \in d|} \tag{2.2}$$

where $D$ is a set of documents, $d$ is a document from $D$, and $w_i$ is a word. Unlike TF, the value of IDF for a word does not vary among documents. TF-IDF is the product of TF and IDF:

$$TFIDF(w_i, d_j) = TF(w_i, d_j) \cdot IDF(w_i, D) \tag{2.3}$$

When the TF-IDF value for a word is high, the word can be thought of as a characteristic word for the document, since the word appeared in the document frequently, and it appears rarely in other documents.

One of the drawbacks of the BOW model is that we cannot consider the similarities among words. One can avoid this drawback by using *stemming* to some extent. Stemming algorithms convert inflected words to their word stems. A word stem is not necessarily a linguistically valid root; however, the inflected words or the derivatives should have the same word stem. One of the most famous stemming algorithms for English is the Porter stemming [76]. The Porter stemming has five steps, each of which has several converting rules for suffixes. For example, the word *sensibility* will be converted as the following:

Step 1c $(*v*)\text{Y} \rightarrow \text{I}$

- sensibilit$y$ → sensibilit$i$

Step 2 $([C](VC)^m[V])\text{BILITI} \rightarrow \text{BLE}(m > 0)$

- sensi*biliti* → sensi*ble*

Step 5a $([C](VC)^m[V])\text{E} \rightarrow \emptyset(m > 1)$

- sensibl$e$ → sensibl

where $C$ denotes a consonant (e.g., *s*) or a consonant cluster (e.g., *st*), $V$ denotes a vowel (e.g., *o*) or a vowel cluster (e.g., *oo*), brackets denote that there is zero or one of the element, $m$ denotes a repetitive number, parentheses in the left side denote the condition for preceding string of a suffix, and $(*v*)$ denotes that the stem contains a vowel. Stemming can reduce the dimensions of BOW vectors, but it is language-dependent, and it cannot deal with synonyms from different stems (e.g., *say* and *speak*, *warranty* and *guarantee*), and words from the same stem of which meanings are different (e.g., universal, university, universe → *univers*).

We can introduce *classes* to handle similarity or dissimilarity among words without using rules from prior knowledge. "Brown clustering" is one of such models

[13]. It assigns classes for words, and it assumes that the conditional probability of the current word given the previous word depends only on the classes of the current word and the previous word. Therefore the conditional probability and the joint probability of a text have the same form as a hidden Markov model (HMM).

$$P(w_i|w_{i-1}) = P(w_i|C(w_i))P(C(w_i)|C(w_{i-1})) \tag{2.4}$$

$$\log P(w_1, w_2, ..., w_n) = \sum_{i=1}^{n} \log P(w_i|C(w_i))P(C(w_i)|C(w_{i-1})) \tag{2.5}$$

where $C(w_i)$ is the class of the word $w_i$. We need to choose the good class assignment to maximize the joint probability; however, the optimization is known to be computationally hard. Generally, we perform Brown clustering in the agglomerative way, and we can pick up $k$ classes from the generated dendrogram. Classes of Brown clustering can be used for the NLP applications such as part-of-speech (POS) tagging [72], and dependency parsing [51].

### 2.1.2 Distributional hypothesis

The BOW model is widely used in NLP for its simplicity, But the major drawbacks of the BOW model are that it ignores word order and also the relationships among word types. The latter drawback can be solved using Brown custering, thus we can use a similar idea. As Brown clustering tends to assign the same class to words sharing their preceding words, we can regard words sharing their context as similar words. This idea, which is often called "Distributional hypothesis", was first pointed out by Harris [38]: "The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear" [55]. Although Harris did not claim that the meaning depends only on the context, the tendency that a word meaning depends on its context is useful for data-driven fields, since the word meanings can be learned without using prior knowledge. Thus, a large amount of recent research in NLP has proposed various distributional semantic models, assuming distributional hypothesis. DSMs can learn word representations based on the context, and massive data can help to learn better representations. At the same time, representations learned in distributional semantic models are dependent on corpora that we use for training.

### 2.1.3 Embedding methods

Distributional semantic models typically adopt embedding methods. They are called *embedding* methods because we embed words, sentences, or other concepts

Table 2.1: An example of a word-document matrix. Example taken from [22]

| Words | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ |
|---|---|---|---|---|---|
| *human* | 1 | 0 | 0 | 1 | 0 |
| *interface* | 1 | 0 | 1 | 0 | 0 |
| *computer* | 1 | 1 | 0 | 0 | 0 |
| *user* | 0 | 1 | 1 | 0 | 1 |
| *system* | 0 | 1 | 1 | 2 | 0 |
| *response* | 0 | 1 | 0 | 0 | 1 |
| *time* | 0 | 1 | 0 | 0 | 1 |
| *EPS* | 0 | 0 | 1 | 1 | 0 |
| *survey* | 0 | 1 | 0 | 0 | 0 |

into vectors or matrices. A word can be embedded in the following ways:

$$W : \{w_i, w_{i+1}, ..., w_k\} \rightarrow \mathbb{R}^n \tag{2.6}$$

$$W' : \{w_j, w_{j+1}, ..., w_l\} \rightarrow \mathbb{R}^{m \times n} \tag{2.7}$$

where $W$ and $W'$ denote embedding methods, $n$ denotes the dimension of the embedding vectors in $W$, $\{m, n\}$ denotes the dimensions of embedding matrices in $W'$, $\{i, j, k, l\}$ denote the indexes of the set of words. Some embedding methods use only embedding vectors, and others use both of the embedding vectors and matrices to represent the dependency relation as a product of matrices.

### (1) Matrix decomposition

First we introduce matrix decomposition methods. Deerwester et al. applied matrix decomposition to a word-document matrix [22]. This technique is called either "Latent Semantic Indexing (LSI)" or "Latent Semantic Analysis (LSA)". Table 2.1 is a word-document matrix, where each value is a count of a word appearing in a document. A singular value decomposition (SVD) of the matrix ($M \in \mathbf{R}^{m \times n}$) can be decomposed as follows:

$$\mathbf{M} = \mathbf{U\Sigma V^T} \tag{2.8}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{2.9}$$

where $\mathbf{U}$ denotes an $m \times m$ orthogonal matrix, $\mathbf{V}$ denotes an $n \times n$ orthogonal matrix, $\mathbf{\Sigma}$ denotes an $m \times n$ rectangular diagonal matrix, and $r$ denotes a rank of $\mathbf{M}$. The diagonal entries of $\mathbf{\Sigma}$ $(\sigma_1 ... \sigma_r)$ are non-negative real numbers, and are called singular values of $\mathbf{M}$. SVD is often used for low-rank matrix approximation. We can obtain the approximated matrix $\tilde{\mathbf{M}}$ by preserving the $k$ largest singular values and replacing other values by zero:

$$\tilde{\mathbf{M}} = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^{\mathbf{T}} \tag{2.10}$$

$$\tilde{\mathbf{\Sigma}} = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \sigma_k & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.11}$$

This approximation is proved to minimize the Frobenius norm of the difference between $\mathbf{M}$ and $\tilde{\mathbf{M}}$. The co-occurence matrix of words can be calculated as $\mathbf{M}\mathbf{M}^{\mathbf{T}}$; thus the approximate co-occurence matrix is:

$$\tilde{\mathbf{M}}\tilde{\mathbf{M}}^T = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^{\mathbf{T}}\mathbf{V}\tilde{\mathbf{\Sigma}}^{\mathbf{T}}\mathbf{U}^{\mathbf{T}} \tag{2.12}$$

$$= \mathbf{U}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{\Sigma}}^{\mathbf{T}}\mathbf{U}^{\mathbf{T}} \tag{2.13}$$

$$= \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \begin{bmatrix} v_1 & \cdots & v_m \end{bmatrix} \tag{2.14}$$

where $v_1 ... v_m$ denote $k$-dimensional vectors for words. We can regard these vectors as embeddings for words. Embeddings for the words in Table 2.1 are shown in Table 2.2. Each dimension $(d'_1, d'_2, d'_3)$ can be interpreted as a topic, and an entry value is considered to be a contribution ratio for the topic of a word. In a similar way, we can retrieve the document embeddings, which are $k$-dimensional vectors. The complexity of SVD is $O(mn^2) \quad (m \geq n)$, but for truncated SVD, which calculates the decomposition only for the $k$ largest singular values, a faster algorithm can be applied using the stochastic approach [37]. The word-document matrix is often large and sparse; thus, truncated SVD is useful since we only need to analyze the low ranked matrices to analyze. The LSI (LSA) technique can also improve the sparsity since similar documents or words can be merged into a single dimension.

While we can regard documents as a global context, a local context can influence the word meaning or functionality more. To extract the semantic representations, Bullinaria and Levy generated co-occurence matrices, where each

Table 2.2: Word embeddings from $\mathbf{U}\tilde{\mathbf{\Sigma}}$ of Table 2.1

| Words | $d'_1$ | $d'_2$ | $d'_3$ |
|---|---|---|---|
| *human* | -0.75 | 0.73 | -0.68 |
| *interface* | -0.67 | 0.35 | -0.92 |
| *computer* | -0.80 | -0.41 | -0.98 |
| *user* | -1.35 | -0.83 | 0.15 |
| *system* | -2.17 | 0.92 | 0.55 |
| *response* | -0.88 | -1.06 | 0.11 |
| *time* | -0.88 | -1.06 | 0.11 |
| *EPS* | -1.02 | 0.84 | 0.31 |
| *survey* | -0.60 | -0.53 | -0.03 |

entry shows the co-occurence in a context window, and they compared the performance of several distance measures: Euclidean, City Block, Cosine, Hellinger, Bhattacharya, Kullback-Leibler, Ratios, pointwise mutual information, and positive pointwise mutual information in their proposed method [15]. Among them, positive pointwise mutual information and Ratios performed the best. Ratios is the pointwise mutual information (PMI) without the logarithm. PMI is defined as follows:

$$PMI(X = x, Y = y) = \log \frac{p(X = x, Y = y)}{p(X = x)p(Y = y)} \tag{2.15}$$

where $X$ and $Y$ denote discrete random variables, and $x$ and $y$ denote outcomes belonging to $X$ and $Y$. This value is one of the co-occurrence measures of two elements. It can be interpreted as a kind of information gain when $X = x$ happens compared to the conditional information of $X = x$ when we know $Y = y$ happens:

$$\begin{aligned} PMI(X = x, Y = y) &= I(X = x) + I(Y = y) - I(X = x, Y = y) && (2.16)\\ &= I(X = x) - I(X = x|Y = y) && (2.17)\\ &= I(Y = y) - I(Y = y|X = x) && (2.18) \end{aligned}$$

where $I(\cdot)$ denotes the self-information: $-\log_2 p(\cdot)$. If $x$ always occurs with $y$, PMI is $I(X = x)$ since $p(X = x|Y = y) = 1$, and if $x$ and $y$ occur independently, PMI is 0. *Positive* pointwise mutual information (PPMI) is defined as follows:

$$PPMI(X = x, Y = y) = \max(0, PMI(X = x, Y = y)) \tag{2.19}$$

PPMI ignores less co-occurrence than the random occurrence. It assumes that $x$ and $y$ co-occur independently even though $x$ and $y$ co-occurred less because the corpus is insufficient, or either $x$ or $y$ is an infrequent word.

While PPMI captures semantic or syntactic aspects of words more than LSI(LSA), we cannot obtain document representations directly from PPMI. Next we introduce the neural network based approach, which can generate document representations from word embeddings.

### (2) Neural network based approach

The neural network based approach is one of the major embedding methods. Embeddings are parameters of neural network models, and we optimize them to train the models. In the neural network based models, embeddings are often called "distributed representations" since *each entity is represented by a pattern of activity distributed over many computing elements, and each computing element is involved in representing many different entities* as Hinton explained [28]. After Bengio et al. proposed the neural probabilistic language models [3], many algorithms and techniques have been proposed to improve neural network based models for training word embeddings. One of the drawbacks of the neural probabilistic language models is that the number of parameters is large, and the computation of training the model is difficult. Several attempts have been made to reduce calculation time. Morin and Bengio created a hierarchical binary tree of vocabulary from WordNet [69]. Morin and Bengio used the *is-a* relationships to create a binary semantic tree where a child node has *is-a* relationship to the parent node. Although the perplexity of the language model with a semantic tree was higher than the original one, the computation time was drastically improved. Mnih and Hinton extended this idea and created a similar binary tree in a bootstrap way utilizing the log-bilinear model, which requires less parameters [67]. The binary tree is generated from the word embeddings of the previous learned model doing agglomerative clustering among word embeddings. This model outperformed the original model, and it improved the performance during iteration to build the binary tree.

Instead of pursuing the better language model, Collobert and Weston tried to embed a sentence into a vector representation using a convolutional neural network (CNN) to apply the same model to several tasks in NLP [19]. CNN is inspired from the visual cortex in the brain and used for deep learning techniques for computer vision and image processing since the parameters can be reduced from fully connected neural networks. CNN has matrices which can be regarded as *filters*, and it takes convolution between two-dimensional pixels and the matrices, which results in obtaining images *filtered* several ways. CNN is often used with "max pooling", which extracts the maximum value from each two-dimensional window. Jarrett et al. concluded that max pooling is useful for CNNs since *max pooling alleviates the need of absolute value rectification* through the experiment on

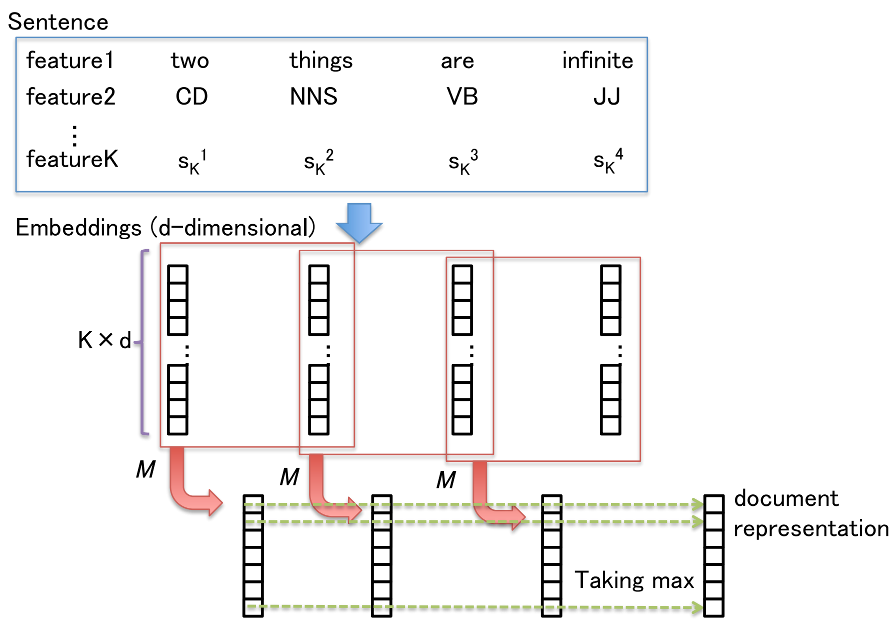

Sentence
feature1 two things are infinite
feature2 CD NNS VB JJ
featureK $s_K^1$ $s_K^2$ $s_K^3$ $s_K^4$
Embeddings (d-dimensional)
K × d
M
M
M
document representation
Taking max

Figure 2.1: The architecture of the CNN model proposed by Collobert and Weston [19].

several datasets [43], and Boureau et al. theoretically discussed the effectiveness of max pooling [8]. Collobert and Weston used this architecture for NLP, because CNN can solve the difficult point to generate document embeddings; the input length is variable despite the restraint that the dimension of embeddings is fixed. Figure 2.1 illustrates their architecture, which has 2 steps, the convolution step and the max pooling step. The word has $K$ features, and the model assigns a vector representation for each feature value; therefore, each word has $K \times d$ features. We take convolution with vectors of words in a window and the matrix $M$, and we obtain convoluted vectors. Finally, we take the maximum value from each dimension, and create a vector for the document (max pooling). We input this document representation to the fully connected neural network that solves a certain task, to train the whole network. This model can be used for multitask learning; however, it consumes much time for training, and the performance is lower for later models. However, the convolutional approach is improved in recent research as the optimization techniques develop. Recent models aim to improve the accuracy of tasks rather than to realize multitask learning.

Another way to obtain document embeddings is to use recursive models. Socher et al. used modified recursive autoencoders and predicted sentiment distribution over sentiment classes [89]. An autoencoder is a feed-forward neural network, which is trained to generate the same values in output as the input values. The

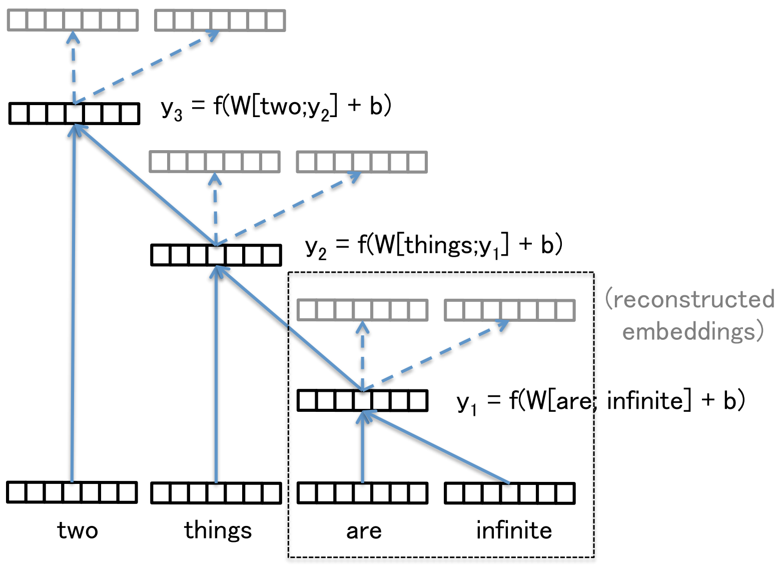

$y_3$ = f(W[two;$y_2$] + b)
$y_2$ = f(W[things;$y_1$] + b)
(reconstructed embeddings)
$y_1$ = f(W[are; infinite] + b)
two
things
are
infinite

Figure 2.2: An example of the recursive autoencoder proposed by Socher et al. [89].

number of nodes in the middle layer is usually smaller than the number of input/output nodes; thus, we can regard an autoencoder as a data compression algorithm. Principal component analysis (PCA) is also regarded as a data compression algorithm, but autoencoders are more flexible since they are non-linear. They constructed a tree from a sentence, where each node has an embedding, and each pair of nodes is an input of an autoencoder, and the values in the middle layer are an embedding for the phrase (Figure 2.2). Traditional autoencoders are unsupervised; however, one of the purposes of this model is to predict sentiment of sentences. Socher et al. modified the algorithm, introducing a matrix to map an embedding to a distribution of sentiment classes, as in Figure 2.3. The root node is the document representation of a sentence, and the matrix ($W^{\text{label}}$) converts to the distribution of sentiment classes. In this way, they were able to combine unsupervised learning (autoencoders) and supervised learning (sentiment prediction) in the model. The drawbacks are that words of the deep nodes in a tree are compressed many times and can lose information, and that it is difficult to choose the best weights for the reconstruction error and the cross-entropy error since it is task dependent. Socher et al. used recursive neural network with parse trees, merging embeddings of dependent elements, and they improved above-mentioned problems [88]. They applied the same architecture to image processing of natural scenes and natural language text, and the model outperformed previous models in image processing. The POS tagger using this model performs slightly worse than previous models, and the calculation time is much longer than the existing tool.

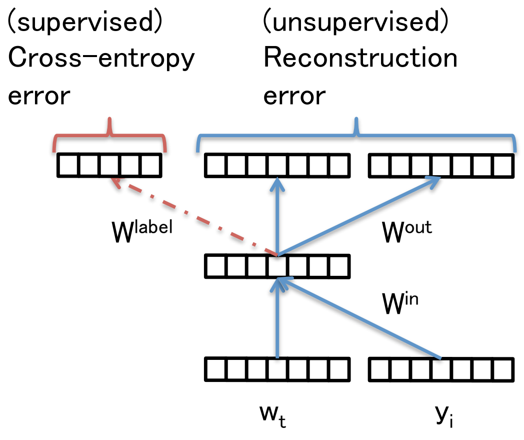


Figure 2.3: Modification to an autoencoder for semi-supervised learning

In addition, it is also difficult to obtain correct parse trees.

While the computation of neural network based models is often difficult, Mikolov et al. improved the training time drastically with their proposed Skip-gram model and Continuous Bag-of-Words (CBOW) model [64]. These models are implemented in the tool "word2vec", which led to many studies of word embedding methods since the models succeeded in producing high-quality word embeddings. The middle layer of word2vec models is linear, and the models adopt a simplified approximation of the output layer; thus the models gain speed. Word2vec is applicable to a large amount of data (e.g., Wikipedia). Therefore, we can obtain better representations than the models above. Mikolov et al. also showed that word2vec embeddings reserve the semantic and grammatical relationships (e.g., $work : worked = sit : sat$, $man : king = woman : queen$) to some extent through the analogy test.

To benefit from good word embeddings of the Skip-gram and CBOW models, Le and Mikolov extended those models and proposed "Paragraph Vector" models [54]. A "paragraph" does not necessarily mean a distinct section of a document here, but a sentence or several sentences. The Paragraph Vector models consist of two models: the paragraph vector with distributed memory (PV-DM) and the paragraph vector with distributed bag of words (PV-DBOW) as shown in Figure 2.4. The PV-DM model is inspired from word2vec's CBOW model, and it predicts a target word from a paragraph combined with words in a context window. The PV-DBOW model is inspired from word2vec's Skip-gram model. The PV-DBOW model predicts words in the window given the paragraph vector. In the models, each paragraph has a unique vector in the same way as word vectors, and analogous to word vectors, paragraph vectors are expected to be similar when paragraphs have similar meanings. The paragraph vector can be used as a fixed-length feature vector for tasks such as classification and clustering. Le and Mikolov showed that these models outperformed the recurrent neural network based model and the CNN model in the sentiment prediction task. Since the word2vec based models learn embeddings fast enough to apply to massive corpora in an unsupervised way, these models can compute embeddings fast enough to process massive text data. The Paragraph Vector models learn a paragraph embedding as a context of word sequence in a paragraph without using any taxonomy or external knowledge; therefore, the models can produce good representations of entities in today's massive data. We will discuss these models further in chapter 3.

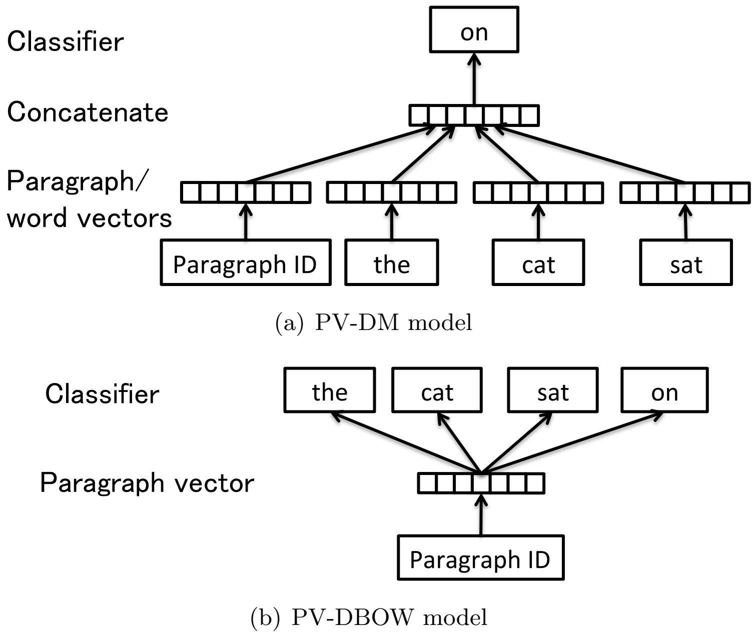


Figure 2.4: Paragraph Vector Models

### (3) Other approaches

Pennington et al. proposed GloVe, which is a log-bilinear model that can obtain good word embeddings in terms of the analogy [73]. They aim to improve the analogy test of word2vec, utilizing both of the global co-occurence (used in LSI) and the local context windows (used in word2vec). They model assigns both of a word embedding and a context embedding to each word, and it assumes that the word $k$ is useful in the context where the word $i$ and the word $j$ appear, when the value of $\frac{P(w_k|w_i)}{P(w_k|w_j)}$ is much larger or smaller than 1, rather than the value of $P(w_k|w_i)$. They aim to improve the analogy task using the difference between word embeddings, and model $\frac{P(w_k|w_i)}{P(w_k|w_j)}$ as follows:

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P(w_k|w_i)}{P(w_k|w_j)} \tag{2.20}$$

where $F$ is the function, $w_i$ is the word embedding of $i$-th word, $\tilde{w}_i$ is the context embedding of $i$-th word. Pennington et al. adopted the exponential as $F$ and supplementary terms for simplicity. They claimed GloVe outperformed word2vec in analogy tasks.

However, Levy et al. showed that there is no significant difference among the performance of GloVe, Skip-gram, and PPMI with SVD [57]. On the contrary, GloVe did not outperform any of other models in their experiments. Levy et al. also pointed out that the Skip-gram with negative sampling (an approximation method for the output layer) can be regarded as implicit matrix factorization of *Shifted* positive PMI (SPPMI), which has an additional term to PPMI [56]. SPPMI is defined as follows:

$$SPPMI_k(w,c) = \max(0, PMI(w,c) - \log k) \tag{2.21}$$

where $k$ denotes the hyper parameter, $w$ denotes the target word, and $c$ denotes the context. SPPMI assumes less co-occurrence to be independent than a certain degree of occurrence, $\log k$. The hyper parameter $k$ corresponds to the number of negative samples of the Skip-gram model. Although SVD of the SPPMI matrix can reproduce the result of the Skip-gram model, the Skip-gram model outperforms the matrix decomposition method in the syntactic analogy experiment. This difference can come from the handling of low frequency words; PMI-based methods usually exaggerate them.

Although GloVe and PPMI with SVD can produce good representations of words, there are no direct extensions on these models to produce document representations. In that sense, it is relatively easier to extend neural network based models such as word2vec and Paragraph Vector models.

## 2.2 Categorization and community analysis

As we discussed in chapter 1, categories are groups that people regard as meaningful groups in a dataset. They can be entities sharing the same attributes or labels, or extracted as communities from a network. We obtain embeddings of entities and categories from our model, and we use them as an input for NLP tasks, especially for classification. When entities have explicit labels or classes, we often call such classification tasks categorization classification. In the first part of this section, we introduce traditional methods of categorization and its applications. When entities connect with each other but they do not contain explicit attributes or classes, we usually extract groups from the network and then classify the unknown entities. Such groups are often called *communities*, and we introduce methods of community extraction and its applications in the latter part in this section.

### 2.2.1 Categorization in use of machine learning

In this part, we introduce categorization from text data. We can categorize entities with text using classification algorithms. Classification for more than two classes is called multiclass classification. After we briefly describe the algorithms of multiclass classification, we introduce applications of multiclass classification.

#### (1) Multiclass classification

Text categorization has been approached in various ways. First we introduce classification algorithms that can deal with multiclass settings. These algorithms often take probabilistic approaches.

The naïve Bayes model is a probabilistic classifier that predicts the class from attributes of an instance [84, pp. 808–809]. Let $x_1, ..., x_n$ be attributes and $C_k$ be the class $k$. The probability of the class $k$ given the attributes can be decomposed as the following using Bayes' theorem:

$$p(C_k|x_1, ..., x_n) = \frac{p(C_k)p(x_1, ..., x_n|C_k)}{p(x_1, ..., x_n)} \tag{2.22}$$

Since the model assumes that the attributes are conditionally independent each other, we can decompose Equation 2.22 as follows:

$$p(C|x_1, ..., x_n) = \alpha p(C_k) \prod_i p(x_i|C_k) \tag{2.23}$$

where $\alpha$ is the normalizing factor. We often calculate $p(C_k)$ from the frequency of the class $k$, and we count the frequency of $x_i$ in the class $k$ for $p(x_i|C_k)$. When the attributes are sparse, we can utilize Laplace smoothing:

$$p(x_i|C_k) = \frac{\#x_i + \lambda}{m + \lambda n} \tag{2.24}$$

where $\#x_i$ is the count of $x_i$ in the class $k$, $m$ is the count of the instances in the class $k$, and $\lambda$ is the paramter for smoothing. This algorthm can be used for NLP, where each attribute is a word, and it is applied for multiclass classification on texts.

Logistic regression can be extended to a multiclass classifier [5, pp. 209–210]. The logistic function is expressed as follows:

$$Logistic(z) = \frac{1}{1+\exp(-z)} \tag{2.25}$$

If we put weights for each attribute in each class and values of attributes in z, we obtain the following multiclass logistic regression:

$$\begin{aligned} p(C_k|x_1, ...., x_n) &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} && (2.26) \\ a_k &= \sum_i^n w_{i,k} v_i && (2.27) \end{aligned}$$

where $a_k$ is the activation for class $k$, $w_{i,k}$ is the weight for the attribute $i$ and the class $k$, and $v_i$ is the value of attribute $i$. Unlike the naïve Bayes model, we have to train logistic regression to obtain optimal parameters. Gradient methods are widely used for this purpose.

Next, we extend binary classifiers to multiclass settings. Typically, we use a set of binary classifiers for multiclass classification. Among binary classifiers, support vector machines (SVM) are often used since the performance is generally better than the other classifiers [20]. SVM maximize the margin from each instance to the separating hyperplane; thus, SVM model is considered to be robust. However, it is not easy to extend the SVM model to a multiclass classifier, unlike the above models. Two approaches can be used to extend a binary classifier to a multiclass classifier: *one-versus-the-rest*, and *one-versus-one* [5, pp. 182–184]. The *one-versus-the-rest* classifier uses $n-1$ binary classifiers, each of which classifies its class $k$ ($C_k$) from other $n-1$ classes, where the number of classes is $n$. We assign the class $C_k$ if the classifier for $C_k$ activates and other classifiers do not, and we assign the class $n$ if none of the classifiers activate. The drawback of this approach is that we cannot assign any of $n$ classes if the multiple classifiers activate. The *one-versus-one* classifier uses $n(n-1)/2$ binary classifiers, each of which discriminates the class $k$ ($C_k$) from the class $l$ ($C_l$). We assign the class $C_k$ from the majority of the classifiers; thus, the one-versus-one classifier is more robust than the one-versus-the-rest classifier. The drawbacks of this approach are that the number of classifiers explodes when the number of classes increases, and that we cannot assign any classes if none of the classifiers activate.

It is another option for multiclass classification to make a hierarchy of binary classifiers. A decision tree is a tree of decision rules. A decision rule is not necessarily a binary classifier; however, CART [12] and C4.5 [77], where each node splits the feature space, are commonly used. Random Forests employ many decision trees and use the result of the majority [11]. Yuan et al. proposed the hierarchical SVM model to categorize video genres, which outperformed decision trees and the one-versus-one SVM classifier [99].

As the number of classes increases to 100 or 1000, it is generally difficult to handle the data, even with the above methods. In such situations, we commonly use $k$-nearest neighbors ($k$-NN) [5, pp. 124–127]. Let us assume there are $N$ data points with $n$ classes, and we have a new data point. The $k$-NN algorithm infers from the $k$ nearest points to the new data point. There are several ways to infer the class; one can infer from the majority of the nearest points, or one can consider the distance to the data point to avoid adopting a frequent class. This algorithm works for many classes, but drawbacks are that the result of the $k$-NN is affected significantly by the local structure, and the computation is expensive when the data points are many.

In recent years, *deep learning* techniques have been developed and recurrent neural networks and convolutional neural networks are used for classification. Lai et al. adopted the recurrent convolutional neural network approach, which has a convolutional layer and a max pooling layer [52]. Their model has a full connected neural network between the max pooling layer and the output, and each value of the output vector is an unnormalized distribution for a class. Their model outperforms previous neural network models (except the Paragraph Vector models) as well as traditional methods such as BOW with SVM. Kim proposed a similar architecture to sentence classification tasks [47]. Kim adopted word2vec's word vectors in the initial state[1], and the model outperformed previous models in some NLP tasks. These classification techniques are tightly linked to the model, and the convolutional neural networks are often used for multiclass classification with upto 100 classes. Other neural network models can utilize $k$-NN or other traditional methods, feeding in embeddings.

### (2) Applications of categorization

We describe the application of the categorization techniques here. Mooney and Roy adopted the Naïve Bayes approach to improve the content-based recommendation for books[68]. They used the user ratings as the positive or negative labels for books. The binary classifiers are trained to predict the quality of books. While such recommender systems using "collaborative filtering" [32], which uses similar users' history, suffer from lack of feedbacks of new items (this is called the "cold start problem"), this classifier can tell the popularity from the content. This approach can perform less for the dataset, where each entity has small text data

[1]The vectors trained by another model or task are often called "pre-trained". In the deep learning field, it is common to use pre-trained vectors to obtain good optimization.

and the number of evaluated data is also small. It is difficult to handle diverse text since they treat each word as an independent attribute, and the evaluated data are usually much smaller than the non-evaluated data; therefore, the learned model does not have good scores for infrequent words in the training set. In such situations, the embedding method can improve the performance since it can learn the similarity among words and entities. Joachims et al. used SVM to categorize documents in Reuters dataset and Ohsumed corpus, comparing it to Naïve Bayes and C4.5 [44]. In their dataset, the number of categories is small (less than 100), and one has to train many more classifiers as categories increase. Therefore, it is difficult to apply for a dataset with a large number of categories.

Weigend et al. took the neural network approach and they introduced "meta-topic", where each topic belongs [94]. Their model consists of two parts: a meta-topic network that predicts the probability of a meta-topic given the input, and a network that predicts the probability of a topic given the meta-topic and the input. Related to e-commerce applications, Shen et al. applied a graph algorithm along with text categorization techniques to classify items to hierarchical categories in eBay [87]. Although we aim to apply our model to non-hierarchical labels, we can use the category hierarchy into models and improve the performance with our model.

Several recently studies have used CNN, which is similar to the model discussed above (as shown in Figure 2.1). Xu et al. used CNN for text hashing, where the hash value of a sentence is similar to that of a similar sentence [95]. Although traditional hashing algorithms only guarantee that the probability of identity between entities is high when the hashed value is the same, the hashed values by their algorithm are similar when the entities are similar. Xu et al. also applied CNN to the short text clustering, where the embeddings are used for the input of $k$-means clustering [96]. Since the performance of neural networks often depends on the initialization, most CNN approaches use "pre-trained" embeddings. Our model can enhance the performance using embeddings trained by our model as pre-trained embeddings.

While we discussed the techniques and the applications for multiclass classification so far, we aim to produce good representations of entities with our model. Although our model does not handle hierarchies of categories and our model does not include the classification method itself, our model can be applied to the above works to improve the performance.

### 2.2.2 Community analysis

First, we describe the methods of community extraction from a network. These methods can be applied to most of the networks since they do not have many assumptions to data. Next we introduce community analyses. As we aim to use embeddings of entities for community analysis and to classify unknown entities into communities with our model, we introduce several analyses using communities in networks and attempts to classify entities into communities from text data of entities.Community extraction and analysis is a growing field in the area of social network analysis, particularly after the popularization of social media.

#### (1) Community extraction

A community is a subgraph, in which nodes have dense connections. Therefore, it is natural to find cliques, which are fully connected subgraphs, and extend them to communities. The $k$-Clique Percolation is one of such community extraction algorithms, where we find cliques with $k$ nodes and regard adjacent $k$-cliques as the same community [23]. This algorithm is simple; however, $k$-cliques with $k \geq 4$ are rarely found in a large network. In addition, the amount of nodes connected with a single node is too large in a scale-free network to find adjacent cliques, which can lead to extracting a large amount of tiny communities. We can also express a network as a Laplacian matrix and use the spectral clustering technique on the matrix [17]. The Laplacian matrix ($L$) is defined as follows:

$$L_{i,j} = \begin{cases} -1 & (n_i \text{ is adjacent to } n_j) \\ degree(n_i) & (i = j) \\ 0 & (\text{otherwise}) \end{cases} \tag{2.28}$$

where $n_i$ is the $i$-th node. We reduce the rank using eigendecomposition, and we can apply clustering algorithm (e.g. $k$-means) to the row vectors in a low-ranked matrix. This method can avoid extracting tiny communities. In addition, we have to decide the number of communities in most of the cases, and the algorithm is too complex to process a large network such as a social network.

Instead of extracting densely connected subgraphs directly, we can cut connections that are expected to bridge among communities. Girvan and Newman proposed a community extraction method scoring edges by means of the *edge betweenness* [30]. The edge betweenness of an edge is defined as the number of shortest paths from all vertices to all others that go through the edge, and an edge that has high edge betweenness can be seen as a bridge between communities. This method consists of two steps: removing the edge with the highest edge betweenness and recalculating the edge betwennesses for the rest of the edges. Until all the edges are removed, these two steps are repeated. The complexity of this algorithm is $O(|E||V|^2)$, where $|E|$ denotes the number of edges and $|V|$ denotes the number of vertices. This method is often called "Newman-Girvan clustering". Later Newman proposed the measure of the *goodness* of a graph partition, called *Modularity*, where vertices are connected with each other densely within every community, and vertices have sparse connections to vertices in other communities [71]. Modularity of a graph with $n$ nodes and $m$ edges is defined as follows:

$$Q = \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) \tag{2.29}$$

where $A$ denotes the adjacency matrix, $k_i$ denotes the degree of the vertex $i$, and the $s_i s_j$ represents the membership that indicates whether $i$ and $j$ belong to the same community ($s_i s_j = 1$) or not ($s_i s_j = -1$). Some research includes modularity in Newman-Girvan clustering, which is also called "Newman clustering". It is difficult to compute the edge betweennesses in every repetition. Therefore, the Newman clustering cannot be applied for large graphs such as today's social networks and the Web.

One can take other approaches to maximize modularity; however, the maximization of modularity is known to be NP-complete as Brandes et al. proved [10]. Therefore, several greedy approaches, such as Newman [70] and Good et al. [33], have been proposed. Among such algorithms, the Louvain method is a simple and effective method for detecting communities in a large network [6]. It conducts greedy optimization of modularity in several steps and it is applicable to large networks. Modularity based clustering techniques are the most popular community detection methods for large networks since extracted communities are generally not too small to analyze.

One of the drawbacks of the modularity based methods is that vertices cannot belong to multiple communities since they are essentially graph partition methods. Ahn et al. proposed "Link communities", which assigns communities to edges; therefore, each member can belong to multiple communities [1]. Although this algorithm is applicable to a large network, it adopts the minimum distance in the hierarchical clustering of communities. Thus, one of the communities tends to explode and others are very small.

### (2) Community analyses

Before the expansion of social media, most researchers used topological structure of social networks and ignored the node attributes that are often heterogenous. This is because datasets of social network analysis, such as *Zachary's Karate Club* dataset[2] and *American College Football* dataset[3] , had not included rich attribute information of nodes [103]. However, it is not so difficult to extract attribute information of current social media users. Some research has improved the performance of community detection using both topological information and attribute information on social network services. Tang et al. proposed a joint optimization framework to integrate multiple data sources, which outperformed existing representative methods of community detection in social media [91]. Guneman et al. proposed a method for detecting homogeneous communities, that is, communities that are densely connected and have similar attributes, by integrating results of subgraph clustering and subgraph mining [36].

## 2.3 Document representations with categories

To obtain better representations of entities, we can utilize categories or relationships among entities if such data are available. One can use taxonomy to produce good representations of entities considering their relationship. WordNet[4] [65], which is a lexical database for English, has groups of cognitive synonyms connected by conceptual-semantic and lexical relations. Resnik [79] searched *classes* of words that follow a certain verb, and generated object classes for verbs. This approach handles similar words (e.g. *okapi* and *zebra*) and gathers them into the same class (e.g., *animal*). One of the serious drawbacks to this method, however, is that we need a taxonomy that has a large coverage of vocabulary. WordNet has only English terms; therefore, we have to use other taxonomies to apply for texts in different languages. In addition, there can be technical or colloquial terms that do not agree with the WordNet taxonomy, and it is difficult to select connections between words, which suits the purpose of an application.

In contrast, categories including tags and communities, which do not necessarily have tree structures, can tie entities weakly. It does not mean that the entities are other kinds if they belong to different categories since categories can be alike. We aim to use this weakly structured data, and we take the embedding approach to achieve this purpose.

[2]A social network of a karate club, of which the instructor left with half of the members later and formed another club. Zachary studied this network to model the fission from the social network among members [100].

[3]A network where each edge represents a game match between teams. Girvan and Newman used this network to validate their community extraction algorithm regarding divisions as communities [30].

[4]`https://wordnet.princeton.edu`

# Chapter 3

# Category-enhanced Embedding Method

In this chapter, we discuss the definition of entities and categories. Then we describe the proposed model, which captures the similarity among entities enhanced by category information.

## 3.1 Definition of entities and categories

First, we define entities discussed in this thesis. We assume an entity has the following properties:

**Assumption** Properties of an entity

1. An entity has a text item or items, which describe the entity or are generated from the entity.
2. A set of entities has certain criteria for grouping, which humans can use to understand the entities.

We handle massive text data, where various entities have their own text items, which we also call "contents". Although we do not limit the amount of text items, we focus on entities with text fragments, for which traditional methodologies have difficulties. In addition, we assume that entities can be divided into groups, which help humans to understand the entities, since we aim to use such groups for obtaining better representations of entities. We do not make an assumption on the number of groups. However, a group should have multiple entities.

Second, we define categories:

**Definition** Categories

1. A category is a set of entities, divided by certain criteria.
2. An entity joins one or more categories.
3. (Explicit Categories) An explicit category corresponds to a property or a tag of entities.
4. (Implicit Categories) Implicit categories are generated from the criteria for grouping of entities.

Categories help humans to understand the entities by grouping and extracting a part of the essential properties or aspects of entities. As discussed regarding humans' categorization in chapter 1, humans use the ability to categorize things to understand them. Therefore, this definition almost always holds true for entities in the real world.

Explicit categories make up a set of entities that shares a property or tag. Categories in e-commerce sites and tags in blog articles are examples of explicit categories. Conversely, implicit categories make up a set of entities that does not share a property or tag explicitly; however, the entities share characteristics within a category. Communities in networks are examples of implicit categories. Communities are generated from community detection algorithms, and a large amount of research supports the claim that entities in the same community are similar and share some characteristics.

If we use categories for analyses, we often want to use their labels that describe their characteristics or central idea concisely. For explicit categories, we can use property names or tag names for labels of categories. For implicit categories, we cannot use shared property names of entities. However, we can use the descriptions of entities and extract the keywords of categories, which can give the characteristics of implicit categories. To summarize, the definition of labels is as follows:

**Definition** Labels

1. A label describes the characteristics of a category.
2. A label is a name of the shared property or tag of an explicit category, or it is generated from text items or descriptions of entities of an implicit category.

We can formulate the relationship between entities ($E$) and categories ($C$) as the following:

$$^{\exists}f : E \to C,\ ^{\forall}e_i \in E,\ ^{\exists}c_j \in C \quad \text{such that} \quad f(e_i) = c_j \tag{3.1}$$

$$^{\forall}e_i \in E,\ ^{\exists}t_j \quad \text{such that} \quad t_j \in e_i,\ t_j = w_1 w_2 ... w_{n(j)} \tag{3.2}$$

where $t_j$ denotes the text fragment of an entity, $w_l(1 \leq l \leq n(j))$ denotes the $l$-th word in the $T_j$, and $n(j)$ denotes the length of $T_j$. Practically, we suppose $n(j)$ should take a value up to several thousand since the consistency in a text can decrease for the long length of texts. The projection $f$ corresponds to the certain criteria to divide the entity set. Entities share a property in an explicit category. Therefore, we can realize the projection $f$ using properties of entities as the following:

$$g : E \to P, {}^{\forall}e_i \in E, {}^{\exists}p_k \in P \quad \text{such that} \quad g(e_i) = p_k \tag{3.3}$$
$$h : P \to C, {}^{\forall}p_k \in P, {}^{\exists!}c_j \in C \quad \text{such that} \quad h(p_k) = c_j \tag{3.4}$$
$$f : E \to C, {}^{\forall}e_i \in E, {}^{\exists}c_j \in C \quad \text{such that} \quad h(g(e_i)) = c_j \tag{3.5}$$

where $P$ denotes properties. For implicit categories, there are no explicit $P$. However, we assume that the projections $g$ and $h$ can be done with the projection $f$, which is the algorithm for dividing the entity set, such as accomplished with the community detection algorithm.

## 3.2 Requirements for the category-enhanced model

Our goal is to obtain better representations for entities. Entities in the real world are not often well-structured. Therefore, we cannot assume we have rigid taxonomies. However, we can regard entities as weakly-structured since each entity belongs to a category, as we assumed in the previous section. We call a model utilizing category information as well as entities' contents "the category-enhanced model". We want to apply this model for entities either with explicit or implicit categories.

We also want to embed category information into entities' representations, which can be generated from text items. Although it seems easy to make entities hold category information, we should consider similarities among categories; some categories are similar, some are dissimilar. Therefore the category-enhanced model should produce representations for categories utilizing the information of entities, as well as representations for entities enhanced by categories.

Additionally, we use this model for massive data, where each text item can be short. The training method of this model is necessarily applicable to a large amount of data, since the number of entities and categories can be large. The category-enhanced method should also produce representations from text fragments in which various words are used. This means that the model should consider the similarities among words.

Once we have trained the model, we want to obtain representations just from text items in case we do not have the category information and any prior entity information (e.g., new entities). Moreover, we would want to infer the possible category for the entity from the representations. Therefore the trained model should produce representations from text items, and the representations should enable us to infer the possible categories more accurately than traditional methods.

In summary, the requirements are as follows:

**Requirements** The category-enhanced model

1. The model learns the representations of entities from categories and the representations of categories from entities.
2. The training method of the model is applicable to massive data, where the number of entities and categories can be large.
3. The model produces representations considering the similarities among words.
4. The trained model produces representations from text items of new entities, and the representations enable us to infer the possible categories.

## 3.3 Our methodologies

To satisfy the requirements for the category-enhanced model, we adopt the embedding method approach. As discussed regarding the properties of embedding methods in chapter 2, embedding methods can learn similarities among words in an unsupervised way, and they are generally applicable to massive data using online learning. We take the approach similar to the Paragraph Vector models [54], which learn paragraph representations considering word similarities. We use the Paragraph Vector models to produce entity embeddings, and then we extend them to handle category information.

### 3.3.1 Entity vector models

Although the Paragraph Vector models do not consider categories in their model, they can produce entity embeddings. We add little modification on the Paragraph Vector models to deal with entities instead of paragraphs, and we call the modified Paragraph Vector models the "Entity Vector models". First, we describe the algorithms of Entity Vector models, which closely resemble those of Paragraph Vector models (as shown in Figure 3.1). The entity vector with distributed memory (EV-DM) predicts a target word from an entity and words in a context window. To train EV-DM from a word sequence $(w_1, w_2, ..., w_T)$, we maximize the average log probability of a target word ($w_t$) given context words ($w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}$) and the entity ($e_m$):

$$\frac{1}{T}\sum_{t=1}^{T} \log p(w_t|e_m, w_{t-k}, ..., w_{t+k}) \tag{3.6}$$

where $k$ is the window size of the context, and $T$ is the length of a document. The entity vector with distributed bag-of-words (EV-DBOW) predicts words in the window given the entity. For training, we maximize the average log probability of words in a window given the entity:

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-k\leq j\leq k} \log p(w_{t+j}|e_m) \tag{3.7}$$

The entity vector and word vectors are concatenated or averaged in the middle layer, and the inner product of the vector in the middle layer and a vector in the output layer is used to calculate the posterior probability. Both models define the posterior probability in the form of a softmax function:

$$P(w_i|w_j) = \frac{\exp(u_i^\top v_j)}{\sum_k \exp(u_k^\top v_j)} \tag{3.8}$$

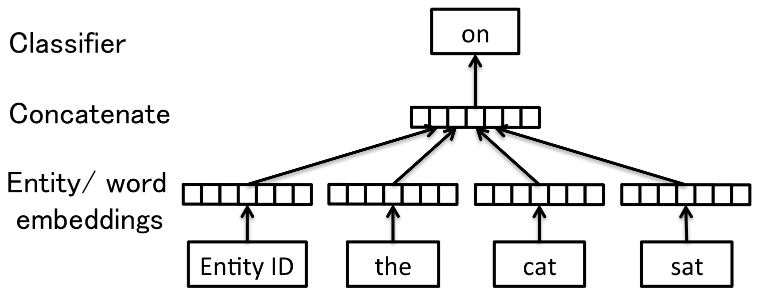
where $v_j$ denotes an input vector of $w_j$, and $u_i$ denotes an output vector of $w_i$. As the entity embeddings appear only in the input layer, we use an input vector of $p_m$ for entity embedding of $m$-th entity.

The difference between Paragraph Vector models and the Entity Vector models here is that an entity can have several texts, while a paragraph corresponds to one set of sentences since we do not assign an entity embedding to each text, but to each entity. Therefore, the same entity vector can appear in different texts in the models. However, the paragraph vector appears only once in a specific text.

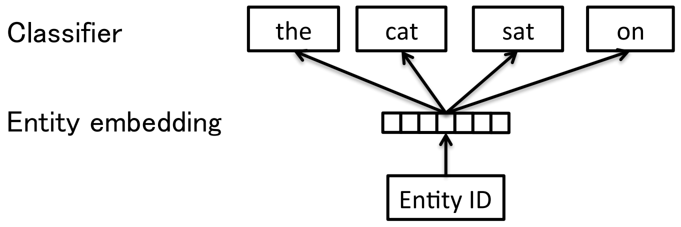
In this way, the entity vector models train entity vectors and word vectors simultaneously. We update parameters using stochastic gradient descent (SGD) in the same way as in the Paragraph Vector models. SGD is one of the typical algorithms for "online" learning techniques because it updates parameters for each instance. Our object function here is the average log likelihood, and we can calculate the gradient vector of this object function ($g$). We apply the update function as follows in iterative way[1]:

$$v_{t+1} = v_t + \eta g(v_t) \tag{3.9}$$

[1]Although we maximize the object function here, normally we minimize the object function. Therefore, we call the algorithm, stochastic gradient *descent*. The update function here is to maximize the object function.



(a) The model of entity vector with distributed memory (EV-DM)



(b) The model of entity vector with distributed bag-of-words (EV-DBOW)

Figure 3.1: The Entity Vector Models.

where $v_t$ is the parameter $v$ at the time of $t$, and $\eta$ is called the "learning rate". The performance of this algorithm generally depends on the learning rate, and SGD sometimes diverges the parameters. Therefore we often decrease the learning rate from $\eta_{max}$ to $\eta_{min}$ throughout the training period to converge the parameters.

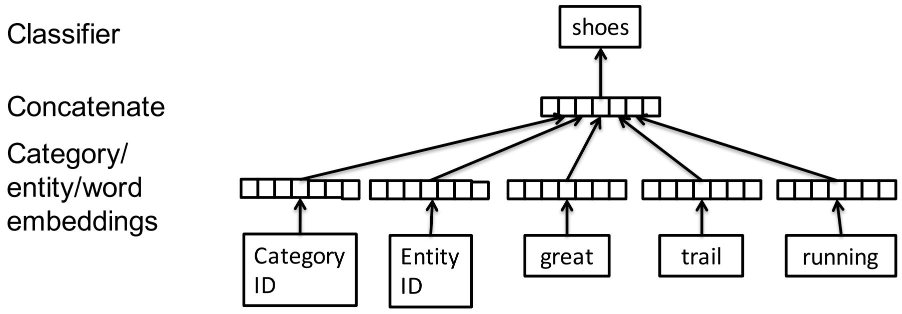
Instead of updating all the output layer parameters, hierarchical softmax (HS) or negative sampling (NEG) is used because the gradient of the softmax function has a large number of terms. We aim to produce good representations for entities and words instead of obtaining the posterior distribution. Therefore, we can take an approximate approach for the output layer.

While we can assume vocabulary as a fixed set, we usually have unseen entities in the test set. To handle unseen entities, these models can train entity embeddings for new ones while fixing word vectors. The entity vector can be used as a fixed-length dense feature vector for tasks such as classification and clustering.

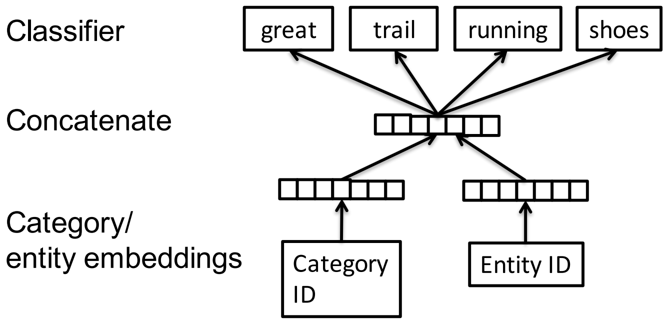
### 3.3.2 Category vector models

We call the category-enhanced embedding method the "Category Vector models". To integrate categories into the embedding method, we follow the approach of the Paragraph Vector models. We model the category vector models as the extra context. As the entity vector plays a role of context for its text items, the embedding for a category, *the category vector*, plays a role of context shared along items in the category. In this way, the category information can help to learn entity embeddings, and the entity information can help to learn category embeddings, simultaneously.

The category vector models consist of two models as in entity vector models: the vector of the category with distributed memory (CV-DM) and the vector of the category with distributed bag of words (CV-DBOW), as shown in Figure 3.2. In both models, each category is mapped to a unique vector.



(a) The model of the category vector with distributed memory (CV-DM)



(b) The model of the category vector with distributed bag of words(CV-DBOW)

Figure 3.2: Category Vector Models.

The CV-DM model predicts a target word from an entity ($e_m$) and a category ($c_n$), combined with words in a context window ($w_{t-k}, ..., w_{t+k}$). The objective of the CV-DM model is to maximize the average log posterior probability:

$$\frac{1}{N}\sum_{i,j}\frac{1}{T_{i,j}}\sum_{t=1}^{T_{i,j}}\log p(w_t|w_{t-k},...,w_{t+k},e_m,c_n) \qquad (3.10)$$

where $N$ is the number of training instances, and $T_{i,j}$ is the word size of paragraphs belonging to the $m$-th entity and $n$-th category. The vectors of a category, an entity, and words in a context window are concatenated or averaged in the middle layer, which is used to calculate the posterior probabilities.

The CV-DBOW model predicts words in a $2k$ word window from an entity ($e_m$) and a category ($c_n$). The objective of the CV-DBOW model is to maximize the average log posterior probability:

$$\frac{1}{N}\sum_{i,j}\frac{1}{T_{i,j}}\sum_{t=1}^{T_{i,j}}\sum_{-k\leq l\leq k}\log p(w_{t+l}|e_m, c_n) \tag{3.11}$$

The vectors of a category and an entity are concatenated in the middle layer, and we take the inner product of the vector in the middle layer and a vector in the output layer to calculate the posterior probability.

To update parameters, we use AdaGrad [24] as well as SGD in update equations. AdaGrad can avoid poor convergence and divergence of frequently updated vectors. The update rule of AdaGrad is as follows[2]:

$$v_{t+1,i} = v_{t,i} + \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \tag{3.12}$$

where $v_t$ is a vector (word, paragraph, category) at the time $t$, and $\sum_{\tau=1}^{t} g_{\tau,i}^2$ is the sum of the the $i$-th dimension of gradients squared in the history. Compared to SGD, AdaGrad decreases the learning rate throughout the training phase since the squared gradient history $\sum_{\tau=1}^{t} g_{\tau,i}^2$ increases as it iterates. In addition, the AdaGrad is more robust against the initial learning rate, since the AdaGrad can adjust the learning rate according to gradients in the past. The drawback is that AdaGrad requires almost twice more the memory size of SGD since AdaGrad stores the history of gradients for each dimension of parameters.

We can also use Adam [48] if the memory size allows. Adam is more robust against the hyper-parameters, such as learning rate, although SGD and AdaGrad depend on the initial learning rate and sometimes require several trials to set the right value. Recent studies use this technique to optimize deep neural networks [97, 35, 18]. Adam estimates the first moment vectors and the second moment vectors as it updates parameters from an instance. The update rule is as follows:

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t \tag{3.13}$$

$$u_{t+1} = \beta_2 u_t + (1-\beta_2) g_t^2 \tag{3.14}$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^t} \tag{3.15}$$

$$\hat{u}_{t+1} = \frac{u_{t+1}}{1-\beta_2^t} \tag{3.16}$$

$$v_{t+1} = v_t + \alpha\frac{\hat{m}_{t+1}}{\sqrt{\hat{u_{t+1}}}+\epsilon} \tag{3.17}$$

[2]Again, we maximize the object function here.

where $m_t$ is the first moment vector at the time of $t$, $u_t$ is the second moment vector, $\hat{m}, \hat{n}$ are the bias-corrected moment estimates, $\alpha$ is the step size, $\beta_1, \beta_2$ are the hyper-parameters for decay rates for the moment estimates, and $\epsilon$ is to avoid the 0-division. The experiment by Kingma and Ba showed that $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ are good hyper-parameter settings for several tasks, and they claimed that these default settings can work well in other tasks [48]. The drawback is that Adam has more hyper-parameters than SGD and AdaGrad, which means it requires more effort to find the best hyper-parameters when it does not work in the default settings, and that Adam requires almost three times more memory size than SGD since Adam stores the first and second moment estimates for each dimension of parameters.

In the same way as the Entity Vector models and the Paragraph Vector models, either HS or NEG is used to approximate computation in the output layer. While the algorithm is similar to word2vec models and the Paragraph Vector models, we change the initialization of input vectors (word, entity, category) as follows:

$$v_i = \frac{r_i}{\sqrt{d}} \tag{3.18}$$

where $v_i$ is the $i$-th element of a vector $v$, $d$ is the dimension, and $r_i$ is a uniform random variable on [-0.5, 0.5]. As Glorot and Bengio pointed out, this initialization technique is heuristic but commonly used [31]. Our preliminary experiments showed that we sometimes have bad convergence (where embeddings agglomerate in some point) using the word2vec's initialization, while this heuristic technique always worked.

### 3.3.3 Inference of categories

We can use the embeddings of categories themselves to see the similarity and dissimilarity among categories; however, it is more useful to infer the category from the new instance. For example, in the e-commerce site, we can reduce the misclassification of products if we can infer the correct category for a new product. In social media analysis, it is rare to have all the network data. The inference can help to classify the users into known communities.

To infer the category of an instance from a trained model, we fix the learned word vectors and train the embeddings of entities and categories. We can only obtain the sum of embeddings of entities and categories since there is one degree of freedom between an embedding of a category and an embedding of an entity. Thus, we use the summed vector of these two vectors for inference of its category. We compute the summed vectors of the category vectors and the paragraph vectors in the training set, and we find the closest vector to the summed vector of the test

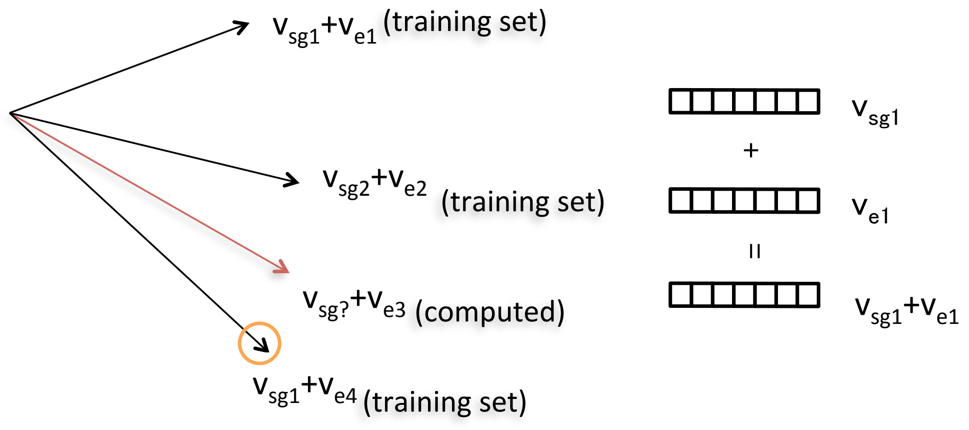

$v_{sg1}+v_{e1}$ (training set)
$v_{sg2}+v_{e2}$ (training set)
$v_{sg?}+v_{e3}$ (computed)
$v_{sg1}+v_{e4}$ (training set)
$v_{sg1}$
+
$v_{e1}$
=
$v_{sg1}+v_{e1}$

Figure 3.3: The summed vectors of category and paragraph vectors. We regard the category of the closest item in the training set as the best candidate for the test instance. Here, we infer that the test instance ($e3$) belongs to the category 1 ($sg1$) because the closest instance ($e4$) belongs to the category 1.

instance from the training set, as in Figure 3.3. We adopt the cosine similarity between normalized vectors to find the most similar item, and we can adopt the category of the most similar item in the training set.

### 3.3.4 Joint model

To obtain better performance, we can combine two models. After we have trained two models separately (e.g., the SGV-DM model and the SGV-DBOW model), we have learned embeddings of words, entities, and categories, which are $d_i$-dimensional vectors for the $i$-th model and $d_j$-dimensional vectors for the $j$-th model. In the joint model, we concatenate two entity/category vectors of dimensions $d_i, d_j$ from two models horizontally. The concatenated vectors are $(d_i + d_j)$-dimensional vectors, as shown in Figure 3.4. As mentioned in the previous section, we use the normalized summed vector of the category vector and the paragraph vector in the training set; therefore, we concatenate two normalized summed vectors from the models. We can concatenate two different models (CV-DM and CV-DBOW models) or the same models with different settings (CV-DM with HS and CV-DM with NEG). For the test instance, after we have trained a summed vector using each model, we concatenate two of the normalized vectors. We find the most similar item in the concatenated vectors in the training set to infer the category of the test instance.

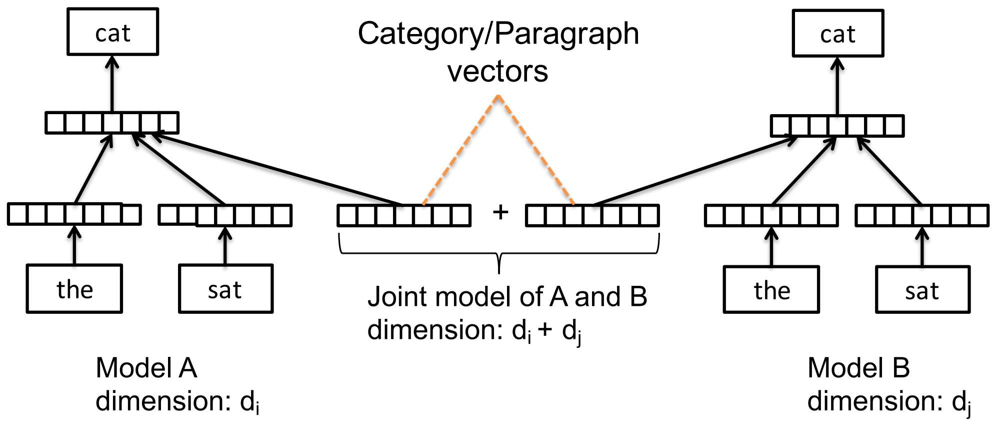


Figure 3.4: Joint model of two models. These models are trained separately, and the joint model combines the embeddings of an entity and/or a category.

### 3.3.5 Multilingual model

The embedding methods can learn the relationships among the words in different languages from multilingual corpus. Several studies have been conducted on models of multilingual word representations. These models can benefit from monolingual corpus, which is easy to obtain, and use a small multilingual dictionary or parallel corpus to constrain the distributed representations in different languages. Mikolov et al. trained distributed representations for words in each language and then calculated the linear transformation matrix of vectors from one language to another using parallel data [63]. They achieved good precision for translation of words between English and Spanish. Klementiev et al. formulated the acquisition of multilingual distributed representations as a multi-task learning problem, which requires co-occurrence statistics from parallel data [49]. Hermann and Blunsom proposed a bilingually aligned word representation model [39], and Pham et al. extended the paragraph vector models to multilingual models [75], which require sentence alignment. These three models are reported to improve document classification tasks on crosslingual documents. However, it is usually very rare to obtain word alignment or sentence alignment in massive dataset, which the aforementioned methods require.

Compared to obtaining word alignment or sentence alignment, it is relatively easy to obtain the category alignment or topic alignment. For example, some global e-commerce companies have marketplaces in several countries with different product taxonomies. Rakuten, which is one of the largest shopping sites in Japan, is such a company, and they have e-commerce sites in Japan, North America, Europe, Southeast Asia, and Taiwan. They have rich taxonomy in Japanese and

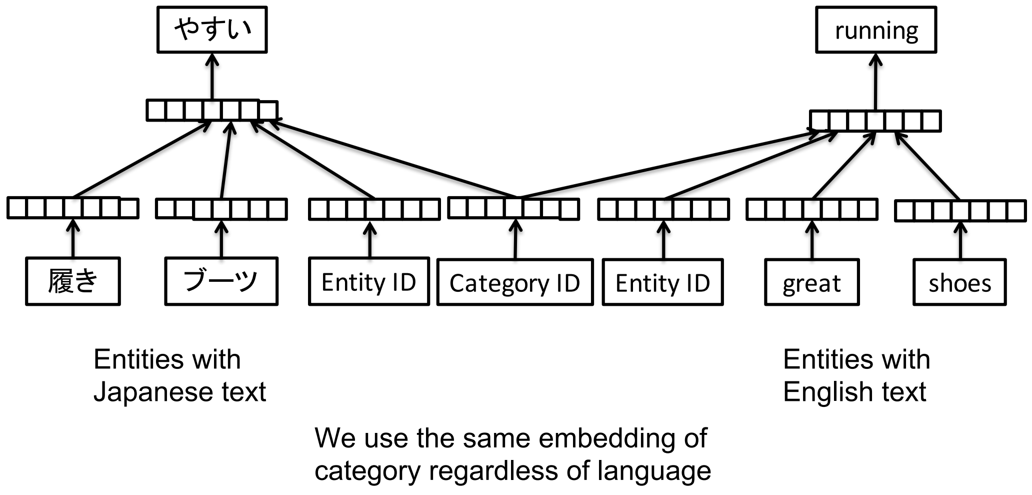

やすい
running
履き
ブーツ
Entity ID
Category ID
Entity ID
great
shoes
Entities with Japanese text
Entities with English text
We use the same embedding of category regardless of language

Figure 3.5: Multilingual model for Japanese and English texts. We have the alignment among categories, and we train the embeddings of words / entities / categories in both languages simultaneously.

English since the number of products is large, while the new marketplaces have less rich taxonomy. They have category alignment among marketplaces in different languages, but they do not have product alignment for most of the products. The embedding method can help to capture the similarity or dissimilarity among categories from multiple language sources, and therefore the rich taxonomy and products in Japanese or English can help to classify products in other language domains. Even if there is difficulty to obtain the product correspondence in the multilingual data, we can sometimes obtain universal categories or an interlingual correspondence table among categories or topics since categories are often a fixed set. This means that the quality of embeddings in a language can be improved by adding additional training data in another language, even though there is no word-level or sentence-level correspondence between the two languages.

To train the model from multiple languages, we assign word embeddings and entity embeddings separately according to the language or the site, but we assign the same embeddings to corresponding categories in multiple languages. In this way, we can train the model giving the training set merged multiple languages without translating the contents. We use the same algorithm for training the model and inference of categories. We can also merge multilingual models trained with different settings.

### 3.3.6 Implementation

We implemented the Entity Vector models and the Category Vector models described previously in Python and C++. Our implementation partially uses word2vec implementation of "gensim" written by Řehůřek [78], and utilizes Cython and BLAS (Basic Linear Algebra Subprograms) library. We published the implementation as an open source software (GNU LGPL license) and the code is available at GitHub: `https://github.com/rakuten-nlp/category2vec`.

## 3.4 Analysis of the relationship between word usage and categories

We modeled the embeddings of a category, and an entity is learned as a context of its text data. Therefore, our model captures not only the sequence of surface representations, but also local contexts of words that can be characteristic in certain categories. Now, how do we confirm that the model learned the difference on the local contexts of words (i.e. word usage) among categories, particularly implicit categories? We propose here the methods for capturing the differences of word usage among categories to deepen an insight into the relationship among categories.

### 3.4.1 Overview of the analysis framework

Next, we explain the overview of our framework to analyze how word usage is different by categories. For the analysis, we exploit the Skip-gram model (one of the word2vec models) to capture the local context of words in each community. After training the Skip-gram model, each word obtains a word vector as the word embedding (or distributed representations).

Our framework is divided into three main parts.

1. First, we train the Skip-gram model using the whole text, or sampled text data from the whole data. As the initial state generally affects the embeddings through training, we compute the word embeddings from the whole or sampled dataset, and we use these as "pre-trained" data in the next step.

2. Using the pre-trained data, we retrain the Skip-gram model from the text data of each category. Since the word embeddings in a category are expected to reflect how the word is used in the categories, they provide us with ways to explore the differences in the use of language among the groups. We obtain the set of word embeddings for each category.

3. Using the trained word embeddings, we analyze the differences in the use of language among categories. If entities use a word in a similar way in different categories, the word embeddings among groups become similar, and if entities use a word in different way, the word embeddings among groups become dissimilar. We compute a similarity measure among categories.

### 3.4.2 Training word embeddings in each category

Here, to grasp how words are used in each category, we train word embeddings in each group. As mentioned previously, each word embedding is expressed as a word vector, which reflects the context in which the word appears. This feature can be used to analyze how the word is used differently among categories.

First, we sample $d\%$ ($d$: arbitrary constant) from the whole entities to create unbiased corpus by categories. Second, we train the Skip-gram model giving the unbiased corpus, and we obtain word embeddings. Third, we sample $c/N_i\%$ text data ($N_i$:entities in the $i$-th category, $c$:constant) from each entity group, and create a balanced corpus for each category. The reason we balance the number of entities in categories is to avoid the influence of content length. If we input larger corpus, then we update the embeddings more times, which can make the embeddings go away from the pre-trained embeddings. Finally, we use previously learned word embeddings as pre-trained embeddings, and we retrain word embeddings giving text data sampled from a category. Here we obtain $M$ word embeddings for a single word ($M$: the number of categories). We use the word embeddings to explore the differences of word usage among categories.

As to the implementation, we use "word2vec" [3], the original implementation by Mikolov et al. [64], and modify the code for retraining. The Skip-gram model simplifies the neural network and adopts surrounding words as a context instead of adopting several preceding words for the target word, as shown in Fig. 3.6. The posterior probability of the Skip-gram model is shown as below:

$$p(w_{t+j}|w_t) = \frac{\exp(v_{w_{t+j}}^{\mathrm{T}} \cdot v_{w_t})}{\sum_k \exp(v_{w_k}^{\mathrm{T}} \cdot v_{w_t})} \tag{3.19}$$

where $v_{w_k}$ represents the word embedding for the word $w_k$, $w_t$ is a target word, and$w_{t+j}$ is a context word ($j \neq 0$). The posterior is expressed as a softmax function, and the number of terms of its gradient increases linearly to the vocabulary size; therefore, the training time becomes considerably long. Mikolov et al. proposed HS and NEG to reduce calculation time in the same way as done with the Entity Vector models and the Category Vector models.

[3]The original code is available at `https://code.google.com/p/word2vec/`

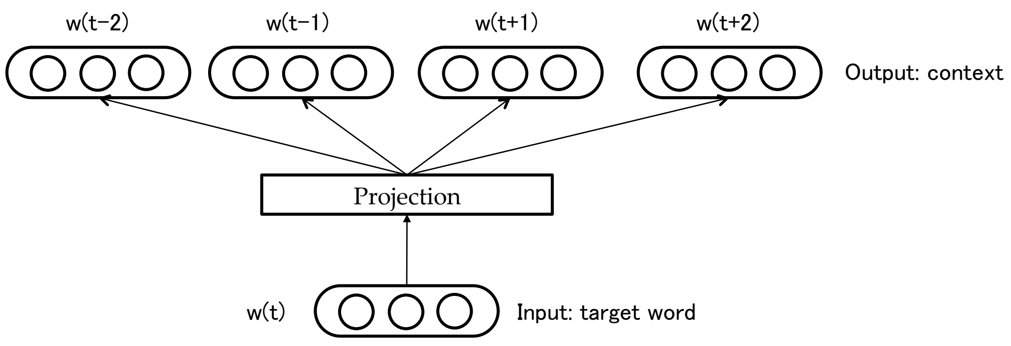


Figure 3.6: The Skip-gram model learns word vectors to maximize the posterior.

Trained word embeddings have the following characteristics: words having similar contexts are projected to similar vectors in the vector space. Still, we cannot completely solve the problem of polysemies and homonyms in adopting the embedding methods. However, we focus on the fact that word embeddings represent the context where the words are mainly used, and that similar word embeddings indicate that the words are often used in the same way.

### 3.4.3 Analysis of the differences in the use of language among categories

#### (1) Detecting ambiguous words among categories

First, we address the following question: how is the same word is used in different senses among categories? In this part we use the embedding method that can capture contexts for each word, which enables us to grasp the set of words that potentially co-occurs with the word. Thus, by seeing how the contexts for the word are different among categories, we can expect to detect ambiguous words among categories — that is, words that are used in different local contexts depending on categories. Please note, the term *ambiguous words* among categories does not mean that the term actually has several meanings, but that the term is used in several contexts among categories, which can mean that the term is likely to have multiple meanings. We do not step into the definition of the meanings and the methodologies to extract meanings. Here we only discuss the difference in local contexts since the local contexts characterize the categories.

To obtain prospective words that used in several local contexts, we calculate how similar a word is between a pair of categories. More specifically, for each word and each pair of categories, we calculate cosine similarity between the word embedding of one category and the word embedding of the other. Then we try to detect ambiguous words from highly dissimilar words by using the context as clues: that is, the set of words that potentially co-occurs with the word.

### (2) Word-usage-based similarity among categories

To determine whether categories have similar word usage, we first define the similarities based on the word usage. We will describe them in the following manner.

**Definition of similarity between categories based on word usage:** We define word-usage-based similarity between the category $i$ and the category $j$, $Sim_{word}(i, j)$, as follows.

$$Sim_{word}(i, j) = \frac{1}{N} \sum_{a=1}^{N} \frac{|S_{ia} \wedge S_{ja}|}{|S_{ia} \vee S_{ja}|} \tag{3.20}$$

where $a$ is an index of word, $w_a$ is a word, $v_{ia}$ is a distributed representation for a word $w_a$ in the category $i$, "$\cdot$" means inner product, and $S_{ia} = \{w_b | \text{top 30 of } v_{ia} \cdot v_{ib}\}$. $N$ is the number of "target words" for community $i$ and $j$. Target words are taken from words in the whole tweets that appear at least 100 times both in tweets in categories $i$ and $j$.

Since word embeddings are trained differently in categories, $w_a$ has different representations according to categories. $S_{ia}$ means a set of similar words to $w_a$ used in the similar local contexts in the category $i$, and $Sim_{word}(i, j)$ is the average of Jaccard similarity between the sets of similar words to a target word in the categories $i$ and $j$.

We can compare this similarity score to the learned categories to explore whether the close categories in their attribute or their network property are also close in the similarity measure.

In the next chapter, we apply the models and the analysis framework mentioned above to explicit and implicit categories. For explicit categories, we use e-commerce data, in which thousands of categories are manually tagged. We show the effectiveness of the category-enhanced models in inferring the category of items from their text descriptions, which helps with manual category tagging of products. For implicit categories, we use social media data, through which users are connected with networks. We also apply the analysis framework for category-enhanced models to see how we can extract the characteristics of implicit categories with embedding methods.

Chapter 4

# Embedding Method for Categories

We often have labels or categories for each item of documents. Such labels and categories are regarded as explicit categories, as discussed in chapter 3. Among datasets having explicit categories, e-commerce sites have large number of categories in which each product belongs (e.g., Electronics, Apparel, and Furniture & Interior). Items that belong to the same category have similar descriptions; therefore, we can use BOW or Entity Vector models to infer correct categories, taking their similarities. However, it is more useful if we have the document representations that contain category information in addition to item information. This is the case particularly for e-commerce sites, which update product information periodically and sometimes label products with incorrect categories. However, if we can generate embeddings of category and words (both of which can be considered as fixed sets), we do not need to store all the data records. For this purpose, we use the Category Vector models for classification of products and we show the effectiveness by comparing the Category Vector models to the BOW and Entity Vector models on the task where we classify the products into thousands of categories.

## 4.1 Dataset

Rakuten Ichiba[1] is the largest e-commerce site in Japan, and Rakuten.com Shopping[2] is an online marketplace in the U.S. We use 2.5M items from Rakuten Ichiba (Japanese) and 3.0M items from Rakuten.com Shopping (English). We call them R-JA data and R-EN data, respectively. We use 25,705 items from R-

[1]`http://www.rakuten.co.jp`

[2]`http://www.rakuten.com`

JA data and 23,877 items from R-EN data as the test data and the rest as the training data. Each item consists of a title, a description, and a category, e.g., Books/Architecture, or Movies & TV/Comedy. We concatenate the title and the description for each product, and we use it as an input. For Japanese data, we use MeCab[3] and UniDic[4] to tokenize.

Categories are constructed in a tree structure, and each category except the top-level category has a parent category, as shown in Figure 4.1. There are 21,764 leaf categories in the training set of R-JA, and 8,245 leaf categories in the training set of R-EN. We manually created a correspondence table where leaf categories of R-JA correspond to those of R-EN categories. Note that a leaf category in one language can correspond to multiple leaf categories in another language. For example, the category "Hard Disk Drives/Internal" in English data corresponds to two categories ("HDD/2.5-inch" and "HDD/3.5-inch") in Japanese data as shown in Figure 4.2. We convert R-JA categories to R-EN categories using the correspondence table since R-JA has more categories than R-EN. We concatenate the training set of R-EN and the training set of R-JA with R-EN categories (we call this R-MIX data). R-MIX data have 5.5M items with 8,783 categories. No overlap occurs between Japanese word/product IDs and English word/product IDs.

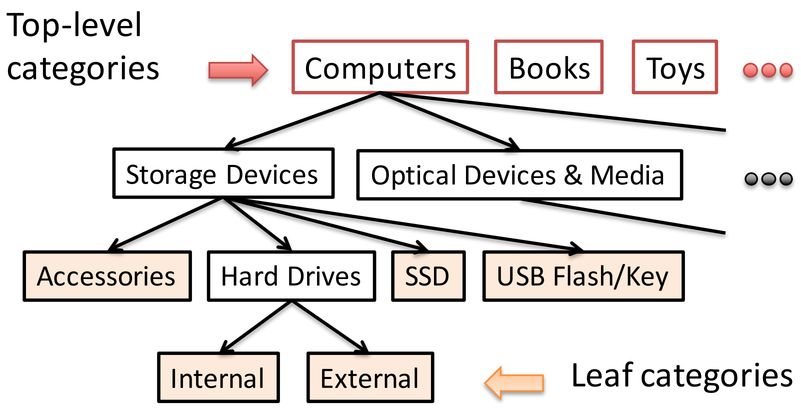


Figure 4.1: Categories in a tree structure. Categories with red frames are top-level, and categories filled with orange color are leaf categories, where items belong.

## 4.2 Experiment

Here we call the Entity Vector models "Product Vector" models, which contain the PrV-DM model (from the EV-DM model) and the PrV-DBOW model (from

[3]`https://code.google.com/p/mecab/`
[4]`http://sourceforge.jp/projects/unidic/`

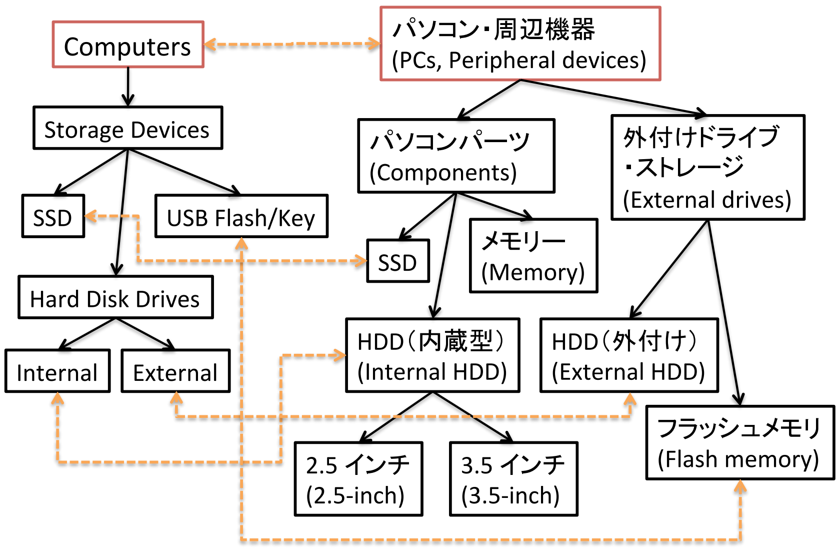

Computers
パソコン・周辺機器 (PCs, Peripheral devices)
Storage Devices
SSD
USB Flash/Key
Hard Disk Drives
Internal
External
パソコンパーツ (Components)
外付けドライブ・ストレージ (External drives)
SSD
メモリー (Memory)
HDD（内蔵型） (Internal HDD)
HDD（外付け） (External HDD)
2.5 インチ (2.5-inch)
3.5 インチ (3.5-inch)
フラッシュメモリ (Flash memory)

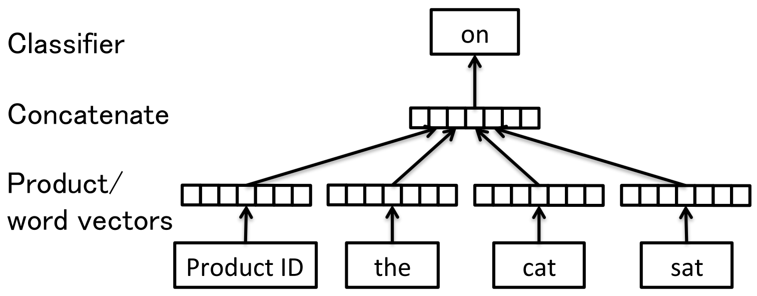
Figure 4.2: Category trees in English and Japanese. A dashed orange line denotes the correspondence between two languages.

the EV-DBOW model), and we use Category Vector models, where we use Product ID as Entity ID. To confirm that the Category Vector models learn category information, we compare the BOW model, the Product Vector models and the Category Vector models by inferring a category of a product from its description. We also compare the joint model of Product Vector models, the joint model of the Category Vector models, and the Category Vector models trained from the multilingual corpus. For this purpose, we use Rakuten Ichiba data and Rakuten.com Shopping data described in section 4.1.

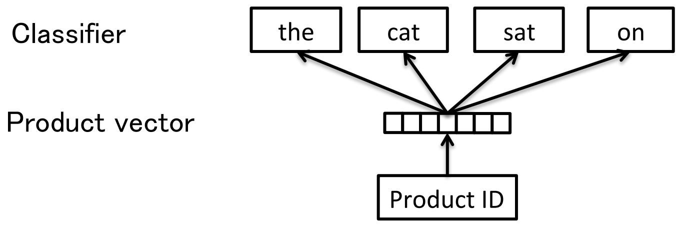
### 4.2.1 Task

In this section we will describe the details of the task. We measure the quality of the category and product embeddings by comparing the accuracy of the inference task, where each model infers a true category from its embeddings of products.

For the BOW model, we computed TF-IDF for each word in the training set and obtained TF-IDF vectors in the test set using IDF of the training set. To obtain embeddings for products, first we trained the Product/Category vector models until the algorithm iterated through the training data for specific times. We trained the Product/Category vector models, exploring several values of hyperparameters enumerated in Table 4.1. Preliminary experiments showed that the dimension of 300 and the iteration of 30 times is sufficient to saturate the accuracy, and that the 5 negative samples perform better than 15 negative samples, despite the study of Levy et al., which showed that more negative samples for

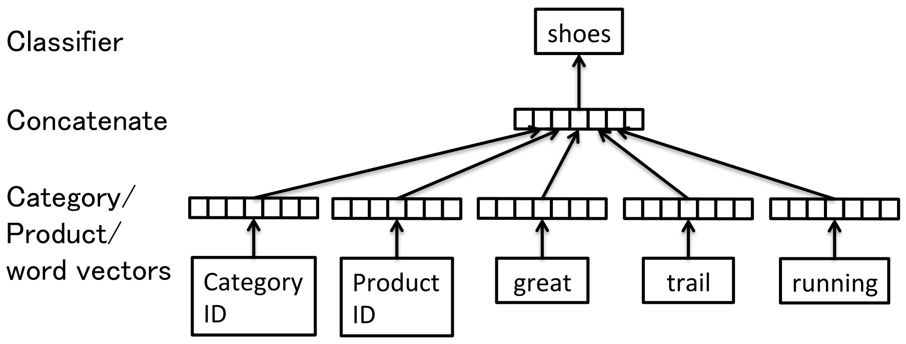


(a) PrV-DM model

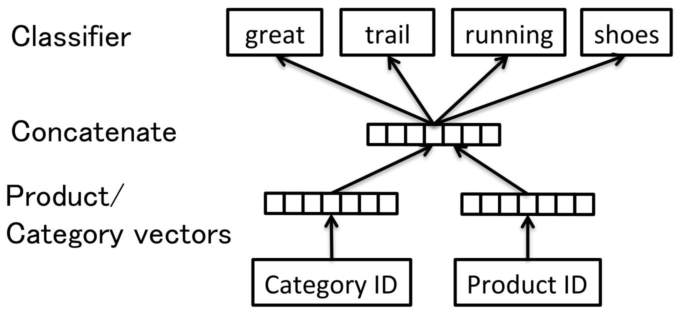


(b) PrV-DBOW model

Figure 4.3: Product Vector Models



(a) CV-DM model



(b) CV-DBOW model

Figure 4.4: Category Vector Models.

word embeddings improve analogy tasks [57]. Second we fixed word embeddings from the trained model and use them to calculate a product embedding in Prod-

Table 4.1: Parameters explored in this task. PrV denotes Product Vector, and CV denotes Category Vector. The iteration, dimension, and #negative samples are explored only in preliminary experiments.

| Hyper-parameter | Explored Values / Settings | Methods |
|---|---|---|
| model | DM, DBOW | PrV, CV |
| output layer | HS, NEG | PrV, CV |
| update rule | SGD, AdaGrad | CV |
| dimension | 200, 300 | PrV, CV |
| #negative samples | 5, 15 | PrV, CV |
| alpha / $\eta$ | 0.001, 0.005, 0.025, 0.05, 0.1, 0.15, 0.2 | PrV, CV |
| iterations | 10, 15, 30, 35 | PrV, CV |
| window size | 5 | PrV, CV |
| subsampling | none | PrV, CV |

uct Vector models or the sum of a category embedding and a product embedding in the Category Vector models from each product in test data. In all the cases, we computed cosine similarity between a embedding in the test set and learned embeddings in the training set, and extracted the most similar item in the training set to infer its category. We regarded the category of the most similar item as the best category candidate of the test instance. Finally, we calculated the accuracy of this inference task both in R-JA and R-EN data, using the BOW model, and the Product Vector models and the Category Vector models.

In addition to seeing how the BOW model and the Product/Category Vector models perform, we trained the Category Vector models using R-MIX data and the same learning rate as R-EN data, and saw how the multilingual category vector models improved the accuracy. Since the R-MIX dataset is aligned with R-EN categories, we used the test set of R-EN data and compared it with the result of R-EN.

As to joint models, we also combined two models from Product Vector models in different conditions (PrV-DM/PrV-DBOW, HS/NEG), or Category Vector models in different conditions (CV-DM/CV-DBOW, HS/NEG, SGD/AdaGrad). We trained the vector of the test instance in each model separately, and we concatenated these vectors horizontally from two models for test.

We used the implementation described in subsection 3.3.6 to train the Product Vector models and the Category Vector models.

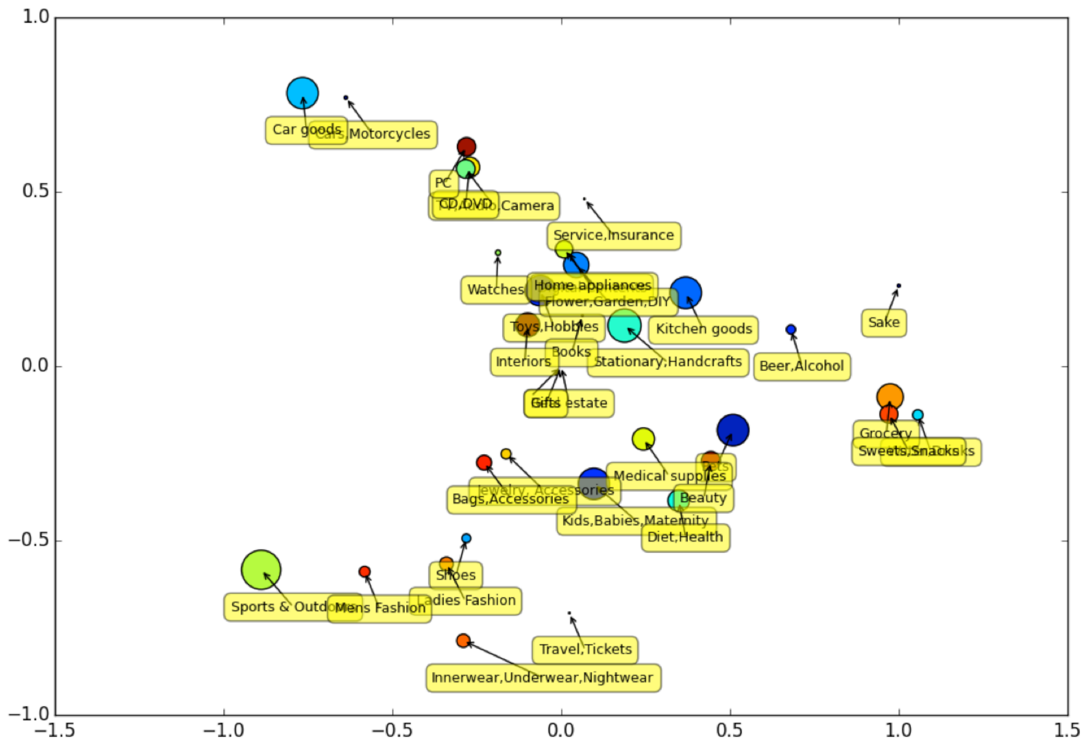


Figure 4.5: Centroids of category embeddings in R-JA. Each point denotes the centroid of a top-level category.

### 4.2.2 Results

First we observe the learned category embeddings. As we have a large number of leaf categories, we calculated the centroid vectors of top-level categories (as shown in Figure 4.1) from the category embeddings for simplicity. We reduced the dimension of centroids from 300 to 2 to plot in the planar space using principal component analysis (PCA). The centroids are shown in Figure 4.5, where each point denotes a top-level category and the size of a point denotes the size of its leaf categories. We can see the tendency that the similar categories are located in the similar space, as the categories “Grocery” and “Sweets, Snacks” are close to each other, and the categories “Sports & Outdoors”, “Mens Fashion”, “Ladies Fashion”, “Shoes”, and “Inner-wear, Underwear, Nightwear” are also close to each other.

As to the inference task, we show the results in Table 4.2 (R-JA data), Table 4.3 (R-EN data), and Table 4.4(R-MIX data). We showed the best learning rate (including AdaGrad’s $\eta$) for each model and output layer. For joint models, we list only the best combinations.

Table 4.2: Category inference task in R-JA data

| Model | Output Layer | |
|---|---|---|
| | HS | NEG |
| BOW | 47.70% | |
| PV-DM SGD | 38.04% | 19.84% |
| PV-DBOW SGD | 49.03% | 48.27% |
| CV-DM SGD | 46.61% | 7.15% |
| CV-DBOW SGD | 52.15% | 48.89% |
| CV-DM AdaGrad | 51.35% | 46.65% |
| CV-DBOW AdaGrad | 55.23% | **56.02%** |

| Joint Model | |
|---|---|
| PV-DBOW SGD HS + PV-DBOW SGD NEG | 49.71% |
| CV-DBOW SGD HS + CV-DBOW SGD NEG | 53.97% |
| CV-DBOW AdaGrad HS + CV-DBOW AdaGrad NEG | **57.14%** |

Table 4.3: Category inference task in R-EN data

| Model | Output Layer | |
|---|---|---|
| | HS | NEG |
| BOW | 59.42% | |
| PrV-DM SGD | 49.65% | 31.14% |
| PrV-DBOW SGD | 59.20% | **59.47%** |
| CV-DM SGD | 47.74% | 7.55% |
| CV-DBOW SGD | 42.39% | 36.33% |
| CV-DM AdaGrad | 59.14% | 55.26% |
| CV-DBOW AdaGrad | 55.82% | 57.11% |

| Joint Model | |
|---|---|
| PrV-DBOW SGD HS + PrV-DBOW SGD NEG | 60.99% |
| CV-DM SGD HS + CV-DBOW SGD HS | 49.21% |
| CV-DM AdaGrad HS + CV-DM AdaGrad NEG | **61.02%** |

In R-JA data, the Category Vector models outperform the Product Vector models and the BOW model. The CV-DBOW model with AdaGrad and NEG

Table 4.4: Category inference task in R-MIX data

| Model | Output Layer | |
|---|---|---|
| | HS | NEG |
| CV-DM SGD | 58.19% | 1.23% |
| CV-DBOW SGD | 58.72% | 50.61% |
| CV-DM AdaGrad | 61.30% | 58.63% |
| CV-DBOW AdaGrad | 59.89% | **61.58%** |

| Joint Model | |
|---|---|
| CV-DM SGD HS + CV-DBOW SGD HS | 62.55% |
| CV-DM AdaGrad HS + CV-DBOW AdaGrad NEG | **63.91%** |

performs the best among the Category Vector models, and among joint models, the joint model of the CV-DBOW model with AdaGrad and HS, and the same model with AdaGrad and NEG performs the best. Joint models improve the accuracy compared to a single model, and we can find the true category out of 21K categories with the accuracy of 57.14% with the joint Category Vector model.

In R-EN data, on the other hand, one of the Product Vector models (PrV-DBOW) outperforms the Category Vector models and the BOW model. Among the Category Vector models, the CV-DM model with AdaGrad and HS performs the best. However, among the joint models, the Category Vector models perform better than the BOW model and slightly better than the Product Vector models. We can find the true category out of 8K categories with the accuracy of 61.02% in R-EN data if we use the joint model of the CV-DM model with AdaGrad and HS and the same model with AdaGrad and NEG.

As to multilingual models, we can see that the Category Vector models trained with multilingual data (R-MIX) outperform the BOW model, the Product Vector models, and the Category Vector models in R-EN data. Most of the multilingual models in R-MIX data outperform the same models in R-EN data except for the CV-DM model with SGD and NEG. Among the Category Vector models in R-MIX, the joint model of the CV-DM model with AdaGrad and HS and the CV-DBOW model with AdaGrad and NEG performed the best, with the accuracy of 63.91%.

To see the effect of the enhancement of adding R-JA to R-EN, we took the difference of accuracy between R-MIX and R-EN for each category. Here we used the best joint models, which are CV-DM AdaGrad HS + CV-DM AdaGrad NEG (R-EN) and CV-DM AdaGrad HS + CV-DBOW AdaGrad NEG (R-MIX). We listed the categories in Table 4.5, where the accuracy difference is high and the items in

Table 4.5: Accuracy difference of each category

| **Category** | $Acc_{R\_MIX}$ $-Acc_{R\_EN}$ | **#items in R-EN** | **#items in R-JA** |
|---|---:|---:|---:|
| Computers/Computer Accessories/ Batteries & Adapters/Batteries | 31.3% | 104,538 | 217 |
| Toys/Vehicles/Trains & Train Sets | 28.6% | 1,179 | 0 |
| Home & Outdoor/Furniture, Décor & Storage/Bedding & Linens/Bedskirts | 23.5% | 1,890 | 0 |
| Toys/Dolls & Stuffed Animals/ Dolls | 18.2% | 1,016 | 0 |
| Consumer Electronics/Personal Electronics/ Portable GPS/GPS Accessories | 18.2% | 1,352 | 1,081 |
| Consumer Electronics/Personal Electronics/ Cell Phones & Accessories/Data Connectivity | -23.1% | 948 | 0 |
| Books/Science/ Earth Sciences · Geography | -21.4% | 6,314 | 7,338 |
| Books/Fiction/ Fantasy · General | -17.6% | 2,266 | 0 |
| Home & Outdoor/Furniture, Décor & Storage/Decor & Artwork/Torches & Lights | -16.7% | 3,034 | 879 |
| Books/Juvenile Fiction/ Science Fiction | -16.7% | 1,417 | 0 |

the test set are greater than 10. We also listed the number of items in the training set from R-EN and R-JA. The multilingual model infers the true category better than the monolingual model, for the categories of "Computers/Computer Accessories/Batteries & Adapters/Batteries", "Toys/Vehicles/Trains & Train Sets" etc. The category "Consumer Electronics/Personal Electronics/Portable GPS/GPS Accessories" contains about the same amount of R-EN items and R-JA items as R-MIX. The Japanese descriptions of GPS Accessories contain manufacturer names (e.g., Panasonic, Pioneer) and some specifications (e.g., display size, display resolution), which can improve the shared category vector. Though 4 categories where R-MIX improves the model contain few or no R-JA items, R-JA items can move nearby categories away. For the categories "Consumer Electronics/Personal Electronics/Cell Phones & Accessories/Data Connectivity" and "Books/Science/Earth Sciences · Geography" etc., the multilingual model infers the true category worse than the monolingual model. R-JA disturbs the category vector of "Books/Science/Earth Sciences・Geography", and it can be the cause of R-JA having many books specializing in Japanese geography in the category, and books in R-EN specializing more in whole earth geography. Again, three categories have no R-JA items in R-MIX. There are three book categories, which can suggest that the descriptions are more descriptive than other categories (e.g., electronic devices), and multilingual models cannot assist in learning better category embeddings without any word alignment. In addition, the correspondence between R-JA categories and R-EN categories can be problematic in some cases. However, the overall accuracy increased by 2.89%, which indicates that the multilingual data without word/product alignment can help the Category Vector model to learn the better category representations directly and indirectly.

## 4.3 Discussion

Our experiments on product data in e-commerce sites show that the Category Vector models perform better than the BOW model and the Product Vector models in Japanese data, and the Category Vector models perform slightly worse than the Product Vector models in English data. The joint Product/Category Vector models outperform single Product/Category models, and the joint model of Category Vector models with AdaGrad performs slightly better than the joint Product Vector models. Our experiment on multilingual data shows that the Category Vector models can gain performance from the different language source, which indicates the embedding models with categories can train better entity embeddings enhanced by multilingual data.

The reason the Product Vector models can outperform the Category Vector models in R-EN data while the Category Vector models perform better in R-JA can come from the difference in the distribution of the word frequency. We plotted the word frequency in R-JA and R-EN in Figure 4.6, which reveals that the frequency in R-EN drops faster than that in R-JA. It indicates that models trained from R-JA need to have good word embeddings more than models trained from R-EN, and it reveals that models of R-JA have more parameters to be well-trained. The Category Vector models have more parameters to be updated (category embeddings), and the number of parameters leads to the difficulty in optimization. Therefore, models of R-EN can be affected more from the increase of parameters.

When we compare the models in R-JA data and R-EN data, we can see that DBOW models outperform DM models in R-JA data both in the Paragraph Vector models and the Category Vector models, while the DM models perform about the same or better than DBOW models in Category Vector models in R-EN data.

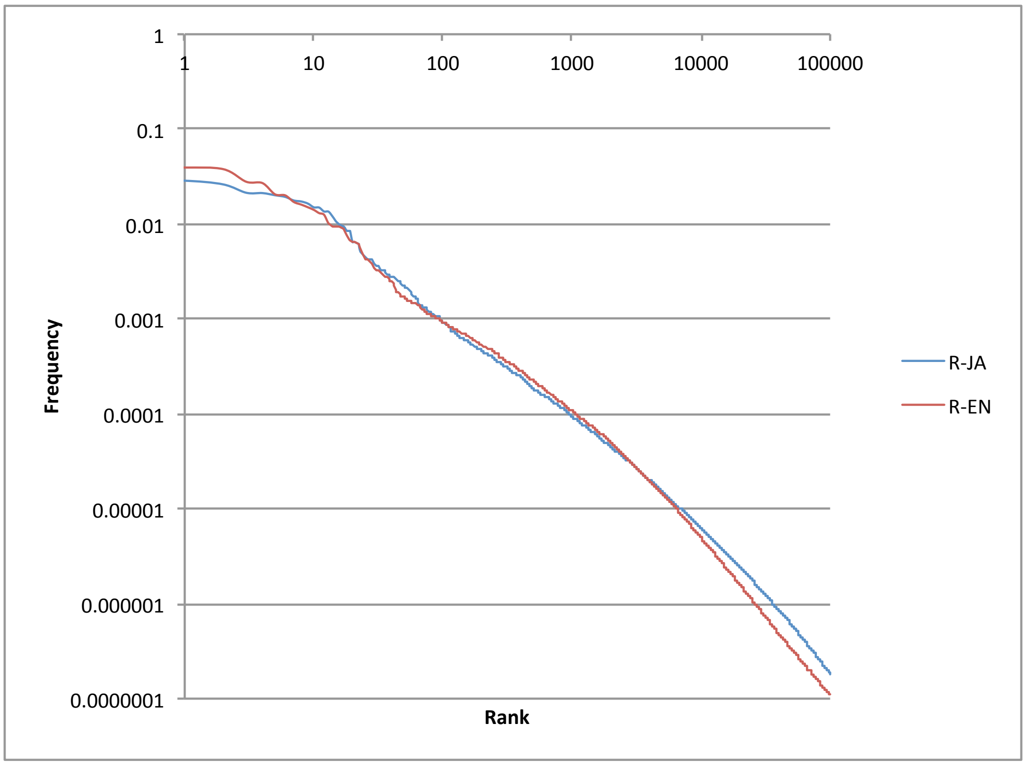


Figure 4.6: The word frequency and the rank in R-JA and R-EN on a log–log scale (rank 1–100000).

This difference may come from the average length of the content. R-JA data have shorter descriptions (36.4 words) compared to R-EN data (65.5 words). The DM models predict the single target word in each step, while the DBOW models predict the word distribution consistent with the description. Therefore, if the description is longer, the DBOW models can perform worse.

# Chapter 5

# Embedding Method for Communities

In the previous chapter, we applied the category-enhanced embedding method to datasets with explicit categories. In this chapter, we apply the same method to datasets with implicit categories. Among such datasets, the dataset of social media has interesting aspects since social media contain much information of users and their contents, despite their frequent lack of structured information. The feature applies to user-generated contents, and among them social media have massive data, which are used for marketing and advertising.

The popularization of social media enables large-scale research for exploring the relationship between friendship and similarity. Extensive research has been conducted across fields relating to "homophily", which is a tendency that people who are connected have to be similar to each other [62]. For example, Romero et al. [81] studied the interplay between social network and the topics they speak about — similarity of the topics in which they are interested. This result indicates that close relationships, "strong ties", tend to occur between similar people, while acquaintances, "weak ties", can give people opportunities to have access to new information, which agrees with the discussion in *The Strength of Weak Ties* by Granovetter [34]. However the question — *are they similar because they are connected* or *are they connected because they are similar?* — remains unanswered [86].

Investigating the relationship between friendship and similarity has the potential to enhance Web-related applications such as targeted advertising and viral marketing in online social network. As Lim et al. pointed out [58], one of the important problems in targeted advertising and viral marketing in online social network is to identify the adequate target audience: that is, users of the adequate demographics who are also highly *connected* among themselves. Detecting users with the right demographics enables the right product-audience matching. At the same time, the connection among people can facilitate word-of-mouth advertising. Thus, investigating the relation between connection and features, such as interests and word usage, which people exhibit can provide insights for the problem. For this purpose, we extract communities from a social network and explore how we can model communities and users in our embedding models.

Among social media, Twitter[1] is used for various research purposes since the data are available from the API. Twitter is a micro-blogging service, and it is primarily used as a social networking service (SNS), where users can post 140-character contents ("tweets"), and users can re-post others' entries ("retweet" or "RT") or message to specific users ("mention"). Each user has a *wall* that displays contents written by their following users. Following is unilateral activity; therefore, it is possible that the user A followed by the user B does not follow the user B. A characteristic of Twitter is that most of the users post personal entries or messages in public; thus, Twitter has been used for a large number of studies in the field of social network analysis recently.

We focus on a conversation network since the tie among users who had conversations is stronger than the following relationship, and it is suitable to extract communities, where users are assumed to be similar and relatively densely connected with each other. We extract the communities from the conversation network, and we assess how well the embedding model can grasp the relationship among communities, users, and their contents. In this chapter we show the effectiveness of the Category Vector models over the BOW model and the Entity Vector models through the inference task, in which each model classifies users into their communities.

## 5.1 Dataset

We collected tweets and profiles with time stamps ranging from January 1, 2012 to December 31, 2012, from 7.4 million users, who were detected tweeting in Japanese by the Twitter API [2]. From this group of data, which contains 4.9 billion tweets, we extracted mentioning tweets to collect mention connections. We constructed a graph in which an edge is created for two users (nodes) if there are bi-directional (mutual) mentions [3]. This period contains 404 million links, of which 125 million

[1]`https://twitter.com`

[2]`https://dev.twitter.com/docs/api`

[3]It is possible to weight the edge based on the number of mentions, but in this research, we consider unweighted graph for simplicity.

links are mutual links (i.e., both users mentioned to each other at least once).

## 5.2 Community extraction and labelling

Before we conduct the inference task, we extract communities from the conversation network. To see the characteristics of the extracted communities, we collected users' biographies and place a label for each community from the keywords.

### 5.2.1 Community extraction from conversation network

The size of the conversation network is large; therefore, we use the Louvain method [6] for community extraction, which is one of the modularity based methods and is applicable to a large graph, as discussed in chapter 2. We used publicly available code [4].

Applying the Louvain method [6], we obtained 34,835 communities. To explore the characteristics of communities well, we need various users in a community. Therefore, we target only the relatively large communities for the rest of this analysis. We selected communities including more than 10,000 users, which account for 97.7% of whole users in the dataset. As a result, the total number of communities that we targeted for the analysis became 38.

### 5.2.2 Labeling community from user profile information

For each community, we extracted users' profile texts, which users can set by the profile function on Twitter. Then we calculated the TF-IDF (term frequency, inverse document frequency) score [85], which is one of the most general numerical statistics for keyword extraction, to calculate the importance of words in each community and extract keywords for each community. We extracted the top 20 words for each community and used them as they represent the characteristics of a community. Then we labeled each community to characterize them by using these keywords as clues for labeling.

We extract the keywords of each community and label the communities as follows:

[4]`https://sites.google.com/site/findcommunities/`

[5]HS stands for "high school".

[6]A Japanese talent agency promoting groups of male idols.

[7]A kind of Japanese rock'n roll band style.

[8]Woman who likes comics depicting male homosexual love.

[9]Japan football league fans.

[10]Fans of a boy idol group.

Table 5.1: Fraction of workers who chose the corresponding rating for each community

| Community | Rate | | | Label[5] |
|---|---|---|---|---|
| | *adequate* | *inadequate* | *unable to judge* | |
| 1 | 80% | 10% | 10% | Johnny's fans[6] |
| 2 | 100% | 0% | 0% | HS students in Tohoku |
| 3 | 80% | 5% | 15% | DJs and Raggae fans |
| 4 | 95% | 0% | 5% | University students in Kansai |
| 5 | 100% | 0% | 0% | University students in Kyushu |
| 6 | 90% | 5% | 5% | Visual-kei[7] |
| 7 | 45% | 25% | 30% | Ethnic Korean |
| 8 | 95% | 5% | 0% | HS students in Chiba/Ibaraki |
| 9 | 100% | 0% | 0% | HS students in Tochigi/Gunma |
| 10 | 70% | 10% | 20% | Fujoshi[8] |
| 11 | 90% | 0% | 10% | Bike fans |
| 12 | 95% | 0% | 5% | J-league fans[9] |
| 13 | 85% | 0% | 15% | Online game fans |
| 14 | 90% | 0% | 10% | HS in Okayama |
| 15 | 100% | 0% | 0% | HS in Shizuoka |
| 16 | 100% | 0% | 0% | HS students in Niigata |
| 17 | 80% | 5% | 15% | Disney fans |
| 18 | 75% | 0% | 25% | Darts and basketball fans |
| 19 | 100% | 0% | 0% | University students in Aichi/Mie |
| 20 | 95% | 5% | 0% | Tsukuba University & surrounding area |
| 21 | 95% | 0% | 5% | Fans of indie label |
| 22 | 80% | 20% | 0% | University students in Tokyo |
| 23 | 60% | 30% | 10% | No Nukes, leftists, housewives |
| 24 | 95% | 5% | 0% | HS students in Osaka |
| 25 | 95% | 0% | 5% | HS students in Nagano/Yamanashi |
| 26 | 80% | 5% | 15% | First-person shooting game fans |
| 27 | 90% | 5% | 5% | University students in Iwate |
| 28 | 90% | 5% | 5% | Korean Pop |
| 29 | 85% | 10% | 5% | Japanese comedy fans |
| 30 | 95% | 5% | 0% | HS students in the Metropolitan area |
| 31 | 85% | 5% | 10% | Anime and music game fans |
| 32 | 85% | 5% | 10% | Fans of vocaloid and singers |
| 33 | 85% | 5% | 10% | Japanese Pop fans |
| 34 | 100% | 0% | 0% | Gambling, mobile game fans |
| 35 | 85% | 5% | 10% | Female idol fans |
| 36 | 75% | 5% | 20% | "Furry" lovers |
| 37 | 75% | 10% | 15% | Fans of "Arashi"[10]and local idol group |
| 38 | 95% | 5% | 0% | University students in Hokkaido |

- We crawl the biography of each user on Twitter profiles.
- For each community, we aggregate the biography texts of users who belong to the community, creating one document that corresponds to the community.

- We calculate the TF-IDF score of all words in each community and extract the top 20 keywords as salient keywords for each community.
- We label each community manually using phrases associated with common features of the top 20 keywords.

To validate the labels, we conducted a user study and evaluated the results. We conducted the user study using a crowdsourcing service in Japan, "Lancers" [11]. We provided the label we created and the top-20 keywords for each community. Then we asked 20 highly acclaimed workers to judge the label for each community by choosing from the following options: (1) *adequate*, (2) *inadequate*, (3) *unable to judge*. For each worker, we paid USD 2.4 for completing the set of tasks: that is, judging the labels for all 38 communities. In the case when a worker chose an *inadequate* label for a community, we asked the worker to provide a more adequate label.

In all the communities, we were able to make sense of the representative words and label the communities. In Tables 5.6 – 5.10, we show a label and the top-3 words instead of top-20 words for each community. We were able to group the communities into three major types (namely all except for one community, "Ethnic Korean"): (a) same/neighboring high schools, (b) same/neighboring universities, and (c) interest-based communities.

We summarize the results of the evaluation from crowdsourced workers in Table 5.1. For each community, we show the fraction of workers who chose the corresponding rating. We can see that for most of the communities, the majority of workers agree that the label is adequate. More specifically, for 32 out of 38 communities (84% of the communities), more than 80% of the workers agree that the label is adequate. Although we asked workers to use a search engine when they could not make sense of the word, some words are polysemous words that are used differently in different communities, and it might be difficult for some workers to make sense of them. This would produce some fraction of an *unable to judge* rating. In the case where workers chose the *inadequate* rating, they were required to provide an alternative label. In most of the cases, the provided labels were not different substantially from the original label; for example, they added auxiliary information to the original label.

## 5.3 Experiment

Here we describe the procedure that is used to prepare the corpus for training and the test, and to train the models.

[11] `http://www.lancers.jp`

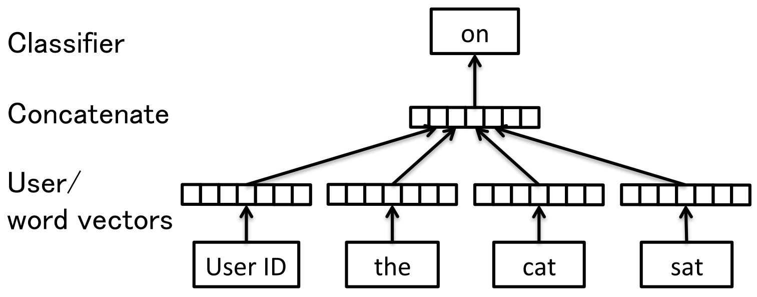
### 5.3.1 Training BOW/embedding models of users

To balance the data size of each community, we randomly sampled 10,000 users from each, and used all the tweets generated by the selected users in the dataset (that is, for a year). We tokenize the tweets with MeCab[12] and IPAdic[13]. We used tweets of 9,000 users for the training set, and 1,000 users for the test set. The data size is shown in Table 5.2.

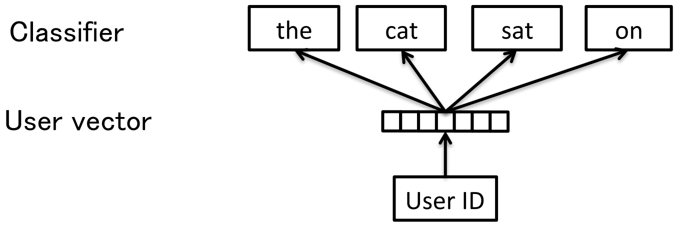
Table 5.2: Data size of the training set and the test set.

| | Data size | Words | Lines | Users |
|---|---|---|---|---|
| Training Set | 12,847 MB | 1,732M | 127,662K | 342,000 |
| Test Set | 1,433 MB | 193M | 14,197K | 38,000 |

As to embedding models, we used the Entity Vector models and the Category Vector models. Here we call the Entity Vector models "User Vector Models" (consisting of UV-DM and UV-DBOW models), and the Category Vector models "Community Vector Models" (consisting of CmV-DM and CmV-DBOW models) respectively. We used a user ID for an entity ID (as in Figure 5.1), and a community ID for a category ID (as in Figure 5.2).



(a) UV-DM model



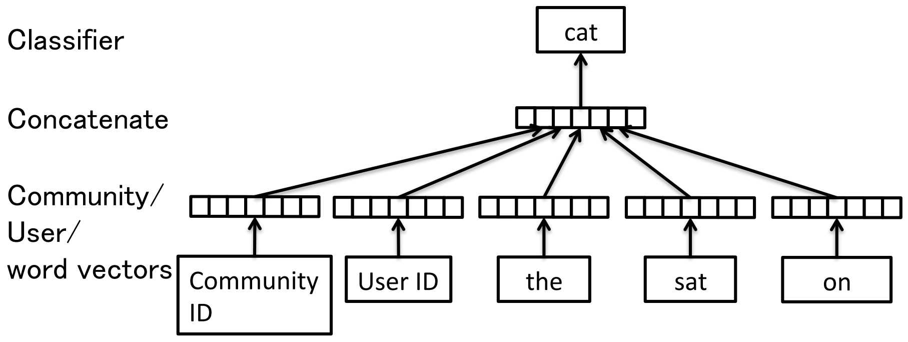
(b) UV-DBOW model

Figure 5.1: User Vector Models.

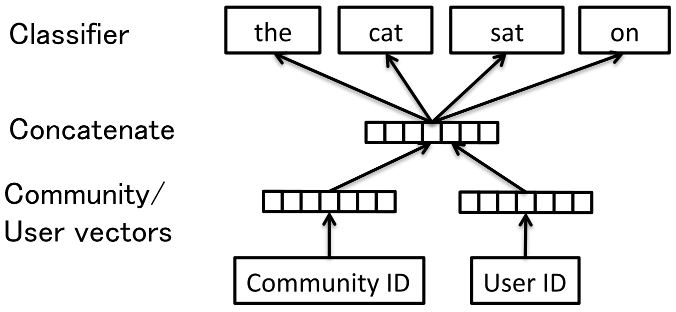
In the BOW model, we created a BOW vector for each user, weighting each word by a TF-IDF score. We also stored the IDF scores for words in the training

[12]`https://code.google.com/p/mecab/`

[13]available from the website of MeCab



(a) CmV-DM model



(b) CmV-DBOW model

Figure 5.2: Community Vector Models.

set, to utilize the same IDF score for the test set. We implemented the BOW model in C++ with OpenMP and it took almost 5 days to process the whole test set on Intel Xeon E5-2680 v2 2.8GHz (10 cores).

For the embedding models, we explored the hyper-parameters shown in Table 5.3. At this time, the explored parameters are smaller than those in chapter 4 since the data size is much larger. However, because the number of the entities is smaller than those in chapter 4, we can adopt Adam [48] for the update rule at this time. We trained the model giving the training set 10 times, and we used the learned word embeddings to train the embeddings of test instances. We used the implementation mentioned in chapter 3. It took 1.9 hours for an iteration of training the DM models and 3 hours for an iteration of training the DBOW models on Intel Xeon E5-2650 v2 2.6GHz (8 cores).

In the same way as chapter 4, we calculated the cosine similarity between an embedding of the test instance and the learned embeddings in the training set, and chose the most similar item in the training set to infer its community.

Table 5.3: Parameters explored in this task. UV denotes User Vector models, and CmV denotes Community Vector models.

| Hyper-parameter | Explored Values / Settings | Methods |
|---|---|---|
| model | DM, DBOW | UV, CmV |
| output layer | HS, NEG | UV, CmV |
| update rule | SGD, AdaGrad | UV, CmV |
| | Adam | CmV |
| dimension | 300 | UV, CmV |
| #negative samples | 5 | UV, CmV |
| alpha / $\eta$ | 0.025 | UV, CmV |
| iterations | 10 | UV, CmV |
| window size | 5 | UV, CmV |
| subsampling | none | UV, CmV |

## 5.4 Results

First, we show learned community embeddings in the Community Vector models. We extracted the community embeddings from the CmV-DBOW model with AdaGrad and NEG, and performed the PCA to reduce the dimension to two. We plotted the community embeddings along with their labels in Figure 5.3. Since the high school communities are agglomerated in a certain area, we plotted their embeddings in the magnified space as in Figure 5.4. As we can see in the figure, the embeddings of high school communities and university communities are plotted close to each other, although there are some exceptions such as the community "Tsukuba University & surrounding area" and "High school students in Osaka". The community "Ethnic Korean" is close to other high school communities because there are students in Korean schools in the community. Although we can see some tendencies among interest-based communities, as the community "Visual-kei" is close to "Indies" (both of them are music fans), the interest-based communities are more scattered in the plot area.

As to the inference task, we summarize the result in Table 5.4. Most of the embedding models outperform the BOW model, of which accuracy is 19.61%, and the Community Vector models outperform the User Vector models except for the CmV-DBOW model with SGD and HS. Among the Community Vector models, models with AdaGrad outperformed the other models, and the models with Adam performed the worst. Although Adam is used for deep neural networks, we did not get good optimizations in our models. The reason could be that the algorithm is almost linear except for the approximation of the softmax function,

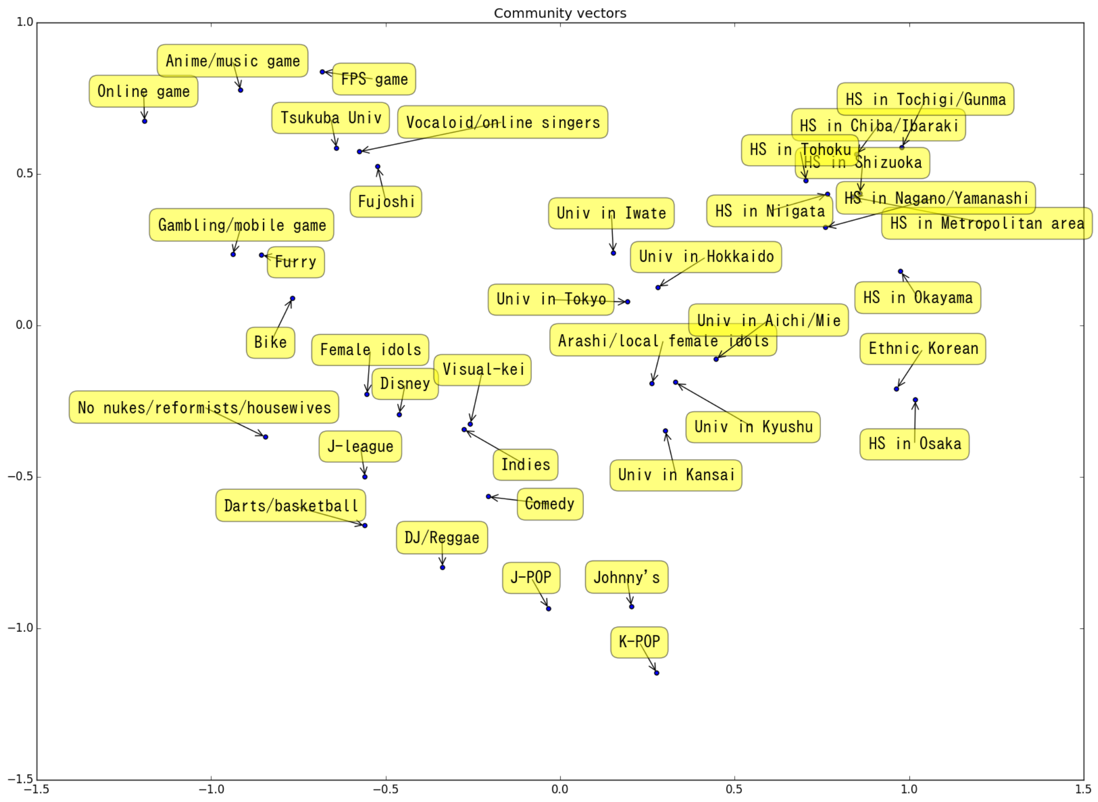


Figure 5.3: Learned Community Vectors.

and it is relatively easy for SGD and AdaGrad to optimize. The CmV-DBOW with AdaGrad and NEG achieved the best accuracy, 42.49%.

Using the result of CmV-DBOW with AdaGrad and NEG (the best model), we listed the communities with high and low accuracy in the inference task in Table 5.5. We can see that users in the communities "K-POP", "High school students in Okayama", and "Johnny's fans" are easy to be inferred, while the model does not classify users in the communities "High school students in Metropolitan area", "University students in Tokyo", and "No nukes / leftists / housewives" into the correct communities. While the model classifies high school communities in low accuracy, it classifies users in two communities, "High school students in Okayama" and "High school students in Osaka" in high accuracy. For interest-based communities, the communities "K-POP" and "Johnny's fans" have high accuracy while the Community Vector model infers users in the community "No nukes / leftists / housewives" with low accuracy.

Table 5.4: Community Inference Task

| Model | Output Layer | |
|---|---|---|
| | HS | NEG |
| BOW | 19.61% | |
| UV-DM SGD | 23.64% | 23.40% |
| UV-DBOW SGD | 13.96% | 12.05% |
| UV-DM AdaGrad | 31.40% | 30.44% |
| UV-DBOW AdaGrad | 36.15% | 34.16% |
| CmV-DM SGD | 24.79% | 34.01% |
| CmV-DBOW SGD | 13.78% | 19.14% |
| CmV-DM AdaGrad | 34.06% | 37.53% |
| CmV-DBOW AdaGrad | 38.02% | **42.49%** |
| CmV-DM Adam | 22.75% | 29.56% |
| CmV-DBOW Adam | 6.72% | 10.38% |

Table 5.5: Accuracy Ranking by Community in Inference Task

| Rank | Community | Accuracy |
|---|---|---|
| 1 | K-POP | 73.9% |
| 2 | HS in Okayama | 65.5% |
| 3 | Johnny's fans | 64.1% |
| 4 | HS in Osaka | 63.1% |
| 5 | FPS game fans | 51.3% |
| 6 | Female idols | 51.2% |
| 33 | University students in Iwate | 30.2% |
| 34 | HS in Shizuoka | 29.5% |
| 35 | HS in Nagano / Yamanashi | 27.2% |
| 36 | No nukes / leftists / housewives | 25.8% |
| 37 | University students in Tokyo | 25.3% |
| 38 | HS in Metropolitan area | 24.7% |

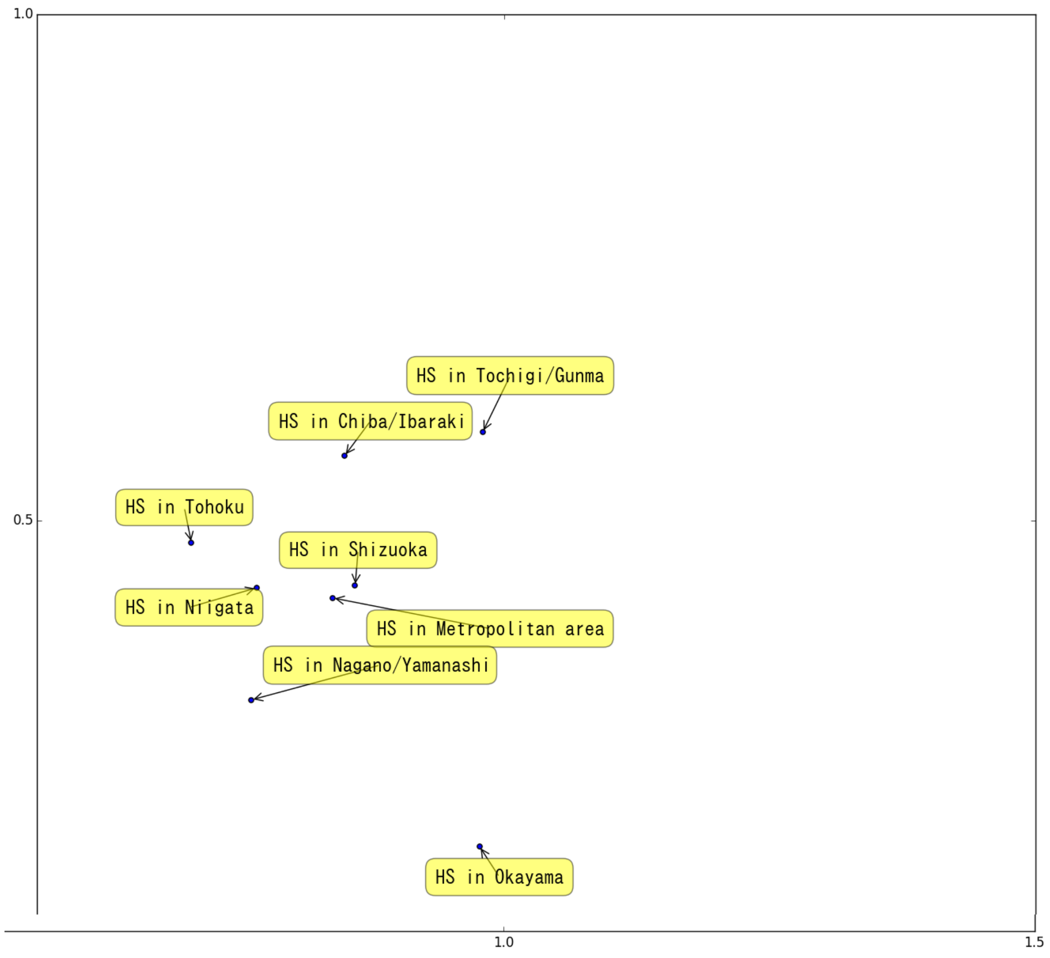


Figure 5.4: Learned Community Vectors (high school communities).

## 5.5 Discussions

In this chapter, we extracted communities genuinely from the network, and we classified users into communities using their contents. To understand the communities, we extracted keywords from their biographies, and we labeled them. As to classification task, the embedding methods outperform the BOW model, and the Community Vector model (i.e., the embedding method considering categories) performed the best among embedding methods.

We used the Louvain method to extract communities. It is applicable to large networks and therefore commonly used in community analysis. However, it has two problems: one problem is that a user cannot join to multiple communities, and another problem is that it often produces one or a couple of big communities along with many small communities. The first problem is because we cut the edges, which are likely to bridge among densely connected subgraphs. The latter problem, which is called “the resolution limit problem”, which is not an easily solvable phenomenon in modularity maximization, as Lancichinetti and Fortunato discussed [53]. Therefore, both of problems are caused from the modularity, which the Louvain method and other methods use to score how well the algorithm divides a network into communities. We can use other algorithms than modularity-based methods; however, it is usually computationally hard to apply such community detection algorithms, as we discussed in chapter 2. As to the resolution limit problem, we can modify the maximization algorithm, accepting lower states of modularity as Fortunato et al. suggests [26], or modify the score itself, as Chen et al. suggests [16].

We labeled communities using keywords created from biographies gathered within each community. Keywords are scored with TF-IDF. However, this method can exaggerate greatly the characteristics that differ from other communities. Due to the resolution limit problem, we tend to have a big community. In addition, large communities have less consistent characteristics. The community “No nukes / leftists / housewives” can be too large since the size of the community is the largest; they have over 1.4 million members. Therefore the keywords of this community can be overemphasized since the TF-IDF score reduces when a term is used in different communities. Probably because of this size, the accuracy in the inference task for this community is low as shown in Table 5.5.

Among high school communities, the embedding model classifies users in two communities, “High school students in Okayama” and “High school students in Osaka” in high accuracy. When we see the posts by users in these communities, we observed that they tend to use dialects in their posts. Their dialects tend to appear in auxiliary verbs or particles, which can be one of the reasons why the embedding models perform better than the BOW model; the embedding models catch the local contexts of words, while the BOW model ignores the word order. Another reason for the improvement in accuracy can be the distribution of word frequency in colloquial expressions. Figure 5.5 shows the distribution in ranking of word frequency, and the distribution has a fat tail that is fatter than that for R-EN data in chapter 4.

We also saw the distribution of community embeddings that reflects the tendency of contents generated by users, and we observed that embeddings of high school and university communities are agglomerated. Therefore it is indicated that high school communities and university communities are similar to each other since they are connected for environmental reasons. We will explore the difference between high school / university communities and interest-based communities in the

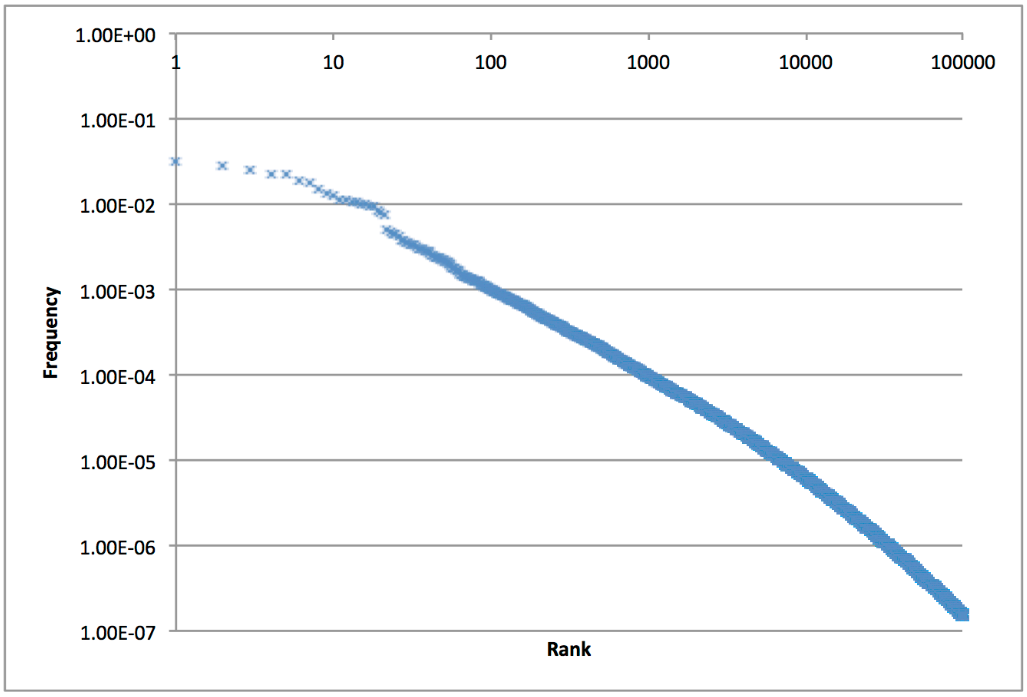


Figure 5.5: The word frequency and the rank in Twitter data on a log–log scale

next chapter.

Table 5.6: Labels, Keywords and Types of the top-38 communities (1)

| Community id | 1 | 2 |
|---|---|---|
| Type | interest-based | high schools |
| Label | Johnny's fans | High school students in Tohoku |
| (Japanese) | ジャニーズ | 東北地方の高校 |
| Keyword-1 | ftr | 磐城 (Iwaki) |
| | [slang meaning a boys idol] | [high school in the area] |
| Keyword-2 | hyphen | 湯本 (Yumoto) |
| | [slang meaning fans of a boy idol group] | [high school in the area] |
| Keyword-3 | drdr | 松陵 (Shoryo) |
| | [slang used by Johney's fan] | [high school in the area] |

| Community id | 3 | 4 |
|---|---|---|
| Type | interest-based | universites |
| Label | DJs and Reggae fans | University students in Kansai |
| (Japanese) | DJ、レゲェ | 関西地方の大学 |
| Keyword-1 | fundoshi | 関大 (Kandai) |
| | [DJ event] | [univ in the area] |
| Keyword-2 | Selector | 同志社 (Doshisha) |
| | [Reggae DJ] | [univ in the area] |
| Keyword-3 | REGGAE | オリター (Oritaa) |
| | | [slang used by univ students] |

| Community id | 5 | 6 |
|---|---|---|
| Type | universites | interest-based |
| Label | University students in Kyushu | Visual-kei |
| (Japanese) | 九州地方の大学 | ヴィジュアル系 |
| Keyword-1 | 九大 (Kyudai) | Teru |
| | [univ in the area] | [name of a vocal of Visual kei band] |
| Keyword-2 | 西南 (Seinan) | ドエル (Doeru) |
| | [univ in the area] | [slang used by fans of a Visual kei band] |
| Keyword-3 | 福岡大学 (Fukuoka University) | ハイヲタ (Haiwota) |
| | [univ in the area] | [slang used by fans of a Visual kei band] |

| Community id | 7 | 8 |
|---|---|---|
| Type | others | high schools |
| Label | Ethnic Korean | High school students in Chiba/Ibaraki |
| (Japanese) | 韓国・朝鮮系 | 千葉県、茨城県の高校 |
| Keyword-1 | (Hangul) | 鉾 (Hoko) |
| | | [high school in the area] |
| Keyword-2 | (Hangul) | 柏陵 (Hakuryo) |
| | | [high school in the area] |
| Keyword-3 | (Hangul) | 松国 (Matsukoku) |
| | | [high school in the area] |

Table 5.7: Labels, Keywords and Types of the top-38 communities (2)

| Community id | 9 | 10 |
|---|---|---|
| Type | high schools | interest-based |
| Label | High school students in Tochigi/Gunma | Fujoshi |
| (Japanese) | 栃木県、群馬県の高校 | ボーイズラブ、腐女子 |
| Keyword-1 | 白楊 (Hakuyo) | 捏造 (lit. forgery) |
| | [high school in the area] | [figment] |
| Keyword-2 | 栃 (Tochi) | 受け (Uke) |
| | [character related to the area] | [slang used by Fujoshi] |
| Keyword-3 | 宇工 (Uko) | 語句 (phrase) |
| | [high school in the area] | |

| Community id | 11 | 12 |
|---|---|---|
| Type | interest-based | interest-based |
| Label | Bike fans | J-league fans |
| (Japanese) | バイク、ツーリング | J-リーグ |
| Keyword-1 | ニコツー (niconico touring) | ゼルビア (Zerubia) |
| | [community of touring] | [soccer club of J-league] |
| Keyword-2 | 新居浜高専 (Niihama-Kosen) | アビスパ (Abisupa) |
| | [college] | [soccer club of J-league] |
| Keyword-3 | レースシム (Race Simulation) | SOCIO |
| | | [word related to soccer] |

| Community id | 13 | 14 |
|---|---|---|
| Type | interest-based | high schools |
| Label | Online game fans | High school students in Okayama |
| (Japanese) | MMO、オンラインゲーム | 岡山県の高校 |
| Keyword-1 | トミーウォーカー (Tommy Walker) | 芳泉 (Housen) |
| | [game company] | [high school in the area] |
| Keyword-2 | バロックナイトイクリプス (Baroque night eclipse) | 西大寺 (Saidaiji) |
| | [title of online game] | [high school in the area] |
| Keyword-3 | PBW | 就実 (Shujitsu) |
| | [kind of online game] | [high school in the area] |

| Community id | 15 | 16 |
|---|---|---|
| Type | high schools | high schools |
| Label | High school students in Shizuoka | High school students in Niigata |
| (Japanese) | 静岡県の高校 | 新潟県の高校 |
| Keyword-1 | 沼津 (Numazu) | 北越 (Hokuetsu) |
| | [high school in the area] | [high school in the area] |
| Keyword-2 | 星陵 (Seiryo) | 五泉 (Gosen) |
| | [high school in the area] | [high school in the area] |
| Keyword-3 | 宮北 (Miyakita) | 敬和学園大学 (Keiwagakuin University) |
| | [high school in the area] | [univ in the area] |

Table 5.8: Labels, Keywords and Types of the top-38 communities (3)

| Community id | 17 | 18 |
|---|---|---|
| Type | interest-based | interest-based |
| Label | Disney fans | Darts and basketball fans |
| (Japanese) | ディズニーマニア | ダーツとバスケットボール |
| Keyword-1 | パレ (Pare) | TDO |
| | [slang meaning parades] | [abbreviation of a darts organization] |
| Keyword-2 | ミキヲタ (Mikiwota) | FiGARO |
| | [slang meaning huge Disney fans] | [name of darts bar] |
| Keyword-3 | 栗鼠 (Risu) | - |
| | [slang meaning Chip'n Dale] | [a special character] |

| Community id | 19 | 20 |
|---|---|---|
| Type | universities | universities |
| Label | University students in Aichi/Mie | Tsukuba University & surrounding area |
| (Japanese) | 愛知県、三重県の大学 | 筑波大学周辺コミュニティ |
| Keyword-1 | 椙山 (Sugiyama) | UEC |
| | [university in the area] | [university related to the university] |
| Keyword-2 | 南山 (Nanzan) | 筑波大学 (Tsukuba University) |
| | [university in the area] | |
| Keyword-3 | 中京 (Chukyo) | klis |
| | [university in the area] | [abbr. a department of the university] |

| Community id | 21 | 22 |
|---|---|---|
| Type | interest-based | universities |
| Label | Fans of indie label | University students in Tokyo |
| (Japanese) | インディーズ系バンド | 東京都の大学 |
| Keyword-1 | NUBO | 立教 (Rikkyo) |
| | [indie label] | [univ in the area] |
| Keyword-2 | BAW | 立教大学 (Rikkyo Univ) |
| | [indie label] | [univ in the area] |
| Keyword-3 | OAT | 法政 (Hosei) |
| | [indie label] | [univ in the area] |

| Community id | 23 | 24 |
|---|---|---|
| Type | interest-based | high schools |
| Label | No Nukes, lefists, housewives | High school students in Osaka |
| (Japanese) | 原発・左翼・主婦 | 大阪府の高校 |
| Keyword-1 | 原発 (Nuclear power plant) | 岸城 (Kishiki) |
| | | [high school in the area] |
| Keyword-2 | 被曝 (Exposure) | 日根野 (Hineno) |
| | | [high school in the area] |
| Keyword-3 | 主宰 (To preside over) | ヒガスミ (Higasumi) |
| | | [high school in the area] |

Table 5.9: Labels, Keywords and Types of the top-38 communities (4)

| Community id | 25 | 26 |
|---|---|---|
| Type | high schools | interest-based |
| Label | High school students in Nagano/Yamanashi | First-person shooting game fans |
| (Japanese) | 長野県、山梨県の高校 | FPS ゲーム |
| Keyword-1 | 美須々 (Misuzu) | SuddenAttack |
| | [high school in the area] | [slang used by FPS game fans] |
| Keyword-2 | 松商学園 (Matsusho-gakuen) | osu |
| | [high school in the area] | [slang used by FPS game fans] |
| Keyword-3 | 巨摩 (Koma) | サドンアタック (Sudden Attack) |
| | [high school in the area] | [slang used by FPS game fans] |

| Community id | 27 | 28 |
|---|---|---|
| Type | universities | interest-based |
| Label | University students in Iwate | Korean Pop |
| (Japanese) | 岩手県の大学 | K-POP |
| Keyword-1 | 岩手大学 (Iwate University) | ジェジュン (Jejung) |
| | | [Korean pop talent] |
| Keyword-2 | 盛岡大学 (Morioka University) | ヌナ (Nuna) |
| | [univ in the area] | [Korean pop talent] |
| Keyword-3 | 盛岡 (Morioka) | shawol |
| | [place in the area] | [slang used by Korean pop fans] |

| Community id | 29 | 30 |
|---|---|---|
| Type | interest-based | high schools |
| Label | Japanese comedy | High school students in the Metropolitan area |
| (Japanese) | お笑い | 東京、神奈川、埼玉の高校 |
| Keyword-1 | エージェンシー (lit. agency) | 松が谷 (Matsugaya) |
| | [abbr. of Yoshimoto Creative Agency] | [high school in the area] |
| Keyword-2 | NSC | 大宮西 (Omiyanishi) |
| | [school for comedians] | [high school in the area] |
| Keyword-3 | nsc | 瀬谷 (Seya) |
| | [school for comedians] | [high school in the area] |

| Community id | 31 | 32 |
|---|---|---|
| Type | interest-based | interest-based |
| Label | Anime and music game fans | Fans of vocaloid and singers |
| (Japanese) | 音ゲー・ギャルゲー・ニコニコゲーム系 | ニコニコ歌系 |
| Keyword-1 | ミサカ (Misaka) | UTAU |
| | [character in a anime] | [singing synthesizer application] |
| Keyword-2 | ニコマス (Nikomasu) | こえる (to exceed) |
| | [slang used by a music game fans] | |
| Keyword-3 | ACV | 真似 (mimicry) |
| | [a game title] | |

Table 5.10: Labels, Keywords and Types of the top-38 communities (5)

| Community id | 33 | 34 |
|---|---|---|
| Type | interest-based | interest-based |
| Label | Japanese Pop | Gambling, mobile games fans |
| (Japanese) | J-POP | パチンコ、ネット麻雀、携帯ゲーム |
| Keyword-1 | Leaders | マジモン (Majimon) |
| | | [monster game] |
| Keyword-2 | 一座 (Ichiza) | 鳳東 (Otorihigashi) |
| | [show] | [slang used by a gambling fans] |
| Keyword-3 | Lead | サミタ (Samita) |
| | [a group of J-POP] | [slang used by a gambling fans] |

| Community id | 35 | 36 |
|---|---|---|
| Type | interest-based | interest-based |
| Label | Girl idol fans | ”Furry” lovers |
| (Japanese) | アイドル | ケモナー |
| Keyword-1 | クノ (Kuno) | ケモノ (Kemono) |
| | [slang used by a female idol fans] | [Furry] |
| Keyword-2 | アイドリング (Idoling) | 組合 (association) |
| | [a group of female idols] | |
| Keyword-3 | AKIHABARA | ケモナー (Kemonaa) |
| | | [slang used by ”Furry” lovers] |

| Community id | 37 | 38 |
|---|---|---|
| Type | interest-based | university |
| Label | Fans of “Arashi” and local girl idol | University students in Hokkaido |
| (Japanese) | 嵐、ローカルアイドル | 北海道の大学 |
| Keyword-1 | ゴゴイチ (Gogoichi) | 北星 (Hokusei) |
| | [slang used by “Arashi” fans] | [high school in the area] |
| Keyword-2 | 磁石 (Jishaku) | 北海学園大学 (Hokkaigakuen University) |
| | [slang used by “Arashi” fans] | [high school in the area] |
| Keyword-3 | nr | 札 (Satsu) |
| | [slang used by “Arashi” fans] | [character related to the area] |

# Chapter 6

# Analysis of Word Usage Using the Embedding Method

In the previous chapter, we applied the category-enhanced embedding method which learns community embeddings for the community inference task. The Community Vector models (i.e., the models of implicit categories) perform the best in the task. Although the Category Vector models perform well in the task of the category inference in e-commerce data compared to the BOW model, the Community Vector models improved the accuracy more. Additionally, there are difference between high school / university communities and interest-based communities, as we saw from the plot of community embeddings through PCA. To investigate the reason for the improvement in the inference task and the differences between community types, we analyze word usage through the word embeddings learned from the contents generated by users. We take the difference in word embeddings learned differently from the corpus of each community. We also see the network ties between communities to explore the relationship between the word usage and network structure.

## 6.1 Datasets

We follow the methodology described in section 3.4. We sampled the 1% from the whole dataset to create the training set for pre-training. Next we create a corpus for each community. We sample the fraction of $10,000/N_i$ ($N_i$:the number of users in the $i$-th community) from users' contents in each community. We use all the data for the community with 10,000 users, and we choose the communities with more than 10,000 users; therefore, we can balance the size of the corpora.

Table 6.1: Hyper-parameters in the Skip-gram model

| Hyper-parameter | Value / Setting |
|---|---|
| output layer | NEG |
| update rule | SGD |
| dimension | 200 |
| #negative samples | 5 |
| alpha | 0.025 |
| window size | 10 |
| subsampling | $10^{-3}$ |

Table 6.2: List of Community Ambiguous Words

| | |
|---|---|
| 1 | ã |
| 2 | talk |
| 3 | ⊠ (Hangul;laughing) |
| 4 | ミート (miito) |
| 5 | des |
| 6 | 知識人 (Chishikijin) |
| 7 | 町長 (Chōchō) |

## 6.2 Training word embedding models

We trained the Skip-gram model under the hyper-parameters shown in Table 6.1. We adopt NEG, which is reported to perform better than HS in several tasks. The value of alpha decreases to 0.001 during pre-training, and resets to 0.025 and decreases to 0.001 once again during training from the sampled community contents. We train the total of 38 models. Therefore, we have 38 embeddings for a single word.

## 6.3 Detecting ambiguous words among communities

As we pointed out in chapter 4, we use the term ambiguous words for the words used in several (local) contexts. Therefore, we do not discuss the definition of meanings here. We show some sensible examples in which a word is used in the different local contexts in different communities. As described in section 3.4, we present a rank list in Table 6.2. From highly-dissimilar words, we show an example, "ミート (miito)", which is the fourth dissimilar word in the rank list.

The pronunciation of "ミート (miito)" corresponds to both the English word "meat" and "meet" for Japanese people. The Japanese language has many loan words from European languages, and the pronunciations of such words are sometimes similar to the original words. However, Japanese people typically do not use the word “ミート (miito)” for *meat* and *meet*, because they use words rooted in Japan, although they can understand the meaning of the word because they know the English word “meat”/“meet”. Thus, it can be inferred that the word is a kind of slang term, which is used by people who share contexts for the word. With these in mind, we then conducted the analysis.

In Table 6.3 and Table 6.4, we show the top-10 context words for each community, with corresponding scores.

In community-13 (Online game fans), most of the potentially co-occurring words are words related to foods. Since the word “ミート (miito)” can mean meat in English as noted above, we can infer that people in this community use “ミート (miito)” to mean meat.

In contrast, in community-17 (Disneyland fans), at a glance it is not clear what the word “ミート (miito)” stands for. However, it is easy to see that these words reflect the characteristics of the community, which is mainly composed of female users who are enthusiastic fans of Disneyland; (1) the face mark (a kind of emoticon) (> _ <) and ??♡ are typically used by some young females, (2) it is also typical word usage by some young females to repeat the same word such as “do do” and “please please”, (3) the word “Inn” is a slang term, which means to enter somewhere in Disneyland, and which is used by enthusiastic fans of Disneyland. From usage in the tweets in this community, we found that the word “ミート (miito)” stands for meeting in Tokyo Disneyland. It seems that they use this loan word for expressing *meet*ing in a particular situation.

Although we used a simple measure (cosine) for the word similarity between communities and a naive method to compute them, it is possible to develop a more adequate measure of similarity and an efficient computation method to automatically extract such *ambiguous words across communities,* which is one of the interesting topics for future work. Particularly, we think such an analysis could enhance cultural studies, by comparing slangs across communities.

## 6.4 Similarity among communities in word usage

Now we know some of words are used differently by communities. We can extract the similarities if we accumulate the difference in word embeddings between communities. As we described the embedding model of categories in chapter 3, embeddings of the categories are modeled as context vectors. If the model captures the difference in word usage among categories, similar categories in Equation 3.20

Table 6.3: Context words for the word "ミート (miito)" in community-13 (Online game fans)

| Word | Score |
|---|---|
| 早食い (speed-eating) | 0.821099 |
| フローズン (frozen) | 0.802880 |
| ドスヘラクレス (Dos hercules; a character name) | 0.800420 |
| クランチ (crunchy) | 0.792865 |
| ラード (lard) | 0.791672 |
| ホットドリンク (hot drink) | 0.782958 |
| アサイ (an abbreviation of "Asai meat"; a company name) | 0.778490 |
| 銀シャリ (white rice) | 0.778290 |
| シモフリトマト (marbled tomato; an item name) | 0.775939 |
| レッグ (leg) | 0.775758 |

Table 6.4: Context words for the word "ミート (miito)" in community-17 (Disney fans)

| Word | Score |
|---|---|
| イン (in) | 0.844532 |
| タイミング合え (timing suits) | 0.803420 |
| 次いつ (when next) | 0.801293 |
| ぜひぜひ (please please) | 0.801245 |
| (>_<)!!! | 0.792765 |
| するする (do do) | 0.790120 |
| まなみん ("Manamin"; nickname for a girl's name) | 0.789295 |
| ??♡ | 0.787898 |
| それでもよければ (if it is still OK) | 0.785390 |
| きんかん ("Kinkan"; a fruit name) | 0.782860 |

have similar embeddings in the vector space.

Table 6.5 shows the top-10 similar community pairs based on the word usage, and Table 6.6 shows the top-10 dissimilar community pairs in the same manner. We can see that communities sharing a generation have similar word usage. More specifically, high school students have similar word usage with respect to one another, although they are not *physically* close. This held true for university students. When we compare these results to the community embeddings, we can see that the embeddings of communities listed in Table 6.5 are also close to each other, as Figure 5.3 shows. Interestingly, communities listed in Table 6.6 are relatively distant to each other in the embedding space. From this tendency we can confirm that the embedding model of categories captures the difference in word usage among the groups.

Table 6.5: Top-10 similar community pairs based on the word usage

| | |
|---|---|
| HS students in Metropolitan area | HS students in Chiba/Ibaraki |
| University students in Aichi/Mie | University students in Kansai |
| University students in Kansai | University students in Tokyo |
| University students in Aichi/Mie | University students in Tokyo |
| University students in Aichi/Mie | University students in Hokkaido |
| University students in Kansai | University students in Kyushu |
| Universtiy students in Aichi/Mie | University students in Kyushu |
| HS students in Chiba/Ibaraki | HS students in Tochigi/Gunma |
| University students in Hokkaido | University students in Tokyo |
| HS students in Metropolitan area | HS students in Tochigi/Gunma |

Table 6.6: Top-10 dissimilar community pairs based on the word usage

| | |
|---|---|
| Online game fans | High school students in Osaka |
| Online game fans | High school students in Okayama |
| DJs and Reggae fans | “Fujoshi” |
| Anime and music game fans | High school students in Okayama |
| Anime and music game fans | High school students in Osaka |
| “Fujoshi” | High school students in Okayama |
| “Fujoshi” | High school students in Osaka |
| “Fujoshi” | Darts and basketball fans |
| Anime and music game fans | DJs and Reggae |
| Ethnic Korean | Online game fans |

## 6.5 The relationship between the similarity in network property and the similarity in word usage

We extract communities from a network, and we see that people do not connect with distant people, even though their manners of writing can be very similar (e.g., users in high school communities). This property can be tricky for online marketing and ads; within communities, we can assume that the information propagates over similar people, while we are not sure that the information can propagate from one to other communities. It can be important to see how communities are connected with each other. Therefore, we define the similarity score in the same manner as the similarity score in word usage.

**Definition of similarity between communities based on a network:** We define the network-based similarity between community $i$ and community $j$ , $Sim_{net}(i,j)$ simply as the edge density between them shown in the following:

$$Sim_{net}(i,j) = \frac{|E_{i,j}|}{|V_i||V_j|} \tag{6.1}$$

where $|E_{i,j}|$ is the number of edges between the community $i$ and $j$, and $|V_i|$ is the number of nodes (users) in the community $i$.

Table 6.7 shows the top-10 similar community pairs, and Table 6.8 shows the top-10 dissimilar community pairs based on the network property $Sim_{net}$. From here we can say that the similar communities in word usage are not highly connected (i.e., interacted), and the connected communities are not close to each other in the embedding space.

Table 6.7: Top-10 similar community pairs based on the social network

| | |
|---|---|
| Anime and music game fans | University students in Tokyo |
| Anime and music game fans | "Fujoshi" |
| Anime and music game fans | Fans of vocaloid and singers |
| Anime and music game fans | No Nukes, leftists, housewives |
| Anime and music game fans | Fans of "Arashi" and local female idols |
| Anime and music game fans | Online game fans |
| Anime and music game fans | "Visual kei" |
| University students in Tokyo | High school students in Iwate |
| "Visual kei" | University students in Tokyo |
| Fans of indies | University students in Tokyo |

Table 6.8: Top-10 dissimilar community pairs based on the social network

| | |
|---|---|
| Ethnic Korean | Bike fans |
| High school students in Shizuoka | Darts and basketball fans |
| Bike fans | High school students in Shizuoka |
| Bike fans | High school students in Niigata |
| Ethnic Korean | Darts and basketball fans |
| High school students in Okayama | Darts and basketball fans |
| Bike fans | High school students in Okayama |
| Bike fans | Disney fans |
| Ethnic Korean | Tsukuba University & surrounding area |
| DJs and Reggae fans | High school students in Shizuoka |

We only focused on highly similar/dissimilar communities in terms of word usage and highly connected/disconnected communities. Here we would like to obtain a broad view of the relationship between these two similarities. First, we created a correlation chart that depicts the relationship between the two similarities for all community pairs. The chart is shown in Figure 6.1 (a). The horizontal axis corresponds to the word-usage-based similarity, $Sim_{word}$, and the vertical axis corresponds to the network-based similarity, $Sim_{net}$. We can see that there seem to be no correlations (the correlation coefficient: 0.06463). But we know that there are several types of communities in the target network — high school community, university community and interest-based community — and the embeddings of high school communities and university communities are agglomerated. Thus we can categorize each point on the map based on the type of the community pair to explore the characteristics for each community type. For example, the similarity between school communities can have a different pattern from the similarity between interest-based communities. Therefore we examine where each type of pair is located on the map.

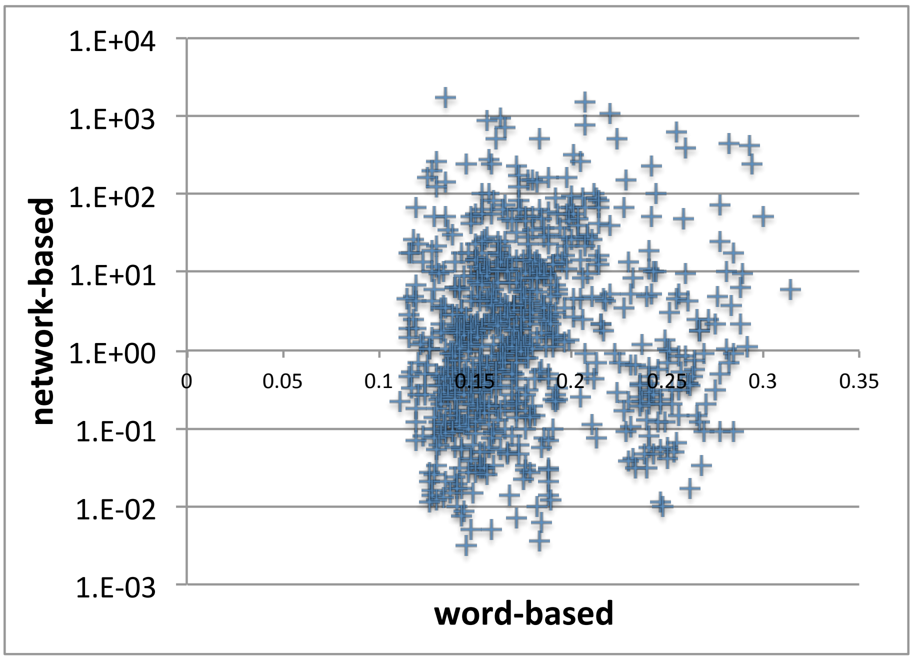
As a result, we found clear patterns. Figure 6.1 (b) shows a correlation map where points are colored based on the type of community pair - blue for community pairs in the university category, red for community pairs in the high-school category and green for community pairs in the interest-based category.

We can examine the map by viewing it in terms of four types:

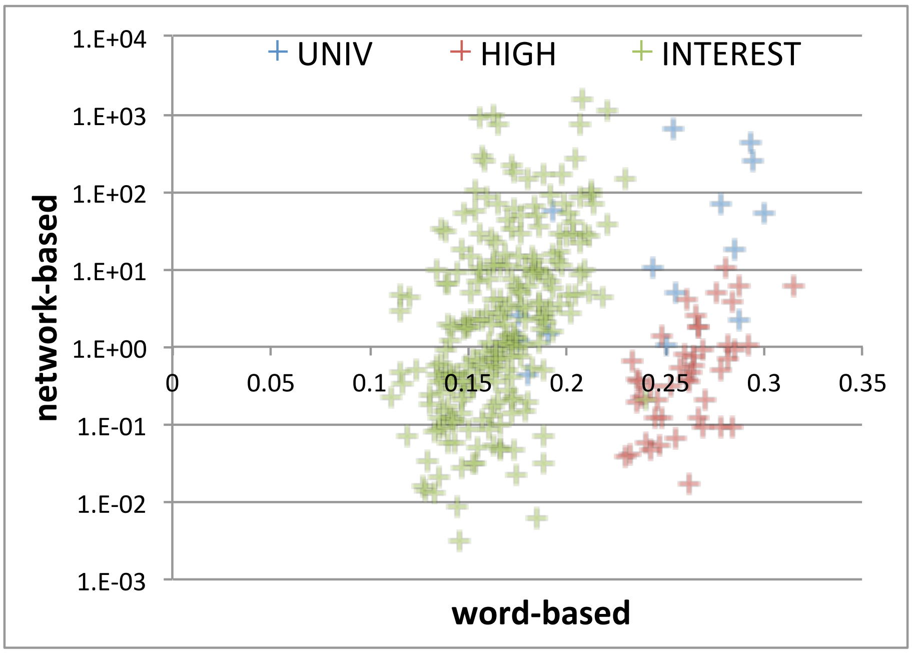
(i) group that has similar word usage and a large amount of interaction
(ii) group that has similar word usage but a small amount of interaction
(iii) group that has dissimilar word usage but a large amount of interaction
(iv) group that has dissimilar word usage and a small amount of interaction

Now we see that university communities can be categorized into type (i), high-school communities can be categorized into type (ii), and interest-based communities belong to type (iii) and (iv). This observation is consistent with the tendency we saw in the previous section by using top-10 similar pairs.

From this observation, we can see that (1) communities based on the member's attributes — high-school or university — tend to have similar word usage, regardless of the interactions they have with respect to one another, while (2) communities based on the member's interests tend to have different word usage, regardless of the interactions they have with respect to one another. It is also suggested that even when people have a common attribute, such as students, the patterns derived from the social network and word usage can differ depending on another attribute, such as generation. These observations suggest the complex relations among social interaction, word usage, and other factors such as their interests and attributes.



(a) Correlation between social-network-based similarity and word-usage-based similarity in entire network



(b) Correlation between social-network-based similarity and word-usage-based similarity for high-school communities, university communities and interest-based communities

Figure 6.1: Social-network-based similarity and word-usage-based similarity.

## 6.6 Discussion

In this chapter, we explored the relationship among the communities in a network and their word usage extracted from a large dataset from Twitter. We found that we can detect *ambiguous words among communities* (that is, words that are used for multiple meanings depending on the communities) by exploiting the embedding method. We also revealed that there are no clear correlations among community-level similarity in interaction and community-level similarity in word usage when we take community as the unit of analysis, but there exist specific patterns based on the types of communities, which are created from people's profile information describing their interests and attributes.

Although the study of Lim et al. [58] and the study of Bryden et al. [14] would be the most related to the analysis described in this chapter, our analysis is different from the extraction of keywords that can characterize communities. Lim et al. detected highly interactive communities with common interests based on the behavioral information of users, the content of hashtags, and mentions. They showed that those communities are more cohesive and connected; however, they focused on only the frequency of words, while we handle words, entities, and communities as embeddings, which enable us to process the words more flexibly and robustly. Bryden extracted communities from a conversation network of Twitter, and extracted the keywords in their tweets that can characterize the communities, while our analysis explored not only the characteristics of communities but also similarities among communities. Our research deals with many more users (7M) than Lim et al. (18 K) and Bryden et al. (189 K).

# Chapter 7

# Discussion

To show the effectiveness of the Category Vector models, we conducted two inference tests on e-commerce data (a dataset with explicit categories) and social media data (a dataset with implicit categories), and we analyzed how our proposing model learned the embeddings of categories that can characterize them. We discuss the limitations and the future perspective of our models and our analysis.

## 7.1 Category Vector models

Here we discuss the issues and the limitations of the Category Vector models.

### 7.1.1 Optimization issues in training models

As we pointed out in chapter 4, the text instances in R-EN are longer than in R-JA, and this factor can affect the performance in the inference task. Here we have several problems: how to choose proper dimension size of embeddings, and how to choose the models and the approximation techniques (and other hyper-parameters). If we have long descriptions, then we should increase the embedding dimension as well. However, practically the dimension is chosen from the limitation of the RAM. We saw DBOW models perform better than DM models except for the case when the description length is relatively long and the frequency of words drops faster. Also, when we employ SGD for the update, use of NEG often drops the performance, while DBOW models with NEG often work fine when we use AdaGrad. From the evaluations, we can say that DBOW models with AdaGrad and NEG often perform well. However, we can explore the relationship between hyper-parameters and the data properties.

It is sometimes difficult to obtain a good optimization since the number of parameters is larger than for the Entity Vector models (Paragraph Vector models), and the number of updating is unbalanced among categories (very frequent), entities (very infrequent), and word (some of them are very frequent and others are infrequent). We improved the accuracy utilizing AdaGrad, which mitigates the fluctuation affected by the hyper-parameter unlike SGD. However, such an adaptive method cannot solve the unbalanced updates among parameters entirely; we need the better method to obtain the better optimization.

Similar to this problem, the training set is sometimes very large in this setting, which leads to the long training time. For example, it took more than a day to train one of our models (SGV-DBOW model) from the social media data used in chapter 5. To obtain optimized models in a shorter time, we can explore how many samples are sufficient for training embeddings of categories and entities since we do not need to use all the data every time in the iterations.

### 7.1.2 The limitation of the models

One of the important limitations to our models is that there is a difficulty in handling the polysemies and ambiguous words since we usually assign a vector for each surface of the term. This limitation is widely applied to other embedding methods and there are no gold-standard way to avoid them. We could assign multiple embeddings for each surface; however, it is difficult to estimate how many embeddings we should assign for a certain surface. Mnih et al. tried to assign several embeddings for words, of which local contexts are near to the separating hyperplane in clustering [67]. However, their method tends to assign multiple embeddings for infrequent words, of which local contexts are not enough.

### 7.1.3 Possible improvements on joint models and multilingual models

To obtain better embeddings of the categories and entities, we can use the joint model, in which we merge the embeddings from two models. We can also use the joint model of three or more models in different hyper-parameters, although it requires more memory space and calculation time.

As to multilingual models, we can use three or more language sources, or we can use the models to complement the correspondence table among different languages in a bootstrap way. Multilingual models usually have constraints between the word or phrase embeddings in different languages, using multilingual corpus and word/phrase alignment e.g. bilingually-constrained recursive auto-encoders (BRAE) by Zhang et al. [101]. We used the same category vectors between English and Japanese data, but we can introduce a matrix to constrain the category vectors among different languages to make the embeddings more flexible.

### 7.1.4 Applications of the models

As shown in experiments, the Category Vector models are widely applied for massive text data with categories since they require non-hierarchical categories, which are generally easily obtained. If properties or tags are not available for obtaining categories, they can also use other methods, such as community detection algorithms, can be used. In addition, the training time is short enough to process all of the data — whole English Wikipedia articles can be processed within several hours with a workstation — and the learned representations are dense vectors, therefore they can be used for machine learning techniques such as SVM and logistic regressions and so on.

In the e-commerce domain, embedding models using the categories can enhance the tasks such as named entity recognition (NER), property extraction, and product linking to identify the product. A number of papers describe such tasks. Ghani et al. extracted attribute and value pairs from product descriptions in a supervised way [29]. Mauge et al. proposed property extraction from the eBay free text product database [60]. The method consists of unsupervised property discovery/extraction from unstructured listings, and supervised property synonym discovery using a clustering algorithm. Zhao et al. tackled product linking using semi-supervised training from little labeled data [102]. They used a grouping and self-training approach to link products from product attributes. Joshi et al. used word representations of word2vec for the NER task, and showed that word vectors trained from in-domain data are more effective than word vectors trained from out-of-domain data [45]. Our models discussed in this thesis can replace the representations of products and they can be used to further improve methods mentioned above.

As many companies gather customer information, the importance of analyzing customers activity in their site has been rising. Generally their customers and their activities is various; the matrices of products and customer activity, including purchased items are typically sparse. To mitigate this sparsity, we can apply the category-enhanced method since customers and products can have their embeddings in the form of dense vectors. For example, we can apply the Category Vector models to the purchase history. Considering sets of purchased products as entities, we can regard a customer as a category, and each customer as an embedding. We can train the customer embeddings from product ID and its product descriptions and see which customers are similar in the purchase activity. We handled the data in which an entity belongs to a single category; however, our models are also applicable to data that includes many-to-many relationships between categories and entities. Utilizing product and customer vectors, it is possible for companies to analyze their service and their customers more flexibly.

In the social media domain, some studies try to embed the network itself (e.g., such as Tang et al.'s LINE [90]). Among such studies, our method can extend the work of Perozzi et al.; they applied the word2vec to the node sequence obtained from random walks on the network [74]. They succeeded in extracting the node embeddings, and they showed that the embedding method of NLP is also applicable to the network data. Our models can also apply to the sequence of random walks on social media, and they can extract the embeddings of nodes, their attributes, and communities. For example, we can apply the Category Vector models to activity logs, such as log data of passengers in transportation, to classify people into several segments or communities.

Social media are used to investigate what people say about certain products and what types of information people are interested in. They use various words to express their various opinions. Their opinions and word usages can vary by their backgrounds, which cannot be obtained easily. In such cases, we can utilize community detection algorithms to divide users into communities in which users are supposed to be alike. At the same time, we should consider the similarity between communities and users since users can be alike even if they belong to other communities. In marketing and advertising, we can also apply the category-enhanced method to obtain such similarities between communities and users. Once we have obtained the embeddings of users and communities, we can capture how user communities are distributed, and we can also predict some future user activities using their embeddings.

## 7.2 Analysis of word usage using the embedding method

In this section, we discuss the issues of the methodologies we took to analyze the difference in local contexts among categories.

### 7.2.1 Limitations and issues of detecting ambiguous words

While we detected a certain kind of ambiguous words, *ambiguous words among communities*, in chapter 6, there are several issues and limitations. One of the issues in this detection is that the result hugely depends on the method by which we divide a sentence into words. In English and other European languages, it is easy to split a sentence into words. However, the Japanese language has no spaces between words, and there is a difficulty in dividing noun phrases. When we divide the noun phrases correctly, we can use the dictionary that covers most of the words. However, it can decrease the accuracy in colloquial texts. We listed the ambiguous words among communities in chapter 6, and the most of the words listed in the high rank come from the mistaken division. One of the limitations is that we cannot always extract the ambiguous words that have multiple meanings just from their local contexts. As Mnih et al. tried to extract ambiguous words in their experiment [67], the local contexts of infrequent words fluctuate often. In our experiment, we filtered ambiguous words among communities of low frequency; however, it can be hard to set an adequate threshold in order to extract ambiguous words.

### 7.2.2 Issues of extracting similarity in word usage

While we use the model trained from globally sampled data and use the model for a pre-trained model, retraining can distort some word embeddings through training. If the word embeddings of highly frequent words change, the embeddings of infrequent words are left behind since they are rarely updated during training. To avoid this issue, we can fix the embeddings for highly frequent words and grammatical words.

We discussed that the Category Vector models can capture the dialect since it classified users who have dialects in chapter 5. This tendency was held true in the analysis in chapter 6, since the communities which have dialects are shown in the most dissimilar community pairs. While we used the word embedding method “word2vec” to extract the difference in local contexts, the algorithms of “word2vec” and the Category Vector models are similar. Therefore, we can use other embedding methods such as PPMI with SVD and GloVe to ensure that the embedding method can capture the dialects or characteristic styles of writing.

### 7.2.3 Applications and future works

We showed in chapter 6 that the relationship among closeness in network and similarity in word usage can reveal the characteristics of communities, which is the high school communities do not connect with each other, while the university communities connect with each other more. This result can help to increase the accuracy of the community inference task, which we conducted from the users' contents in chapter 5. Although it is rare to have an entire network of social media, we can use the local network structure to increase the accuracy of the task. Furthermore, the result can bring the insight to infer the communities in the global network simply from their generated contents and the local network structure.

# Chapter 8

# Conclusion

In this thesis, we proposed the embedding models — the Category Vector models — which learn embeddings of the categories, entities, and words. We focus on the categories because there are various entities in the massive data without any taxonomies or external knowledge, and the categories bind entities loosely. We model the categories to analyze entities in massive text data. The models train embeddings, which maximizes the posterior probability of a target word or words in a window given a category and an entity ID as well as the context words. The learned embeddings can be used to plot to the planar space to see how the categories or entities are distributed.

We applied the models to e-commerce data and social media data. Regarding e-commerce data, the models capture the similarity and dissimilarity among categories of products, and the models classify the products into thousands of categories with better accuracy than the previous models. Our target e-commerce sites have a category alignment between English data and Japanese data, and when we merged the data and used the data for training, the multilingual models classified the products in a higher rate than the monolingual models. As to social media, we saw the characteristics of communities in the embedding space; high school communities and university communities are gathered in a certain area, while the interest-based communities are scattered. Again, the models outperformed the previous models in the inference task; they can be used for the inference of community structure in the whole network from the locally sampled data.

To understand our proposed embedding method further, we conducted the analysis of word usage among communities using social media. We regarded communities as categories extracted from networks, and we observed that close communities in the embedding space in our models are also close to each other in the proposed measure, the word-usage-based similarity. This result indicates that our embedding models can learn the differences in word usage. We also examined the relationship between the word-usage-based similarity and the network-based similarity to understand communities better, and we found that each type of community has its own tendency in these similarity scores; university communities are connected to each other relatively highly, and they tend to write in a similar manner. High school communities are not connected to each other, and they also tend to write in a similar way, while the interest-based communities use the characteristic style of writing regardless of their interactions.

In the time of Big data, it is becoming increasingly important for us to process various entities, which have weak structures between themselves, to extract the tendencies or relationships of entities directly from the data. The embedding approach is often suitable for such processing, and we expect that models of categories will be strong tools for online recommendations and ads. We also expect our method to eventually leads to new methodologies or techniques for managing different categories of a variety of types of massive data.

# Acknowledgments

First, I would like to express my sincere gratitude to my supervisor Prof. Ichiro Sakata, for the continual support of my Ph.D. study. His patience and insightful advice helped me throughout my research. It would not be possible to continue my Ph.D. study if he were not my supervisor.

Second, I would greatly like to thank Dr. Junichiro Mori, who continuously gave me opportunities to discuss research throughout my Ph.D. study, even when I was abroad. His comments significantly helped me in writing conference and journal papers and conducting my research. Without his precious support I would not have been able to finish this thesis.

My sincere thanks also goes to the rest of my thesis committee: Prof. Kazuro Kageyama, Prof. Hideaki Takeda, and Dr. Nariaki Nishino, for their comments and encouragement. Prof. Kazuro Kageyama gave me advice and support for my Ph.D. study, which motivated me to continue the study. His continual support throughout my bachelor, master, and Ph.D. years enabled me to continue research. Prof. Hideaki Takeda motivated me to study computer science during my undergraduate years, and the discussions with him gave me new insights into my studies. Dr. Nariaki Nishino also gave me insightful advice for the Ph.D. study in the field of technology management.

Furthermore, I would like to thank Dr. Takeshi Sakaki, who also helped me to continue my Ph.D. study. He also gave me opportunities to work with Hottolink, a social data provider, and this experience led to the first idea of this thesis. Without his support, it would not be possible to conduct research on social media.

I thank Dr. Masato Hagiwara, Ms. Kaoru Yamada, Dr. Javier Artiles, Prof. Satoshi Sekine, and Mr. Tomoyuki Tokugawa who provided me an opportunity to join Rakuten Institute of Technology New York as an intern, and helped with the research project on e-commerce data. In particular, Dr. Masato Hagiwara greatly supported the research project, which led to the core idea of this thesis. The precious experience to work with him led me to the field of natural language processing and my current career.

I thank my lab mates and friends for the stimulating discussions. I am grateful to Ms. Nozomi Nori for providing much assistance with the research project on social media. I thank Dr. Julia Anna Hollnagel and Mr. Masanao Ochi for the sleepless nights as we worked together before deadlines, encouraging each other, and for all the fun we had.

Last, but not least, I would also like to show my gratitude to my family. My father gave me insightful advice for research activities, and my mother supported me spiritually throughout my Ph.D. study, and my life in general.

# Publications

1. Junki Marui, Nozomi Nori, Takeshi Sakaki, and Junichiro Mori. Empirical Study of Conversational Community Using Linguistic Expression and Profile Information. In *Active Media Technology*, pp. 286–298. Springer International Publishing, 2014.

2. Junki Marui, and Masato Hagiwara. Category2Vec:単語・段落・カテゴリに対するベクトル分散表現. In 言語処理学会第 *21* 回年次大会発表論文集 *(NLP2015)*, pp.680–683, 2015.

3. Junki Marui, Masato Hagiwara, Takeshi Sakaki, and Junichiro Mori. Community Vector: 分散表現を用いたソーシャルメディアのコミュニティ推定. In *2015* 年度 人工知能学会全国大会 *(JSAI2015)*, 2015.

# References

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, Vol. 466, No. 7307, pp. 761–764, 2010.

[2] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web (WWW '12)*, pp. 519–528, 2012.

[3] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pp. 137–186. Springer, 2006.

[4] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250. ACM, 2001.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[6] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E.L.J.S. Mech. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, pp. 10008–10019, 2008.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. ACM.

[8] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 111–118, 2010.

[9] Danah Boyd and Kate Crawford. Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, communication & society*, Vol. 15, No. 5, pp. 662–679, 2012.

[10] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 20, No. 2, pp. 172–188, 2008.

[11] Leo Breiman. Random forests. *Machine learning*, Vol. 45, No. 1, pp. 5–32, 2001.

[12] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[13] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, Vol. 18, No. 4, pp. 467–479, 1992.

[14] John Bryden, Sebastian Funk, and Vincent A. A. Jansen. Word usage mirrors community structure in the online social network Twitter. *EPJ Data Science*, Vol. 2, No. 1, 2013.

[15] JohnA. Bullinaria and JosephP. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, Vol. 39, No. 3, pp. 510–526, 2007.

[16] Mingming Chen, Thin Nguyen, and Boleslaw K Szymanski. On measuring the quality of a network community structure. In *2013 ASE/IEEE International Conference on Social Computing (SocialCom 2013)*, pp. 122–127. IEEE, 2013.

[17] Fan RK Chung. *Spectral graph theory*, Vol. 92. American Mathematical Soc., 1997.

[18] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37. JMLR. org, 2015.

[19] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning(ICML-08)*, pp. 160–167. ACM, 2008.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, Vol. 20, No. 3, pp. 273–297, 1995.

[21] Niraj Dawar and Philip Parker. Marketing universals: Consumers' use of brand name, price, physical appearance, and retailer reputation as signals of product quality. *The Journal of Marketing*, pp. 81–95, 1994.

[22] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, Vol. 41, No. 6, pp. 391–407, 1990.

[23] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Phys. Rev. Lett.*, Vol. 94, p. 160202, Apr 2005.

[24] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159, 2011.

[25] Jack L. Engledow, Hans B. Thorelli, and Helmut Becker. The information seekers-a cross-cultural consumer elite. *Advances in Consumer Research*, Vol. 2, No. 1, pp. 141–156, 1975.

[26] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical review E*, Vol. 70, No. 5, p. 056104, 2004.

[27] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pp. 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[28] Hinton G.E., D.E. Rumelhart, and J.L. McClelland. *Distributed representations.*, Vol. 1: Foundations. MIT Press, 1986.

[29] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, Vol. 8, No. 1, pp. 41–48, 2006.

[30] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, Vol. 99, No. 12, pp. 7821–7826, 2002.

[31] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[32] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, Vol. 35, No. 12, pp. 61–70, 1992.

[33] Benjamin H Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, Vol. 81, No. 4, p. 046106, 2010.

[34] Mark S Granovetter. The strength of weak ties. *American journal of sociology*, pp. 1360–1380, 1973.

[35] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37. JMLR. org, 2015.

[36] Stephan Gunnemann, Ines Farber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *Proceeeindgs of the 10th IEEE International Conference on Data Mining*, pp. 845–850. IEEE, 2010.

[37] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, Vol. 53, No. 2, pp. 217–288, 2011.

[38] Zellig S Harris. Distributional structure. *Word*, Vol. 10, pp. 146–162, 1954.

[39] Moritz Karl Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 58–68. Association for Computational Linguistics, 2014.

[40] Donald Homa, Joseph Cross, Don Cornell, David Goldman, and Steven Shwartz. Prototype abstraction and classification of new instances as a function of number of instances defining the prototype. *Journal of Experimental Psychology*, Vol. 101, No. 1, p. 116, 1973.

[41] Donald Homa and Joan C Cultice. Role of feedback, category size, and stimulus distortion on the acquisition and utilization of ill-defined categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, Vol. 10, No. 1, p. 83, 1984.

[42] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613. ACM, 1998.

[43] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153. IEEE, 2009.

[44] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

[45] Mahesh Joshi, Ethan Hart, Mirko Vogel, and Jean-David Ruvini. Distributed word representations improve ner for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015)*, pp. 160–167, 2015.

[46] Slava M Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, Vol. 35, No. 3, pp. 400–401, 1987.

[47] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1746–1751. Association for Computational Linguistics, 2014.

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

[49] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pp. 1459–1474. The COLING 2012 Organizing Committee, 2012.

[50] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, Vol. 1, pp. 181–184. IEEE, 1995.

[51] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pp. 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[52] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*. American Association for Artificial Intelligence, 2015.

[53] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, Vol. 84, No. 6, p. 066122, 2011.

[54] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning(ICML-14)*, pp. 1188–1196, 2014.

[55] Alessandro Lenci. Distributional semantics in linguistic and cognitive research. *From context to meaning: Distributional models of the lexicon in linguistics and cognitive science, special issue of the Italian Journal of Linguistics*, Vol. 20, No. 1, pp. 1–31, 2008.

[56] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pp. 2177–2185. Curran Associates, Inc., 2014.

[57] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, Vol. 3, pp. 211–225, 2015.

[58] Kwan Hui Lim and Amitava Datta. Tweets beget propinquity: Detecting highly interactive communities on twitter using tweeting links. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '12, pp. 214–221. IEEE Computer Society, 2012.

[59] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute, 2011.

[60] Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. Structuring e-commerce inventory. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 805–814. Association for Computational Linguistics, 2012.

[61] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[62] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, Vol. 27, No. 1, pp. 415–444, 2001.

[63] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[65] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, Vol. 38, No. 11, pp. 39–41, 1995.

[66] John Paul Minda and J David Smith. Prototype models of categorization: Basic formulation, predictions, and limitations. *Formal approaches in categorization*, pp. 40–64, 2011.

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

[68] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pp. 195–204. ACM, 2000.

[69] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pp. 246–252. Citeseer, 2005.

[70] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, Vol. 69, No. 6, p. 066133, 2004.

[71] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, Vol. 103, No. 23, pp. 8577–8582, 2006.

[72] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 380–390, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[73] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empiricial Methods in Natural Language Processing (EMNLP 2014)*, Vol. 12, pp. 1532–1543, 2014.

[74] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 701–710, New York, NY, USA, 2014. ACM.

[75] Hieu Pham, Minh-Thang Luong, and Christopher D Manning. Learning distributed representations for multilingual text sequences. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015)*, pp. 88–94, 2015.

[76] Martin F Porter. An algorithm for suffix stripping. *Program*, Vol. 14, No. 3, pp. 130–137, 1980.

[77] J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[78] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[79] Philip Resnik. Wordnet and distributional analysis: A class-based approach to lexical discovery. In *AAAI workshop on statistically-based natural language processing techniques*, pp. 56–64, 1992.

[80] Tiago Rodrigues, Fabrício Benevenuto, Meeyoung Cha, Krishna Gummadi, and Virgílio Almeida. On word-of-mouth based discovery of the web. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 381–396. ACM, 2011.

[81] Daniel M. Romero, Chenhao Tan, and Johan Ugander. On the interplay between social and topical structure. In *Proceedings of the Seventh International Conference on Weblogs and Social Media (ICWSM-13)*, 2013.

[82] Eleanor Rosch. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, Vol. 104, No. 3, p. 192, 1975.

[83] Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, Vol. 8, No. 3, pp. 382–439, 1976.

[84] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[85] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[86] Cosma Rohilla Shalizi and Andrew C Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods and Research*, Vol. 17, pp. 211–239, 2011.

[87] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 595–604. ACM, 2012.

[88] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 129–136, 2011.

[89] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pp. 151–161. Association for Computational Linguistics, 2011.

[90] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pp. 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[91] Jiliang Tang, Xufei Wang, and Huan Liu. Integrating social media data for community detection. In *Modeling and Mining Ubiquitous Social Media*, pp. 1–20. Springer, 2012.

[92] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1591–1601, Doha, Qatar, October 2014. Association for Computational Linguistics.

[93] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, Vol. 8. Cambridge university press, 1994.

[94] Andreas S Weigend, Erik D Wiener, and Jan O Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, Vol. 1, No. 3, pp. 193–216, 1999.

[95] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. Convolutional neural networks for text hashing. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 147–153, 2015.

[96] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. Short text clustering via convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics  Human Language Technologies (NAACL-HLT 2015)*, pp. 62–69, 2015.

[97] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37. JMLR. org, 2015.

[98] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 545–550, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[99] Xun Yuan, Wei Lai, Tao Mei, Xian-Sheng Hua, Xiu-Qing Wu, and Shipeng Li. Automatic video genre categorization using hierarchical svm. In *Proceedings of the 2006 IEEE International Conference on Image Processing*, pp. 2905–2908. IEEE, 2006.

[100] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pp. 452–473, 1977.

[101] Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 111–121. Association for Computational Linguistics, 2014.

[102] Xin Zhao, Yuexin Wu, Hongfei Yan, and Xiaoming Li. Group based self training for e-commerce product record linkage. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics:*

*Technical Papers*, pp. 1311–1321. Dublin City University and Association for Computational Linguistics, 2014.

[103] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, Vol. 2, No. 1, pp. 718–729, 2009.

[104] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*, pp. 22–32. ACM, 2005.

[105] G. K. Zipf. *Human Behaviour and the Principle of Least-Effort.* Addison-Wesley, Cambridge MA, 1949.