FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automatische Kategorisierung von e-Commerce Produkten durch Multimodale Klassifikation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Alper Kirim, BSc
Matrikelnummer 0927700

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Wien, 8. Jänner 2019

Alper Kirim	Andreas Rauber

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Automatic Classification of e-Commerce Products via Multimodal Classification

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Alper Kirim, BSc
Registration Number 0927700

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Vienna, 8th January, 2019

Alper Kirim	Andreas Rauber

Erklärung zur Verfassung der Arbeit

Alper Kirim, BSc
Hermann-Bahr-Straße 18 1210 Vienna

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. Jänner 2019

Alper Kirim

Acknowledgements

First and foremost, I would like to thank my thesis advisor Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber. Throughout the course of writing this thesis, his thoughtful comments helped me immensely. Most importantly, I would like to thank him for developing my appreciation for the scientific research.

I would like to thank my girlfriend Hannah for her continuous support and patience. My life would be empty, just like the pages of this thesis would have been, if it wasn't for her.

I would like to thank my parents for supporting me morally and financially throughout my education. I would like to thank my cousin Hande for being the best cousin possible, for opening my eyes to fantasy worlds and for proofreading this thesis. I would also like to thank my uncle Sedat for the moral support.

Last but not least, I would like to thank my friends and former colleagues Paolo and Patrick for their helpful comments on the various drafts of this thesis and their moral support.

Kurzfassung

Das Organisieren von Produkten in die Produktkategorien ist ein wesentlicher Bestandteil der E-Commerce-Anwendungen. Allerdings in einigen Second-Hand-Marktplatzanwendungen, bei denen Benutzer keine professionellen Verkäufer sind, wird die Eingabe einer Kategorie optional gehalten, um die Erstellung der Inserate zu vereinfachen. Diese Elemente müssen dann von einem internen Team kategorisiert werden, damit sie gefunden werden können, wenn Suchergebnisse nach Kategorien gefiltert werden. Das Ziel dieser Arbeit ist es, das Potenzial eines multimodalen Klassifikators als Alternative zu der manuellen Kategorisierung, die von einem internen Team eines Unternehmens im Second-Hand-Marktplatz-Geschäft betrieben wird, zu evaluieren. Da einige nicht-professionelle Benutzer Gegenstände ohne relevanten Text oder mit Bildern von schlechter Qualität auflisten, wird ein multimodaler Ansatz gewählt, um das System robuster zu machen.

Zunächst wird die Leistung des Inhouse-Teams untersucht. Nach der Analyse und Vorverarbeitung der verfügbaren Daten wird ein multimodaler Klassifikator entwickelt und ausgewertet, der die Leistung des internen Teams erreichte. Neben der Entwicklung und Evaluierung eines Klassifikators werden mögliche Einsatz- und Überwachungsstrategien diskutiert.

Abstract

Organising products into product categories is an integral part of e-commerce applications. However in user-driven secondhand marketplace applications, where users are not professional sellers, input of a category is kept optional to make the listing process easier. These items must then be categorized by an in-house team so that they can be found when search results are filtered by categories. The goal of this thesis is to investigate the potential of a multimodal classifier as an alternative to the manual categorization done by an in-house team at a company in the secondhand marketplace business. Since some non-professional users tend to list items without descriptive text or with bad quality images, a multimodal approach is chosen to make the system more robust.

Initially performance of the in-house team is evaluated. After analyzing and preprocessing the available data, a multimodal classifier is developed and evaluated which reaches the performance of the in-house team. In addition to the development and evaluation of a classifier, possible deployment and monitoring strategies are discussed.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Categories in e-commerce websites provide a concise way of filtering provided items contextually. A special case of e-commerce websites is online marketplaces for secondhand items. Companies in this business offer websites and mobile apps, which, among other features, let users create an item entry with images and text (mostly a title and a description) and also pick a category during the process of posting an item entry. To make this process easier, picking of a category is mostly kept optional. As a result uncategorized product entries must either be manually categorized by humans or automatically categorized using machine learning techniques.

There are various challenges in this categorization task. Depending on the granularity of the category structure, there can be overlapping categories, which makes evaluation of models more difficult. Additionally text data is noisy, because the users are non-professional sellers. There are also different success criteria for such categorization tasks depending on whether the resulting model will be deployed to suggest users a category during the listing process or to help with subsequent categorization of uncategorized items by expert editors.

Multimodal classification is particularly suited for item categorization because there are items where users provide a descriptive title and description but a bad quality image and vice versa.

With the recent success of Deep Neural Networks (DNN) in image categorization tasks [He+16a] [KSH12] [Sze+15], the question arises as to whether it is possible to design a multimodal classifier using a DNN as image classifier to improve accuracy in item classification tasks.

1.2 Problem Statement & Research Questions

Categorization of the uncategorized products in a company in secondhand marketplace business is currently done manually by the user support team. As a result of this manual categorization the available workforce for customer support tickets is decreased.

Advantages of a multimodal classifier architecture with a pretrained neural network for image classifier component will be investigated to tackle this problem. The multimodal classifier will use text and image features for the classification. For the image component of the multimodal classifier advantages and disadvantages of transfer learning will be evaluated.

The aim of the research is to answer the overarching research question, "To what extent can a multimodal DNN-based classifier for classification of e-commerce products be deployed to reduce the manual effort of classifying?". To answer this question we need to break it down to the following specific questions:

- **RQ1:** How unique are the categories? If there are overlapping ones how do we deal with them?
- **RQ2:** Which preprocessing techniques are useful against noisy text data caused by input from mobile devices?
- **RQ3:** How can performance of the classifier be measured for a dataset with overlapping classes? What kind of evaluation is useful for the given problem setting?
- **RQ4:** Are there significant benefits of retraining the last layer or more than one layer of a model pretrained on ImageNet [Den+09] classes instead of using predictions for ImageNet classes?
- **RQ5:** Which fusion method for the image and text modalities is better for the given problem setting?
- **RQ6:** What is the best function to determine uncertainty of the multimodal classifier using probability outputs for the given problem setting?

1.3 Methodology and Structure

CRISP-DM [Cha+00] (Cross Industry Standard Process for Data Mining) will be the guiding model for tasks that are needed to be performed during the development of the classifier.

Before going into CRISP-DM related tasks, a theoretical background will be given about data science projects in general and also more specialized topics that are related to this thesis in chapter 2. After giving a brief overview about the CRISP-DM model,

state-of-the-art in item classification in e-commerce will be summarized. Furthermore supervised learning methods and evaluation metrics that will be used in this thesis will be explained.

According to the CRISP-DM model, the first phase in the lifecycle of a data mining project is business understanding. In chapter 3, business objectives and requirements will be analyzed. After the business understanding phase, quality of the data sources needs to be investigated. There are two possible data sources: items classified by the support team and items with a provided category by the user. An experiment will be performed to determine which data source has a higher quality. 5 support team members will be asked to count the misclassified items in a list that contains equal amount of items from both data sources. After the data source quality analysis, the data will be collected from company's database. In addition to textual data collected from company's data warehouse, scripts will be written to automate downloading of images from media servers. Features will be derived from collected data. Following that, data will be analyzed and preprocessed, which corresponds to the data preparation phase of CRISP-DM. Following the analysis and preprocessing of the data, fitting evaluation metrics for the project will be discussed.

In chapter 4, in which the modeling phase will be handled, various models will be built and evaluated for each component of the multimodal architecture according to the defined evaluation metrics. First, the text classifier will be developed. After evaluating various input representations and preprocessing techniques, transfer learning for the image classifier will be investigated thoroughly. A state-of-the-art image classifier will be retrained while varying numbers of layers from the start of the network are frozen during training. An image classifier without retraining will be also evaluated. After the preparation of components for text and image modalities, early and late fusion methods will be compared for the given problem setting. Furthermore, because of overlapping categories and high precision expectancy, uncertainty functions will be evaluated as a solution. These functions will determine when the classifier's prediction should be accepted. Two uncertainty functions will be constructed and compared against each other. Finally configuration of the best performing multimodal architecture will be described and evaluated on the test set.

In the final chapter, research questions that were posed in the first chapter and corresponding answers acquired during the subsequent chapters will be summarized. In addition to these answer, improvements and questions which were beyond the scope of this thesis will be mentioned as potential future work.

CHAPTER 2

Related Work

In this section the guiding methodology CRISP-DM framework and techniques for image and text categorization that will be used for the implementation of the multimodal classifier are explained. Special emphasis is given to deep neural networks, since the main goal of this thesis is to improve classification accuracy with the usage of a deep neural network for images. Additionally a brief overview of state-of-the-art in product classification in e-commerce is given.

2.1 CRISP-DM Framework

CRISP-DM is a process which guides data mining projects by defining tasks that are needed to be performed. These tasks are described in four levels of abstraction from general to specific: phase, generic task, specialized task, process instance. General tasks can be mapped to specific ones according to the project. Details can be removed or added depending on project's needs.

Since data mining is an iterative process, phases of CRISP-DM have a cyclical nature (figure 2.1). Obtained results from one task can make it necessary to repeat a task from a previous phase. Six phases in the CRISP-DM reference model are as follows:

Business Understanding In this phase objectives and requirements of the project are defined according to business perspective. Building on these definitions a data mining problem definition is created. A preliminary plan is made according to the data mining problem definition to accomplish the objectives.

Data Understanding Tasks in this phase ensure that the data is collected and quality of it is assessed. After collecting, acquired data is described before the initial exploration of it. After understanding the structure of the data, the quality of the data is verified, since data can have missing values or contain errors.

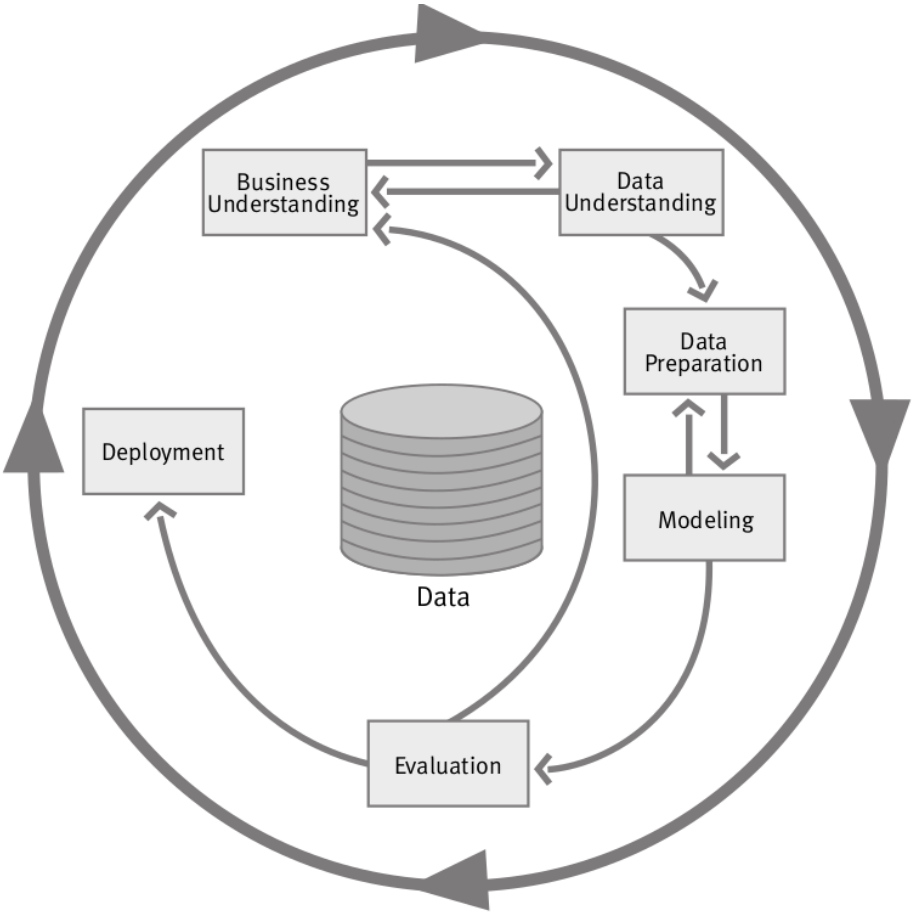

Figure 2.1: Phases of CRISP-DM Reference Model

Data Preparation In this phase the data is brought to a form, with which models should be trained. Firstly, data attributes and rows are selected. This data is then cleaned. The cleaning process aims to raise the data quality. Clean subset selection and usage of default values for missing values are some of the techniques that could be applied for this general task.

Modeling This phase includes selection and building of the model to be applied to data. Before building the model, test criteria are defined and after building it these criteria are assessed.

Evaluation While the evaluation in the modeling phase focused on the model itself, in the evaluation phase the business objectives are assessed.

Deployment This phase includes deployment, monitoring and maintenance of the model. A final report is generated which summarizes the results obtained in the previous

phases. Finally the project is reviewed to document the experience gained during the project.

2.2 Item Classification in e-Commerce

Item classification is a critical task for e-commerce platforms. As this thesis investigates the advantages of using a deep neural network-based image classifier in a multimodal classifier for product classification in an e-commerce company, a brief overview of state-of-the-art for this task is given here. Research teams of big e-commerce companies investigated various methods to solve this task. Many researchers tried to solve it using text classification techniques. There were also multimodal learning approaches combining text and image classifiers.

Kozareva [Koz15] compared four different feature extraction techniques for classifying products using their titles into 319 categories with 6 levels with a linear classifier. Categorization was done using title of products on Yahoo!'s shopping platform. Best result was achieved using word embeddings as input, reaching 0.88 f-score.

Shen, Ruvini, and Sarwar [SRS12] proposed a system with two phases for large scale text categorization problems. The proposed system uses only title to categorize products in E-bay's category hierarchy, which has more than 20000 leaf categories. First phase uses a simple but scalable classifier which maps products to latent groups. In the second phase a set of more complex models for finer classification are used to classify the product in the latent group chosen by the classifier from the first phase. The aim of this approach is to reduce the class space by first classifying products to latent groups. Authors were able to improve precision from 72.2% of the hierarchical classifier leveraging eBay category structure to 75.4% of their two-level classifier.

Sun, Rampalli, Yang, and Doan [Sun+14] developed a system for product classification called Chimera, which combines hand-crafted rules, machine learning, and crowdsourcing. Authors used only title of products to classify them into over 5000 product types. Initial version of the system used only learning-based classifiers and could only achieve 50% precision and 50% recall. With the addition of hand-crafted rules it could reach 93% precision for 90% coverage on 2.5 million products. Authors argued that crowdsourcing can be effectively used to evaluate the performance of the system coupled with in-house analysts and developers.

Kannan, Talukdar, Rasiwasia, and Ke [Kan+11] combined a text classifier with an image classifier in a model called Confusion Driven Probabilistic Fusion++ (CPDF++). Addition of the image predictions led to 12% improvement in accuracy in comparison to text-only classifier. Initially training starts with only the text classifier. During training 3-way image classifiers are generated according to the confusion matrix of the text classifier. 3-way image classifiers are between two confusing categories and a background category which represents all other categories. Amount of 3-way image classifiers is a hyperparameter of this model.

Although these approaches are promising for complex category structures, they don't deal with overlapping categories since all of the category structures in these papers have a fine granularity. In spite of this difference, some techniques from these approaches will be integrated in the multimodal classifier in this thesis. Word embeddings will be evaluated as a potential way to deal with noisy text data. Additionally an image classifier will be combined with the text classifier to improve the performance. Performance of the classifier will be further improved by using an uncertainty function combining in-house analysts with machine learning. The uncertainty function will determine whether the classifier is confident about it's decision. The classifier will leave hard-to-classify products to the user support team. None of the results from the mentioned papers in this section can be made comparable to the results of this thesis, because none of the mentioned papers use a publicly available dataset.

2.3 Supervised Learning

Supervised learning is a field of machine learning where a function is learned for given inputs and outputs. In this chapter supervised learning techniques that are used throughout this thesis will be explained.

2.3.1 Evaluation metrics

There are various metrics to evaluate the performance of a classifier. All metrics that are used in this thesis are defined using the terms in the table 2.1.

Table 2.1: Contingency table for evaluation metric building blocks

	Prediction positive	Prediction negative
Actual class positive	True Positive (TP)	False Negative (FN)
Actual class negative	False Positive (FP)	True Negative (TN)

A common metric is accuracy, which shows how many of the predictions were correct:

$$Accuracy = \frac{TP + TN}{\text{TP+TN+FP+FN}} \tag{2.1}$$

The complementary metric to accuracy is error rate, which shows how many of the predictions were false:

$$Error\ rate = 1 - Accuracy \tag{2.2}$$

Although accuracy is a good measure for balanced datasets, it might be misleading when dealing with imbalanced datasets. If a large amount of instances belong to class A, accuracy will be still very high if the classifier can predict instances of class A correctly but cannot predict most of the other classes properly. To circumvent this problem, two extra measures, precision and recall averaged over classes, can be used. Precision and

recall of a class x are defined as follows:

$$Precision(x) = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall(x) = \frac{TP}{TP + FN} \tag{2.4}$$

Precision for a class x shows how many of the the positive predictions of x were correct. Recall for a class x shows how many of the instances of class x were predicted correctly.

F1-score is a harmonic mean of precision and recall:

$$F_1(x) = 2 \cdot \frac{Precision(x) \cdot Recall(x)}{Precision(x) + Recall(x)} \tag{2.5}$$

Since precision, recall and f1-score are metrics for binary classification tasks, they need to be averaged over classes to obtain a single measure for multiclass classification problems. These averages can be calculated in two ways:

- **Macro:** Scores for each class are averaged without weighting.
- **Micro:** Scores for each class are averaged weighted by their sample count.

In some cases condition for trueness of a prediction can be relaxed. Top-5 accuracy, for example, accepts a prediction as correct if the instance class is in the top 5 predictions for that instance.

2.3.2 Deep Neural Networks

Neural networks are computational models that consist of connected artificial neurons. These basic building stones of neural networks are very loosely based on biological neurons. Simplest neural network is the perceptron [Ros58] (figure 2.2), which is only one artificial neuron.

It has inputs x_i, weights for these inputs w_i and a bias b [Roj13]. Perceptron calculates a function $f(x)$, which multiplies weights with the corresponding inputs and sums up the results with the bias.

$$f(x) = \sum_{i=0}^{n} w_i x_i + b. \tag{2.6}$$

Result of $f(x)$ is fed to an activation function a for classification. A basic example for an activation function for binary classification is the heaviside step function.

$$a(f(x)) = \begin{cases} 0, & \text{if } f(x) < 0 \\ 1, & \text{if } f(x) \geq 0 \end{cases} \tag{2.7}$$

For correct classification, the weights of the perceptron need to be set properly. One method for finding them is to calculate the error of the network for a given input and

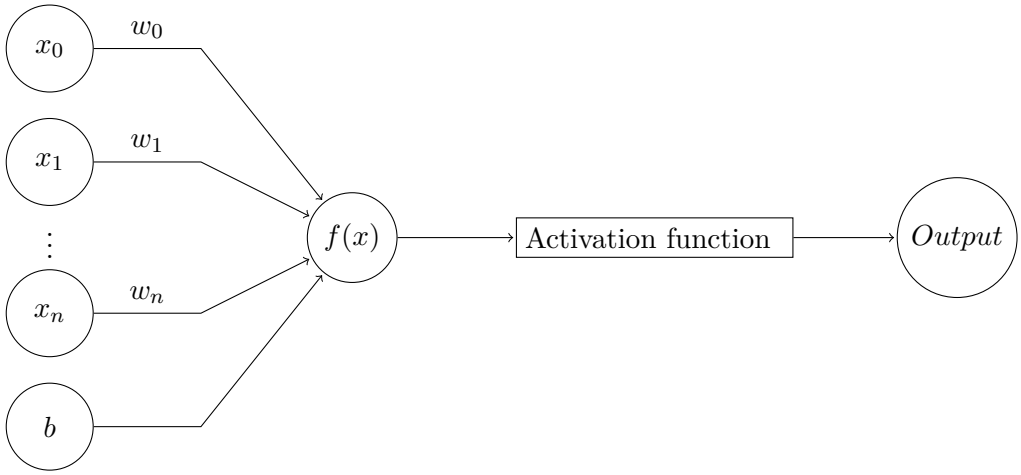

Figure 2.2: Perceptron

adjust the weights according to that error. Error is defined as the difference between the correct output T and predicted output of perceptron O.

$$Err = T - O \tag{2.8}$$

To minimize the error, value of O must be increased or decreased according to input. This is achieved through a learning rule (2.9). The magnitude of weight updates for each example is controlled by the learning rate α.

$$w_i^{t+1} = w_i^t + \alpha \cdot x_i \cdot Err \tag{2.9}$$

The perceptron is a linear classifier, which means that it can only represent linearly separable functions. In other words, the perceptron can only divide an input space in two along a boundary [RN03]. For data that is not linearly separable, multi-layer feed-forward networks can be used. MLPs contain one or more hidden layers, which contain one or more perceptrons. This allows MLPs to represent more complex functions. Weights of MLPs can be trained using an algorithm called back-propagation, which is a generalized version of perceptron training. This algorithm makes use of the gradient of the error function to calculate a weight update which lowers the error value the most. Since MLPs contain one or more hidden layers, learning rule needs small adjustments for the output layer.

$$w_{j,i} = w_{j,i} + \alpha \cdot a_j \cdot Err_i \cdot g'(in_i) \tag{2.10}$$

Where in_i is the result of applying the function f (defined in equation 2.6) to the output of the previous layer, a_j is the output of neuron j after applying the activation function, Err_i is the error of the neuron i, and g' is the derivative of the activation function.

For connections between other layers a new error term needs to be defined. The main idea is that each neuron is responsible for some fraction of the error $\Delta_i = Err_i \cdot g'(in_i)$ in each of the output nodes to which it connects. The Δ_i values are divided according to the strength of the connection between the neuron in the hidden layer and the neuron in the output layer. These values are then propagated back to calculate Δ_j values for the neuron in the hidden layer. The propagation rule for the Δ values is defined as follows:

$$\Delta_j = g'(in_j) \sum_i w_{j,i} \Delta_i \tag{2.11}$$

Algorithm 2.1: Backpropagation algorithm

Input: A multi-layer network *network* , a set of input/output pairs *examples* and a learning rate α

repeat
  foreach *e in examples* **do**
    `// Compute the output for this example`
    $O \leftarrow$ FORWARD-PROPAGATION(network, I^e)
    `// Compute the error`
    $Err^e \leftarrow T^e - O$
    `// Update the weights leading to the output layer`
    $\Delta_i \leftarrow Err_i^e \cdot g'(in_i)$
    $w_{j,i} \leftarrow w_{j,i} + \alpha \cdot a_j \cdot \Delta_i$
    foreach *subsequent layer in network* **do**
      `// Compute the error at each node`
      $\Delta_j \leftarrow g'(in_j) \sum_i w_{j,i} \Delta_i$
      `// Update the weights leading into the layer`
      $w_{k,j} \leftarrow w_{k,j} + \alpha \cdot a_k \cdot \Delta_j$
    end
  end
until *stopping criteria is fulfilled*;
return *network*

Although backpropagation algorithm works well for moderately deep MLPs, after a certain amount of depth, training the first layers gets very slow. This is called the vanishing gradient problem. The problem is caused by the gradient, which becomes smaller as the error is backpropagated to the initial layers of the network. To mitigate this problem a popular activation function for neural networks called rectified linear activation unit (ReLU) can be used [LBH15]. For an input x, ReLU is defined as follows:

$$a(x) = max(0, x) \tag{2.12}$$

Another commonly used activation function is called softmax. Softmax function is often used as the output of a classifier, to represent probability distribution over n different

classes [GBC16]. For an input x_i, where i denotes the output of a unit out of N units in a layer, it is defined as follows:

$$softmax(x_i) = \frac{e^{x_i}}{\Sigma_{j=1}^{N} e^{x_j}} \tag{2.13}$$

Because the weight updates are done for every input, and gradient of the error function is used, this algorithm is also called gradient descent algorithm. Rumelhart, Hinton, and Williams [RHW+88] suggested to use the complete input dataset for calculating the gradient. However this approach is not efficient for big datasets. To solve this issue, mini-batch gradient descent is used, which calculates weight updates using a subset of the dataset as input.

There are various optimization algorithms to accelerate learning with gradient descent. Momentum algorithm adds an extra momentum term for the weight update:

$$\begin{aligned} v_t &= \gamma v_{t-1} + (\alpha \cdot a_j \cdot Err_i \cdot g'(in_i)) \\ w_{j,i} &= w_{j,i} - v_t \end{aligned} \tag{2.14}$$

where γ is called momentum term and is a hyperparameter of the algorithm. Momentum term acts like momentum in physics. Solution space has hills and ravines with different steepness. If gradient is large (in other words a steep slope in solution space), weight updates get bigger. Momentum algorithm was used to train weights of residual neural networks [He+16a], which will be used as a component for the image modality in the multimodal classifier.

For the retraining of the image classifier for the multimodal classifier, Adam algorithm will be used because it requires less tuning of hyperparameters. Adam algorithm [KB15] computes individual learning rates for each parameter using estimates of first and second moment of gradients. First and second moments are calculated as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned} \tag{2.15}$$

where g_t is the gradients at timestep t, m_t is the estimate of the first moment of the gradients, v_t is the estimate of the second moment of the gradients and β_1 and β_2 are hyperparameters that control the exponential decay of the moving averages. The moving averages are initialized as vectors of 0s and because of that they are biased towards 0, especially in the beginning of optimization and when β values are close to 1. To alleviate the bias, bias-corrected versions of m_t and v_t are used that are calculated as follows:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \tag{2.16}$$

These values are then used for the update rule for the weights:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.17}$$

Convolutional Neural Networks

A specialized type of neural network are convolutional neural networks. These networks are very efficient for inputs that have a grid-like topology. Each layer of an MLP multiplies an input vector with the matrix that contains weights of connections between the current and the previous layer to calculate the weighted sum of the inputs which are then passed through an activation function to calculate the output. A neural network is considered to be a CNN if at least one of the layers uses convolution operation instead of general matrix multiplication. Instead of multiplying the whole input with a single big weight matrix, input is transformed by convolving it over filters (also called kernels), which are weight matrices that are smaller than the input size. Output of this stage goes through an activation function as in MLP.

Convolution is a mathematical operation, which can be seen as a product of two functions. Although convolution is formally defined for continuous input, in machine learning applications the discrete form is used. Additionally most neural network libraries use cross-correlation instead of convolution, which is a related function of convolution [GBC16].

For a two-dimensional input, cross-correlation function is defined as follows:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \tag{2.18}$$

where I is the input and K is the filter (Fig. 2.3). For the rest of this thesis, cross-correlation function will be referred to as convolution.

Convolutional layers have advantageous properties. First of all, they have sparse interaction, meaning that outputs of a single neuron are not connected to all neurons in the next layer, which reduces the required amount of parameters (Fig. 2.4). Secondly, parameters of filters are shared. Since filters are smaller than the input and are applied to the whole image by sliding the filter, they can be used to detect many low-level features in the initial layers, like edges in an image. Lastly, convolutional layers are equivariant to translation, meaning they are able to detect identical features even if positions of these features (for example in an image) are changed.

Outputs of the activation functions can further be manipulated by pooling layers, which replace these values with a summary statistic of the neighbouring outputs. Two common types of pooling layers are the following:

- Max pooling: outputs maximum of values in a rectangular neighbourhood

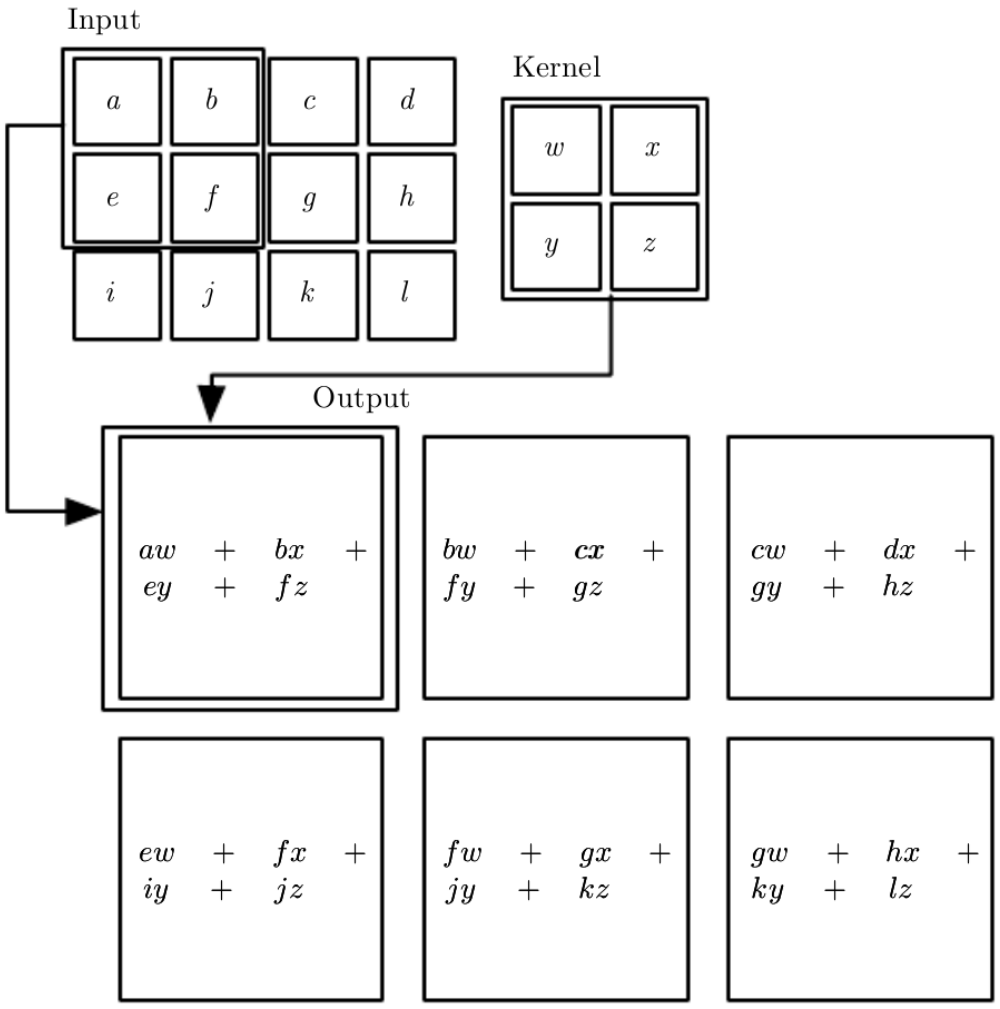

Figure 2.3: Convolution with a two-dimensional kernel (from [GBC16])

- Average pooling: outputs average of values in a rectangular neighbourhood

Filters of pooling layers or convolutional layers can move more than one pixel during calculation of the output. The size of that move is called stride. Max pooling with stride of 2 is shown in figure 2.5.

Approaches to Image Recognition with Deep Neural Networks

Since 2012 interest in deep neural networks has been growing rapidly thanks to major improvements in image recognition task results. In this chapter a brief history of image recognition with deep neural networks is given.

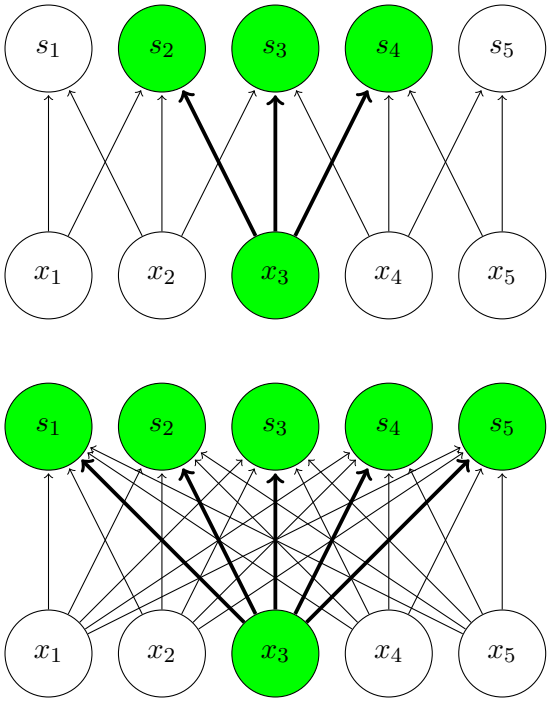

Figure 2.4: Sparse interaction. Input x_3 and the corresponding connections have been marked. Convolutional layer (top) has less connections than a fully connected layer like in MLP (bottom)

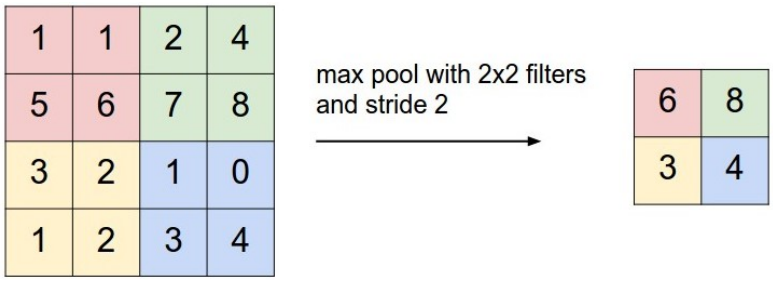

Figure 2.5: Max pooling applied to a 4×4 matrix with stride of 2. (from [Kar])

The basic inspiration for Convolutional Neural Networks comes from neuroscience. Neocognitron, proposed by Fukushima and Miyake [FM82], was the first model architecture inspired by the mammalian visual system. Major advantage of Neocognitron was that it was not affected by the position shifts and the distortion in shape of the input patterns.

LeCun, Bottou, Bengio, and Haffner [LeC+01] applied CNNs to classify handwritten digits in checks with 5% test error. This was one of the first successful applications of CNNs in a production environment.

In 2012 interest in CNNs has risen because of a CNN submitted by Krizhevsky, Sutskever,

and Hinton [KSH12] for the ImageNet ILSVRC challenge which reduced top-5 error enormously. Their submission had 16% top-5 error which was 10% lower than the runner-up. From this point on ImageNet ILSVRC challenge has been won by different CNN-based submissions every year.

Zeiler and Fergus [ZF14] improved the convolutional network proposed by Krizhevsky, Sutskever, and Hinton and reached first place in ILSVRC 2013. Their main contribution was a new way of visualizing activations inside a convolutional neural network. Their findings suggested that the features learned by CNNs are not uninterpretable patterns, which makes it possible to investigate problems within the model. Additionally, they have shown that features learned by CNNs have good properties like compositionality, increasing invariance and class discrimination as activations reach deeper layers.

Oquab, Bottou, Laptev, and Sivic [Oqu+14] showed that CNNs that have been trained on an image dataset can be successfully applied to a different dataset by only updating weights in the last layers of the model. This approach is called transfer learning.

Yosinski, Clune, Bengio, and Lipson [Yos+14] explored limits of transfer learning by inspecting learned features by neurons at different layers in the network. They show two main issues that negatively affect performance of the retrained network. First issue is co-adapted layers, that interact with each other in a complex way to detect a feature, which makes it hard for upper layers to relearn such features when only some of these layers are transferred. Second issue is the specialization of higher layer features, which becomes a dominant issue when the dataset for retraining is particularly different from the dataset on which the base model was trained on. Additionally they showed that using transferred weights for initial layers and fine-tuning them improved generalization of the model to the new task better than using random weights and training the model completely.

Ioffe and Szegedy [IS15] proposed a method called batch-normalization which improved training speed and overall accuracy of the trained model. The main problem which batch-normalization is solving is called internal covariate shift. Internal covariate shift refers to the fact that a small change in weights in an initial layer is amplified in the successive layers into a big change. In other words, a small change in an initial layer results in a big change for inputs of layers at the end of the network. To remedy this issue, at certain points of the network, activations of layers for a mini-batch are normalized to mean of 0 and variance of 1. To ensure that the representational power of a layer does not change because of the normalization, two parameters γ and β are introduced to scale and shift the normalized value. These parameters are learned during training like the weights of the network. For output of a single neuron x_i, where i denotes the sample in a mini-batch $\mathcal{B}$, this transformation is defined as follows:

$$\hat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2.19}$$

$$y_i \leftarrow \gamma \hat{x_i} + \beta \tag{2.20}$$

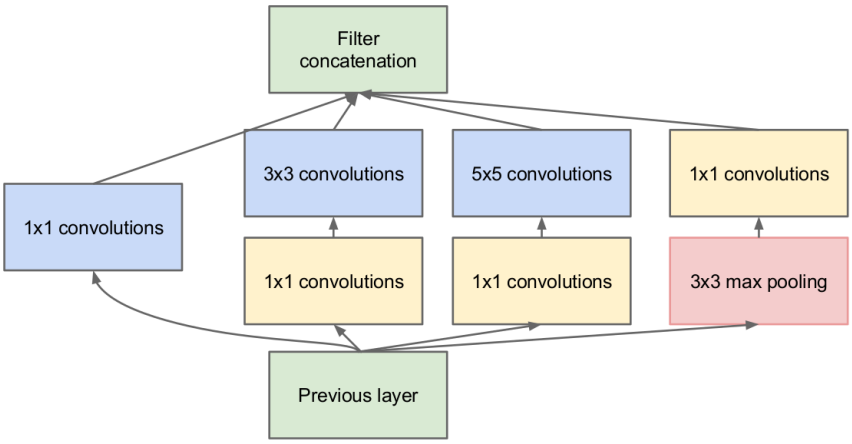

Figure 2.6: Inception module (from [Sze+15])

where y_i is the transformed input, $\mu_{\mathcal{B}}$ is the mean of values in the mini-batch, $\sigma^2_{\mathcal{B}}$ is the variance of values in the mini-batch and ϵ is a constant to ensure numerical stability. This transformation ensures that changes in initial layer don't have huge impact for the layers at the end of the network.

Simonyan and Zisserman [SZ14] evaluated networks of increasing depth with CNNs that use small 3×3 convolution filters. Their results showed that accuracy increases with depth. Using their results they could develop a model, which won the localization task of ImageNet Challenge 2014 and was runner-up for the classification task.

Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke, and Rabinovich [Sze+15] developed a module called inception, which reduces number of parameters drastically. The inception module consist of $1 \times 1, 3 \times 3, 5 \times 5$ convolutions and a 3×3 max pooling layer (figure 2.6). This module finds an optimal sparse structure. The module contains also 1×1 convolutions before 3×3 and 5×5 convolutions and after 3×3 max pooling to reduce dimension to keep computational requirements low. Concretely, given an input for the inception module with size $28 \times 28 \times 192$ and convolutions with stride 1, a 3×3 convolution needs 9 times and a 5×5 convolution needs 25 times more computation for each filter then a 1×1 convolution. By using 1×1 convolutions with lower number of filters the input size for a convolution with 3×3 or 5×5 is reduced (e.g. to $28 \times 28 \times 64$ instead of $28 \times 28 \times 192$), which significantly reduces the amount of computation that is needed. The inception model is a network of inception modules stacked on top of each other. The network contains two auxiliary classifiers, that are connected to intermediate layers of the network, to combat the vanishing gradient problem. Gradients of the auxiliary classifiers are propagated back in a more focused fashion, since they have to go through less layers.

He, Zhang, Ren, and Sun [He+16a] proposed residual building blocks to combat vanishing gradient problem and won ILSVRC 2015 classification competition. These blocks contain

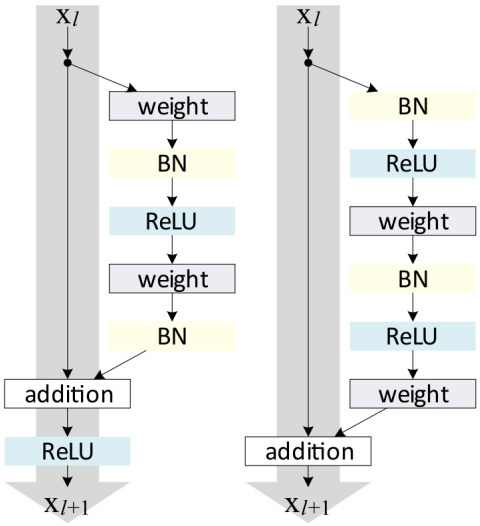

Figure 2.7: Originally proposed residual block (left) and improved residual block (right) (from [He+16b])

an identity function, which adds input of the building block to the output. During back-propagation, gradients have shortcuts so that they can efficiently reach initial layers without auxiliary classifiers. This allowed authors to train a network with 1001 layers. In a following paper, an improved version of the residual block has been proposed, where the identity function does not go through a rectified linear activation unit [He+16b] (figure 2.7).

2.3.3 Naive Bayes

Naive Bayes algorithm is a probabilistic approach for classification, which makes use of Bayes' theorem. Bayes' theorem is used to calculate posterior probability $P(x|y)$, which is defined as probability of x, given y. For example, for two consecutive coin tosses y can be the probability of the first coin landing face up and x can be the probability of the second coin landing face up. The posterior probability can be calculated as follows:

$$P(x|y) = \frac{P(y|x) \cdot P(x)}{P(y)} \tag{2.21}$$

In this thesis Naive Bayes algorithm will be evaluated for text classification. Every feature is assumed to be independent of each other. Although the independence assumption is unrealistic, Naive Bayes classifiers are highly effective in practice [RHT01]. Given the features a_1, a_2,..., a_d posterior probability of class c_i can be calculated with:

$$P(c_i \mid a_1, \ldots, a_d) = \frac{P(c_i) \cdot \prod_{j=1}^{d} P(a_j \mid c_i)}{P(a_1, \ldots, a_d)} \tag{2.22}$$

Since the denominator $P(a_1, \ldots, a_d)$ is constant for each class, it can be removed for classification purposes. This simplifies the equation to the following:

$$P(c_i \mid a_1, \ldots, a_d) \propto P(c_i) \cdot \prod_{j=1}^{d} P(a_j \mid c_i) \tag{2.23}$$

Naive Bayes classifiers pick the class with the highest posterior probability for the given values for the features.

The independence assumption has an undesired side effect. Given a sample with d features $a_1, a_2, ..., a_d$, if one of these features a_j does not appear in samples of class c_i in the training set, $P(a_j|c_i)$ becomes 0 which cancels out all other features for that sample. To avoid having probability of 0, laplace smoothing can be used for categorical data. Laplace smoothing is performed for each posterior probability with the following formula:

$$\hat{\theta}_{ji} = \frac{N_{ji} + 1}{N_i + d} \qquad (j = 1, \ldots, d, i = 1, \ldots, n) \tag{2.24}$$

where $\hat{\theta}_{ji}$ is the smoothed variant of $P(a_j|c_i)$, N_{ji} is the number of times feature a_j appears in a sample of class c_i and N_i is the total count of all features for class c_i [Sci].

2.3.4 Support Vector Machine

Another technique that will be evaluated for the text classifier is the support vector machine. SVMs separate feature space with a hyperplane optimally. The separation is optimal if the hyperplane separating the data has the maximum margin to the closest data points for two classes it separates (Fig. 2.8).

Core definition of support vector machines allows only binary classification. It is, however, possible to apply support vector machines to multiclass classification problems by transforming such problems to multiple binary classification problems using one vs rest strategy. For a dataset with n classes, n classifiers are trained, where each one calculates if the input belongs to one specific class or not. The ensemble classifier output is determined by looking at the results of all binary classifiers and selecting the class with the highest score. Calculation of optimal hyperplane is done by solving an optimization problem, which maximizes the distance between hyperplane and the closest data point for both classes.

Although support vector machines are inherently linear classifiers, it is possible to separate linearly non-separable data using kernel functions. These functions map the input to high-dimensional feature space, where the data is linearly separable.

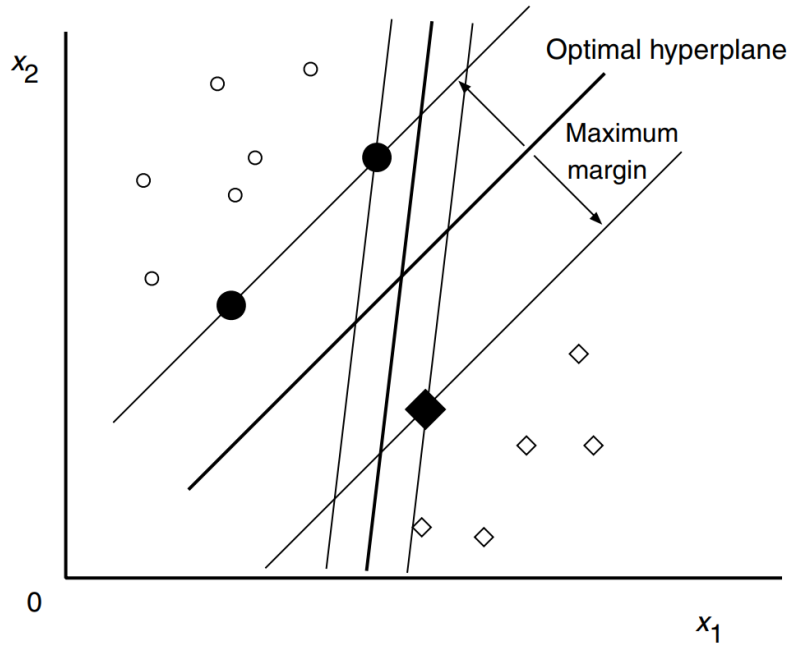

Figure 2.8: The optimal hyperplane and a nonoptimal hyperplane (from [Abe10])

2.4 Summary

In this chapter, related work for this thesis was reviewed. The guiding methodology for this project, CRISP-DM, has been briefly described. Furthermore, related work about item classification in e-commerce was summarized. Last but not least, in supervised learning section, evaluation metrics and supervised learning techniques that will be used in the upcoming chapters for image and text modalities were explained. In the next chapter, after describing the domain and the business goal of the project, the available data will be collected, analysed and prepared for training of the classifiers.

CHAPTER 3

Data Analysis

In this chapter the data that will be used for training the classifier will be analyzed and the problem domain will be explained in detail. First, the domain will be described. Secondly, data generated by user support team will be compared with data that has been generated by users of the e-commerce application. After determining which data source has fewer errors, the data collection process will be described in detail. Lastly, various preprocessing techniques will be applied on the data to make it ready for the training as described in CRISP-DM methodology.

3.1 Domain

The data to be used comes from a company in classifieds business, that is active in six different countries. The company develops an application which is available for iOS, Android and Web to enable users to buy and sell items. Users can register a new account or login using Facebook or Google accounts to have access to the main features of the application. Users have access to the following features in the app:

- **Listing an item**: Users can list an item by taking a picture, adding title and description, setting a price and (optionally) setting a category.
- **Browsing**: Users can browse through a list of items that are being sold at the current location of the user sorted by distance. Items in the list can be filtered by category among other options.
- **Making offers**: Users can make offers on items that have been listed by other users. An item is considered as sold when both seller and the person making the offer agree on the price.
- **Questions**: Users can ask questions to the seller on the page of a listed item.

The multimodal classifier's task is to map items that have not been assigned a category by the user to one of the 10 main categories. These main categories are:

1. Cars and Motors
2. Baby and Child
3. Electronics
4. Fashion and Accessories
5. Home and Garden
6. Movies, Books and Music
7. Other
8. Pets
9. Services
10. Sport, Leisure and Games

"Other" category can be considered optional for the classification, since it exists only to provide a safe option for the user if they are unsure.

If users do not set the category of their items, the user support team has to categorize them manually, which reduces the available workforce enormously. Additionally it is also intended to assist users during the listing of an item by suggesting categories in the future. That is why features that are not available before listing an item will not be used.

The business goal of this project is to reduce the amount of manual categorization done by the user support team. Project will be considered successful from the business perspective when all or part of the categorization is automated in such a way that the accuracy of classifier is at least equal to the accuracy of the user support team or normal users depending on which one is less accurate.

3.2 Data Quality Experiment

To answer the first research question of this thesis **RQ1** and to clarify the business and data mining success criteria, an experiment was performed to compare the error count of two subsets; one containing items that were categorized by users and the other one containing items that were categorized by the user support team of the company. Each subset contains 600 samples, 500 of which were picked randomly and 100 of these 500 samples were added twice to measure indexer consistency. Distribution of the categories for both subsets is the same.

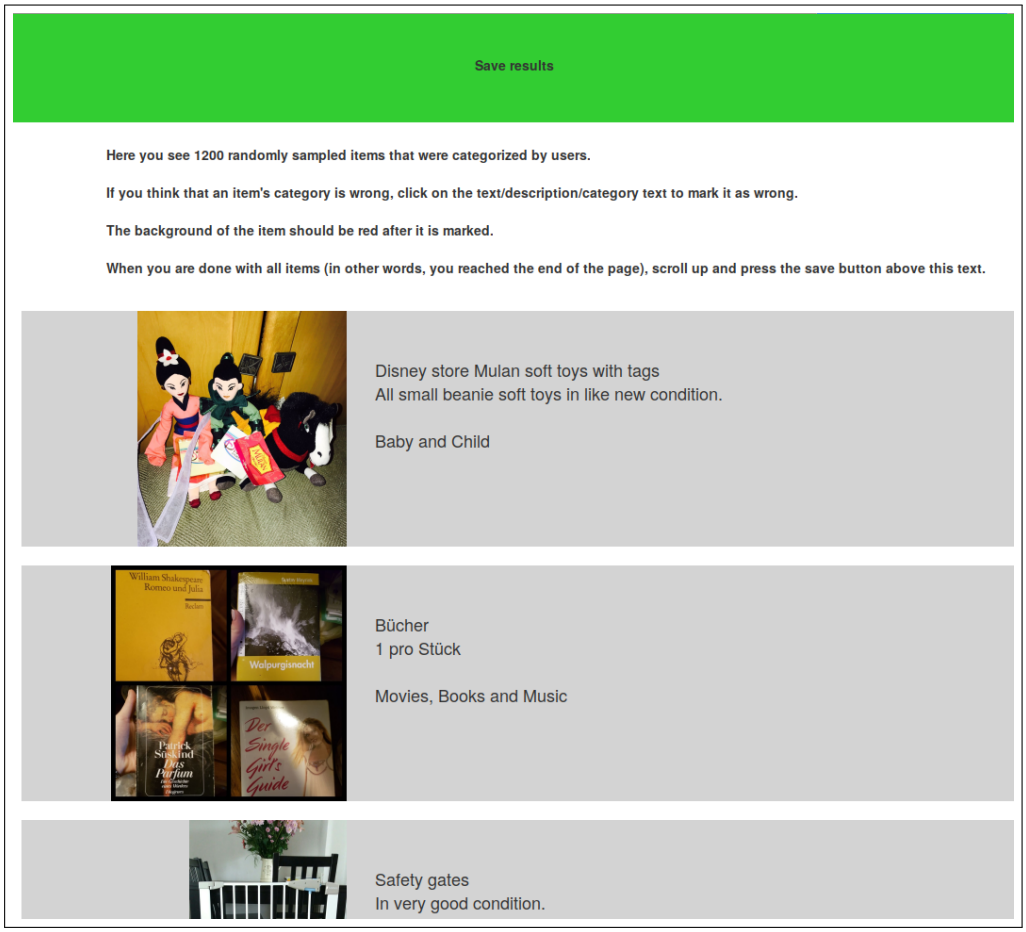

Figure 3.1: Web application used for the experiment

The experiment was performed through a web application. On the website a list of items were shown with image, title, description and the category for each sample as shown in figure 3.1. Before the real experiment, the web application was tested in a pilot experiment by the author. After ensuring application stability, 5 members of the user support team were asked to mark items in the list that were in the wrong category. They were also told that all the samples were categorized by users to avoid them having second thoughts during the experiment, since they might have been less likely to mark samples as wrongly categorized if they were told that the samples might also have been categorized by the user support team.

Results of the experiment show that the user-categorized samples have substantially less categorization mistakes than samples that were categorized by the user support team.

Out of 800 items that were in the list only once, 194 items were marked as wrongly categorized by different indexers. 6 items were marked as wrongly categorized by all indexers. The distribution of marked items for number of indexers is shown in figure 3.2.

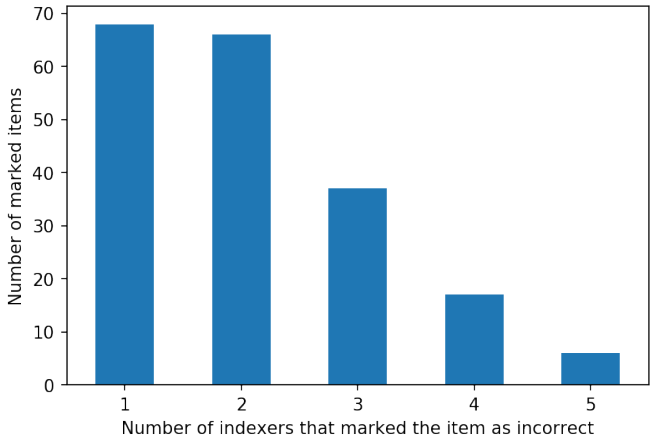

Figure 3.2: Marked item count according to the number of indexers that marked them

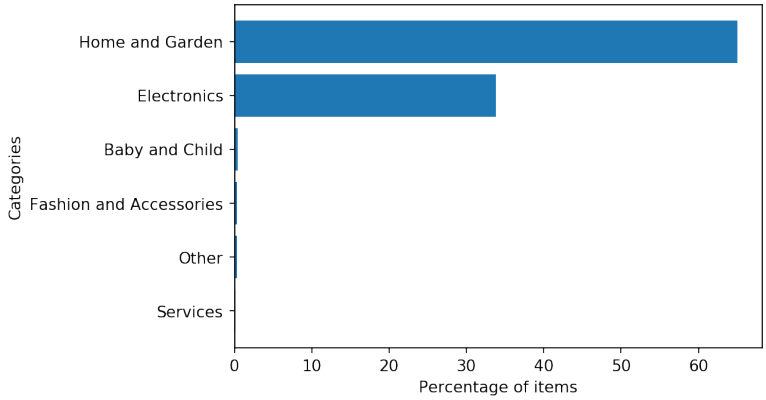

Figure 3.3: Category distribution of samples that contain substring "coffee machine" in the title

The high amount of items that have been marked by only some of the indexers show that categories are not specific enough. Some items can semantically belong to two or more categories. For example a coffee machine can belong to both "Electronics", "Home & Garden" and "Other" category. Category distribution of samples in the collected data for this work (described in detail in section 3.4) that contain "coffee machine" in the title is shown in figure 3.3. Two possible ways of dealing with overlapping categories will be investigated in this work. Firstly, since items that can belong to two categories are correct for both cases, appropriate evaluation metrics will be discussed in section 3.6 for a system with full coverage. Secondly, uncertainty functions will be evaluated in section 4.5 for a system which trades off coverage for top-1 accuracy by filtering out predictions about which the classifier is not confident enough.

The number of marked items between indexers were different but the error ratio between

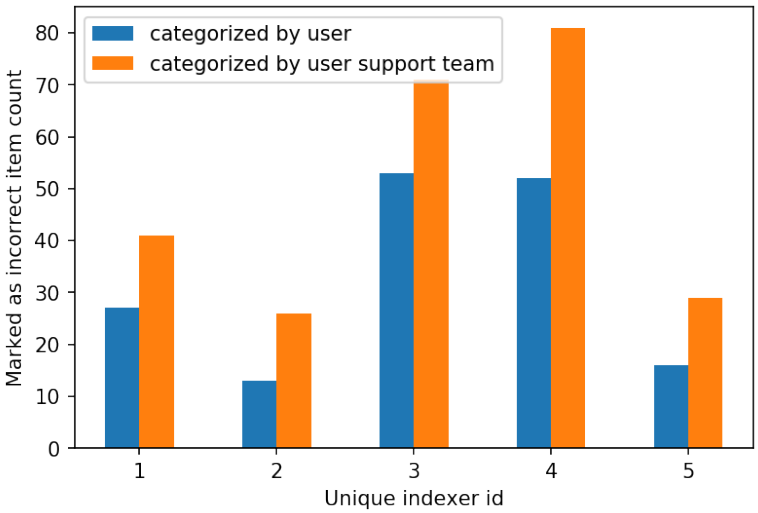

Figure 3.4: Marked item count by each unique indexer for each categorization subset

user-categorized and user-support team categorized subsets stayed the same, favoring user-categorized subset as can be seen in figure 3.4.

A possible explanation for the clear error rate difference is the bias caused by the fact that user support team only categorizes items that have not been assigned a category by the user. For most items that are easy to categorize, users sets a category, but for the items that are hard to categorize, more users might tend to post an item without selecting a category.

Additionally, average number of marked items by the experts (members of user support team, that evaluated the list of items) were calculated to determine accuracy of user support team and users. In addition to results for all categories, the results were also analyzed after removal of the "Other" category. These results are shown in table 3.1 and 3.2.

Categorized by	Total samples	E1	E2	E3	E4	E5	Avg. marked	Accuracy
User support	400	41	26	71	81	29	49.6	87.60%
Users	400	27	13	53	52	16	32.2	91.95%

Table 3.1: Performance related metrics for all categories

Furthermore, precision for each category for both groups have been calculated. A sample has been considered as marked, if it has been marked by one or more experts. Category specific results are depicted in table 3.3 and 3.4.

It must also be noted that the indexer consistency was not high. In total, out of 147

Categorized by	Total samples	E1	E2	E3	E4	E5	Avg. marked	Accuracy
User support	367	27	25	48	60	23	36.6	90.03%
Users	373	14	12	31	34	10	20.2	94.58%

Table 3.2: Performance-related metrics after removal of the "Other" category

Category	Total samples	Marked sample count	Precision
Baby and Child	33	11	66.67%
Cars and Motors	18	1	94.44%
Electronics	38	11	71.05%
Fashion and Accessories	138	18	86.96%
Home and Garden	77	17	77.92%
Movies, Books and Music	18	11	38.89%
Other	33	28	15.15%
Pets	7	1	85.71%
Services	5	2	60.00%
Sport, Leisure and Games	33	15	54.55%

Table 3.3: Category specific analysis for items categorized by user support team

Category	Total samples	Marked sample count	Precision
Baby and Child	32	9	71.88%
Cars and Motors	25	3	88.00%
Electronics	45	10	77.78%
Fashion and Accessories	134	6	95.52%
Home and Garden	77	6	92.21%
Movies, Books and Music	15	3	80.00%
Other	27	24	11.11%
Pets	7	1	85.71%
Services	5	1	80.00%
Sport, Leisure and Games	33	16	51.52%

Table 3.4: Category specific analysis for items categorized by users

marked items that appeared two times in the list, 97 of them were marked correctly. 50 items were marked only once by different indexers, although they were in the list twice.

3.3 Available Resources

The project will be developed on an Amazon instance of type p2.xlarge, which, at the time of writing, has the following specs:

- 1 NVIDIA K80 GPU
- 4 virtual CPU cores
- 61 GB RAM

The reason for the choice of instance is the GPU, which speeds up the training of neural networks tremendously. An extra volume with 500GB free space has been attached to the instance to store images for training the image classifier.

Main data resource is the data warehouse of the company managed by the IT Operations team. As data-warehousing solution Amazon Redshift is used. Images will be collected from media servers of the company with a python script using internally used ids of the images stored in the data warehouse.

3.4 Data Collection

The dataset was queried from the company's data warehouse and saved as a csv-file for further preprocessing. The dataset is limited to 2 million items that were listed in a one year period sampled randomly from Great Britain. The dataset is limited to this amount because the developed model will be a proof of concept. It contains only samples that have been categorized by users, because the quality of this source is higher as shown in the previous section. Data was gathered from "items" view in the data warehouse and enriched by joining user information about the seller in "users" view for each item. User information includes age and gender. User attributes were collected, because they can act as a helpful prior for item categorization as shown in the next chapter about preprocessing. The structure of the views is illustrated in figure 3.5.

The columns from both of these views, which were considered to be not relevant for item classification by the author, were omitted for clarity and confidentiality reasons. Collected data contains following fields:

- **i_id**: Unique id of the item, that is assigned after creation of an item entry.
- **i_title**: Title of the item.
- **i_dsc**: Description of the item.
- **i_title_length**: Character count of the title string.
- **i_dsc_length**: Character count of the description string.
- **i_price**: Price of the item.
- **i_media_ids**: Image ids of the item. The ids are stored as a single string seperated with comma; e.g. "imageid1,imageid2,imageid3".

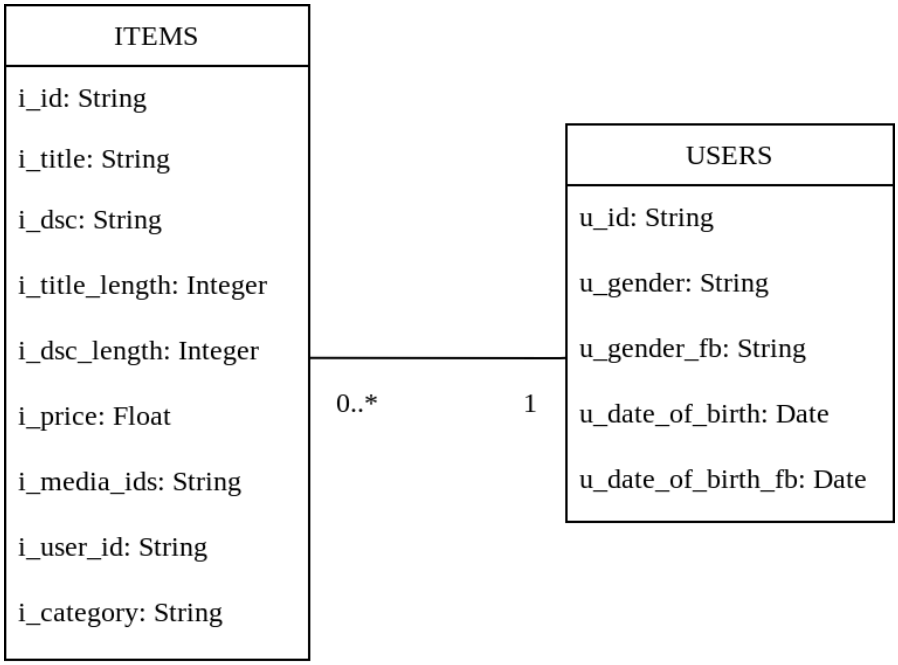

Figure 3.5: Structure of the views that were used for the data collection

- **i_user_id**: User id of corresponding seller of the item.
- **i_category**: Category of the item. This is the target variable for the classifier. Categories are saved as short codes in the database e.g. "ot" for the "Other" category.
- **u_id**: Unique id of a user account.
- **u_gender**: Gender of a user entered in the app.
- **u_gender_fb**: Gender of the user in the linked Facebook account.
- **u_date_of_birth**: Date of birth entered by the user in the app.
- **u_date_of_birth_fb**: Date of birth of the user in the linked Facebook account.

i_id, i_user_id and u_id were used only for data collection and will not be used for training.

3.5 Data Analysis and Preprocessing

Although some preprocessing was done with the SQL query during the dataset collection, more preprocessing was needed to make the data ready for training for image and text classifiers.

Along the 10 main categories there are various small categories that are used internally for a few items, which can not be mapped to any main category. Samples with these categories have been removed since they are not useful for the classification task at hand, which reduced the dataset size to 1998680 samples. Additionally subcategory values have been replaced with corresponding main category values. Distribution of the remaining categories can be seen in figure 3.6.

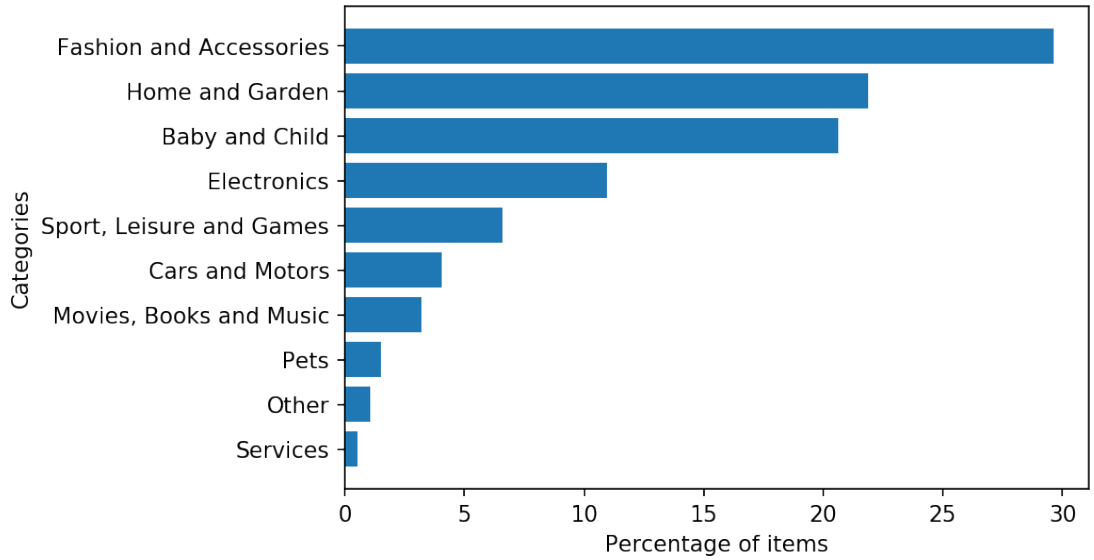

Figure 3.6: Size of each category in percentage

It must be noted that the "Other" category is a hard problem for classification, since many items are categorized as "Other" by users or user support team if they are unsure. Additionally by definition the "Other" category has no clearly distinguishing features. This is proven by the low precision values for both users and user support team in section 3.2. Because of these reasons "Other" category was removed from the dataset, which brought the dataset size down to 1977645.

Text preprocessing for title and description will be handled in the next chapter because comparisons of model performance for different text preprocessing techniques are needed to answer **RQ2**.

Since the prices of the items are determined by users and there is no binding agreement until both the seller and the buyer agree on the price, some users tend to list items for fake prices that are astronomically high to get buyers' attention. This can be easily seen by looking at price statistics in table 3.5.

These outliers were replaced with the the mean price of items below 0.99 quantile for each category. After this value replacement statistics turned to acceptable values as shown in in table 3.6.

Category	mean	std	99%	max
Baby and Child	1,237,482.32	289,717,486.79	200.00	99,999,999,999.00
Cars and Motors	1,244,064.89	191,366,659.64	11,500.00	50,000,000,000.00
Electronics	1,106,107.41	247,099,150.62	750.00	99,999,999,999.00
Fashion and Accessories	663,032.88	229,340,669.62	250.00	99,999,999,999.00
Home and Garden	385,316.16	141,875,879.22	500.00	90,000,000,009.00
Movies, Books and Music	18,227.93	3,978,206.93	300.00	999,999,999.00
Pets	6,438,045.16	652,387,585.11	1,150.00	99,999,999,999.00
Services	4,068,625.98	178,341,976.90	2,600.00	10,000,000,000.00
Sport, Leisure and Games	4,368,157.88	600,069,683.08	750.00	99,999,999,999.00

Table 3.5: Price statistics before removal of outliers

	mean	std	99%	max
Baby and Child	13.49	22.60	125.00	200.00
Cars and Motors	937.78	1,544.33	7,990.00	11,500.00
Electronics	91.96	120.54	599.00	750.00
Fashion and Accessories	16.65	27.15	150.00	250.00
Home and Garden	38.46	63.56	350.00	500.00
Movies, Books and Music	14.02	30.70	175.00	300.00
Pets	71.77	140.52	750.00	1,150.00
Services	58.62	187.77	1,234.00	2,600.00
Sport, Leisure and Games	50.84	85.09	450.00	750.00

Table 3.6: Price statistics after removal of outliers

The column "i_media_ids" contains all image ids for one specific item. The values are stored in the data warehouse as one string. The ids are sorted according to their position in the string. Only the first image for each sample was downloaded to keep the distribution of images per category same as the distribution of samples per category. The reason for downloading the first image is that the users tend to put the most representative photo as the first one. Additional images are often used to show defects of an item or receipts to show the buying price. Furthermore an additional feature containing image counts was added to data. Samples with missing images have been removed which brought the sample count down to 1977640.

For image classifier of the multimodal classifier ResNet-50 architecture [He+16a] will be used, since weights of this network trained on ImageNet data are available. For downloaded images, the same preprocessing described in the residual network paper is used. No augmentation is done due to amount of collected images and limited resources. Images were resized with their shorter side equal to 224 and center-cropped into 224x224 squares, which is the input size for ResNet-50 architecture. Per-pixel mean of ImageNet

Figure 3.7: Original image (left) and the preprocessed image (right)

images were subtracted from the downloaded images, since the weights of the Resnet-50 model were calculated for these images. Order of color channels were changed from RGB to BGR because ImageNet models were trained on images with color channels in this order. Although color channel change and resizing was done before the training of image classifiers, per-pixel mean subtraction was done on-the-fly during loading of images into memory during training because it was not possible to store images with negative pixel values in a space-efficient format like JPEG. Preprocessing for one image is shown in figure 3.7.

Age of a user is also an important feature, since older users might tend to list items in certain categories more often than younger users. To calculate an age for each user two columns about date of birth (u_date_of_birth and u_date_of_birth_fb) were used. These data are available for only some of the users. Sample counts for different data sources are listed in table 3.7.

Before calculating the age of users, both sources were analysed for possible errors. Looking at the distribution of u_date_of_birth shown in figure 3.8, it is clearly visible that there are anomalies for years 1989 and 1990. On a closer inspection it was found that in year 1989 only December contains errors. On the other hand errors are distributed throughout 1990, with an extreme case in January as shown in figure 3.9. There are approximately double the amount of birthdates in other months of 1990 in comparison to 1989 and 1991. A horizontal line at $y = 2500$ has been drawn to show the distributed error in year 1990

Data source	Sample count
No age data	1131396
App	512641
Facebook	228219
App & Facebook	105384

Table 3.7: Availability of date of birth data grouped by sources

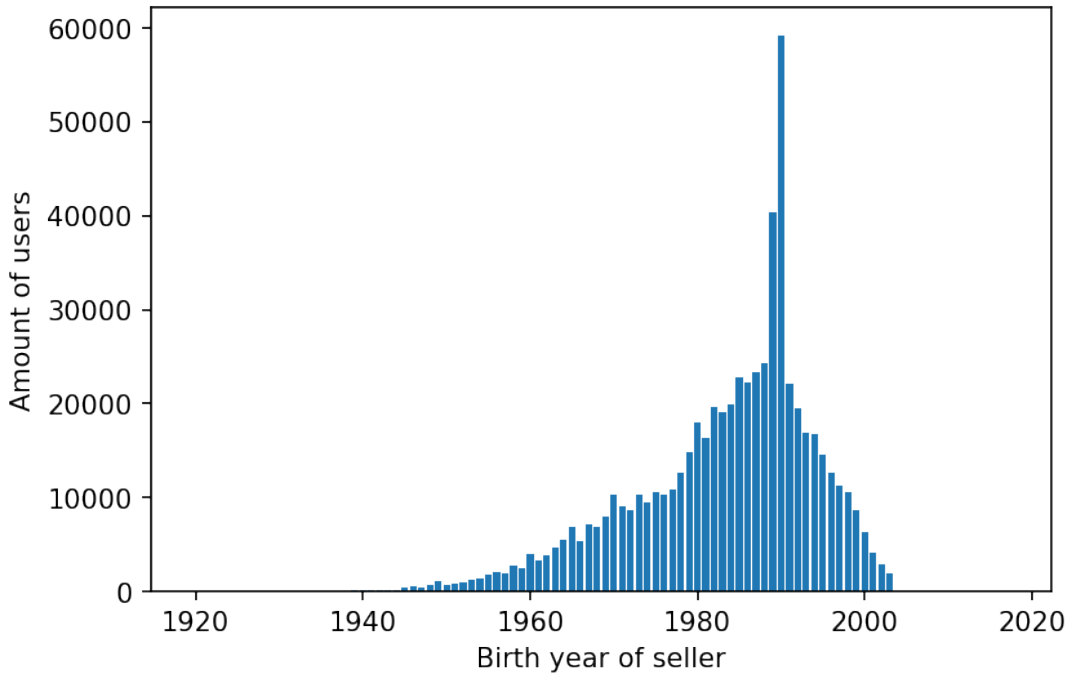

Figure 3.8: Distribution of birthdates entered in the app

more clearly.

To throw away as little information possible, December of 1989 has been inspected even further by plotting number of users with birthdays on each day in that month. In figure 3.10 it can be seen that the amount of users is exceptionally high from 29th till 31st of December.

Although it was not as drastic as with u_date_of_birth data, u_date_of_birth_fb data had an extreme case as well for year 1970 as shown in figure 3.11.

On a closer inspection the error could be localized to September 1970, specifically to 29th of September as shown in figure 3.12. This case was caused by a bug in a certain version of the app.

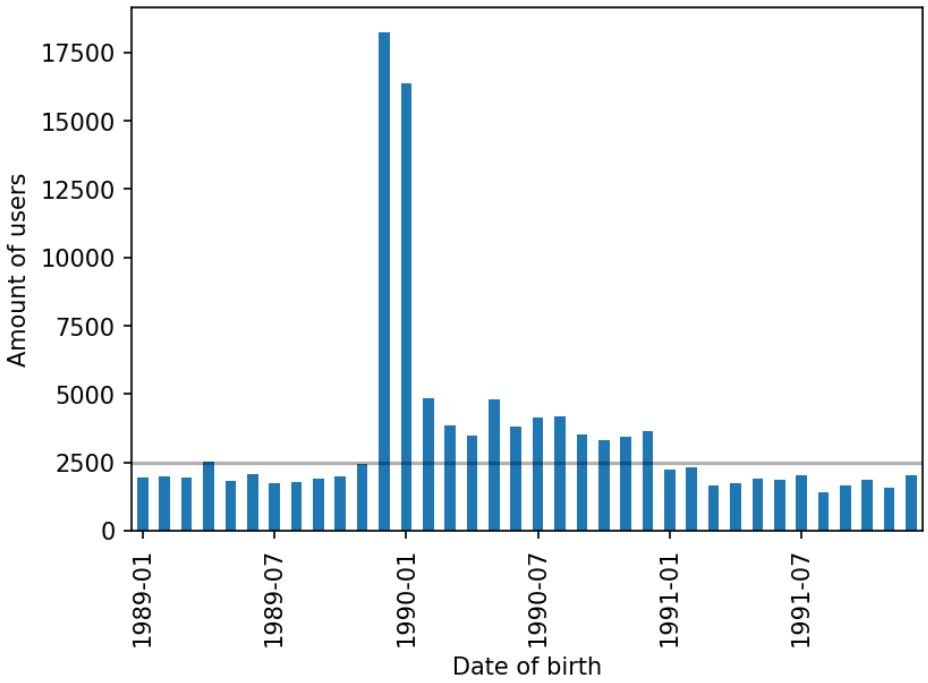

Figure 3.9: Distribution of birthdates entered in the app from 1989 till 1991

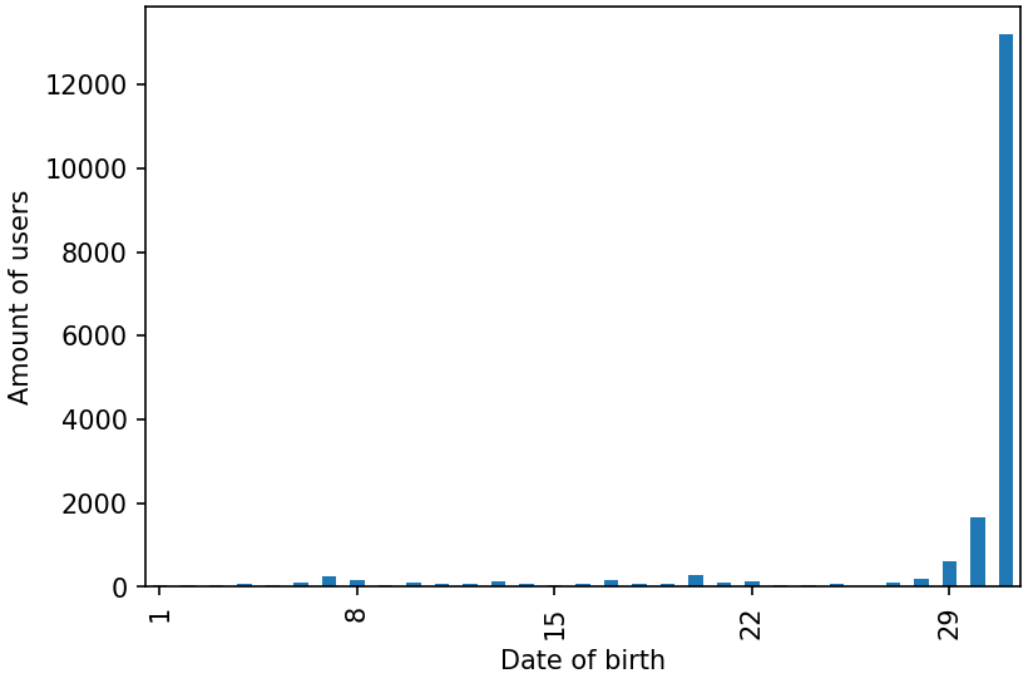

Figure 3.10: Distribution of birthdates entered in the app in December 1989

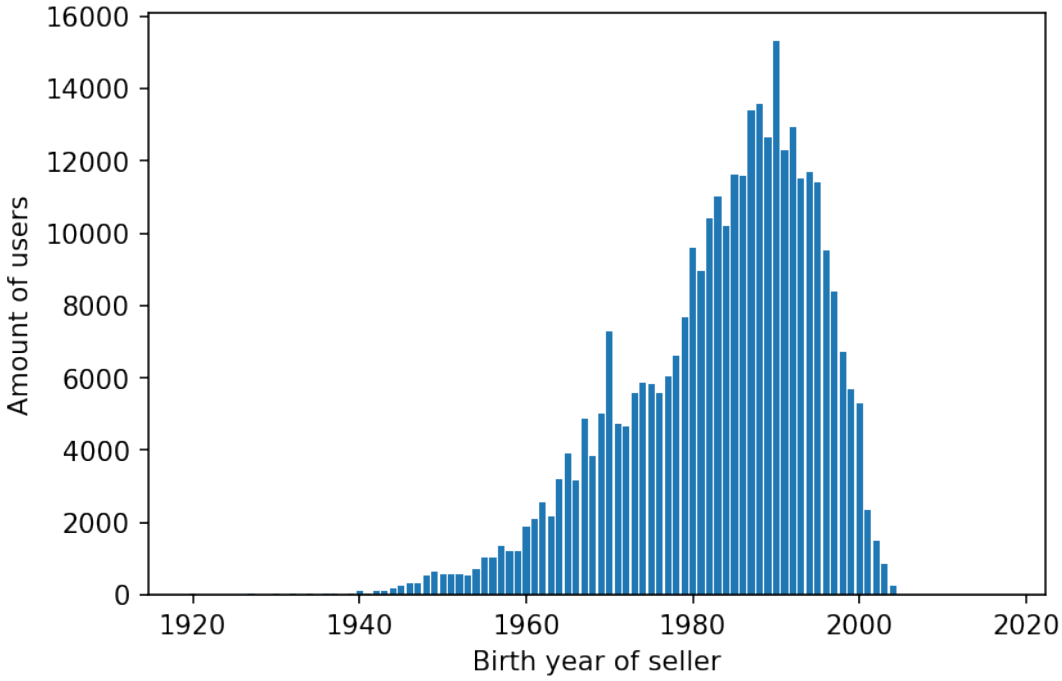

Figure 3.11: Distribution of birthdates from linked Facebook accounts

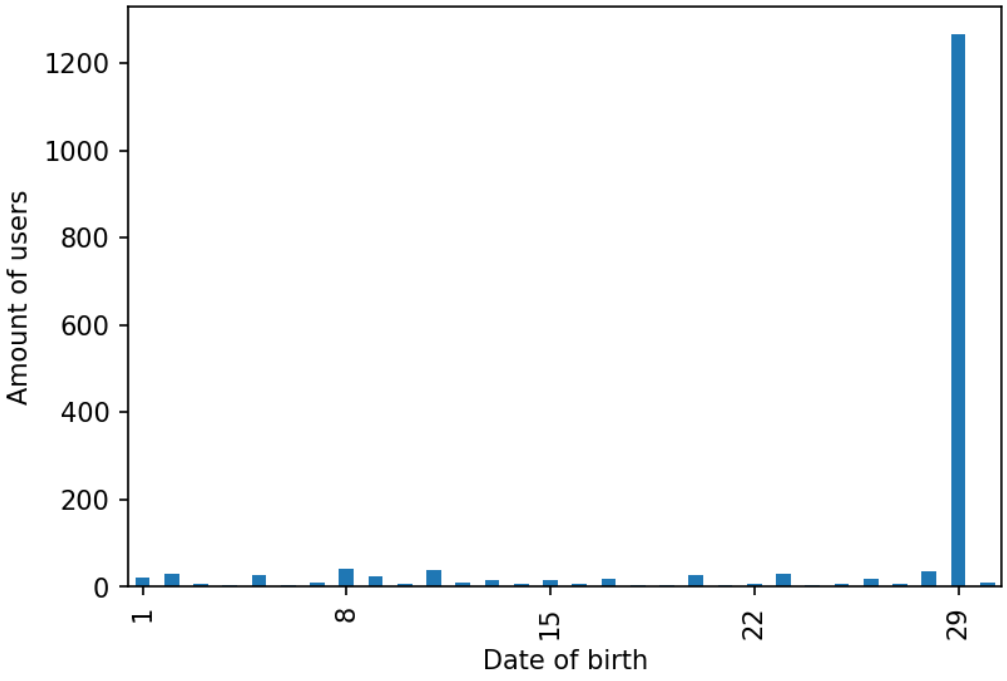

Figure 3.12: Distribution of birthdates from linked Facebook accounts in September 1970

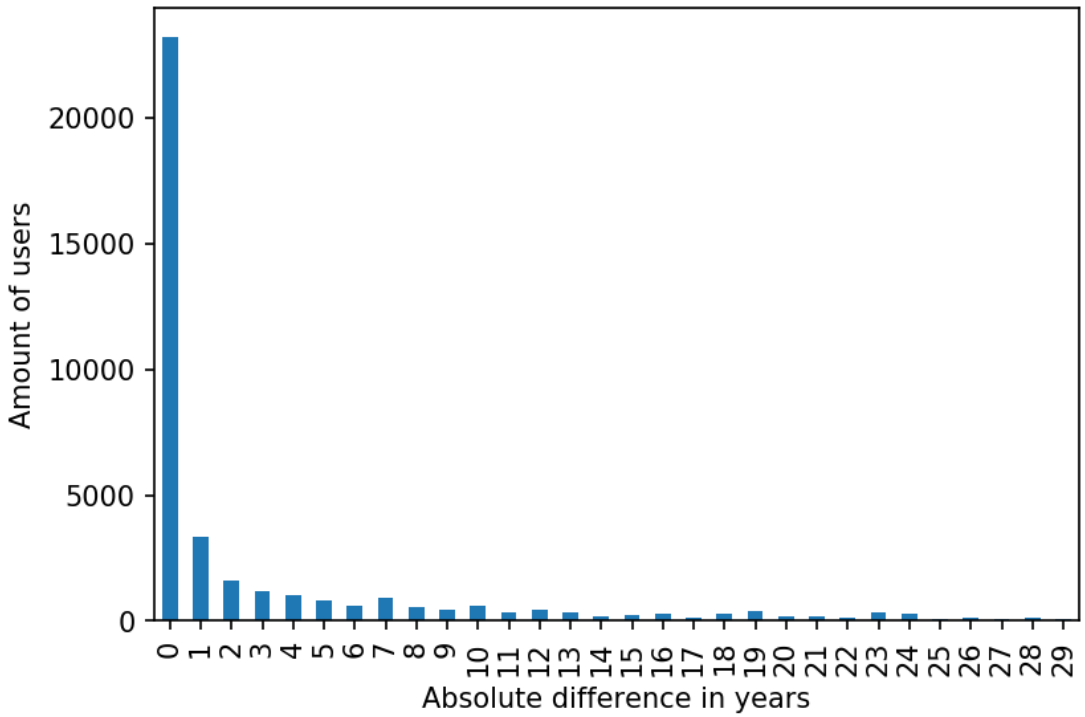

Figure 3.13: Mismatch statistics between App & Facebook birth years

In addition to these inspections, absolute difference of birth years between two age data sources for users that have a birth year for App and for Facebook were compared. As shown in figure 3.13 the birth year for most users match.

For the age calculation, Facebook data was given higher priority than data entered in the company's app, assuming Facebook data is more accurate. Users have a better reason to enter their real date of birth on Facebook, since there is a social incentive, whereas on an e-commerce app there is not a benefit in entering a correct date of birth for the user. Age of users that have a birthday in one of the error intervals have been calculated using the mode of birthdates that do not fall into these intervals. If no birthday is available, birthday has been set to mode of age distribution as well. Samples with ages above 100 were also mapped to mode of age distribution, since it is highly likely that samples with e.g. age 118 are fake. Comparison of listings of users with age 18-20 and 38-40 in main categories shown in figure 3.14 proves that the reasoning behind selecting this feature holds true. Younger users tend to list more products in 'Fashion and Accessories' and 'Electronics' categories whereas adult users of age 38-40 are listing more products in 'Home and Garden' and 'Baby and Child' categories.

The reasoning behind using user age also applies to gender feature. There are two possible gender data sources in the data warehouse; u_gender, which is entered value by user in

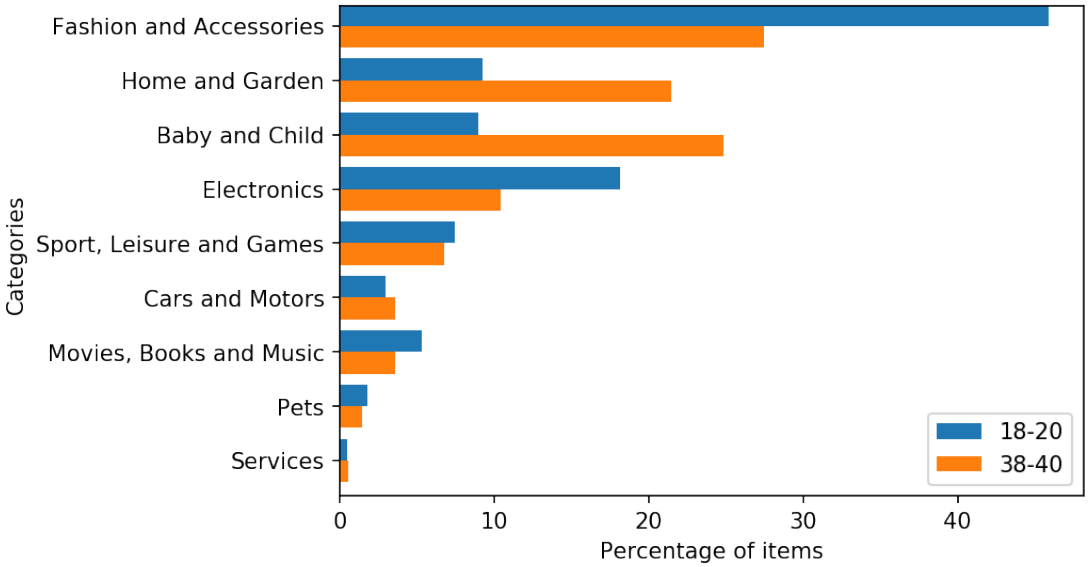

Figure 3.14: Comparison of listings of users with age 18-20 and 38-40 in main categories

Data source	Sample count
No gender data	356667
App	403187
Facebook	1036871
App & Facebook	180915

Table 3.8: Availability of gender data

the app, and u_gender_fb, which comes from the linked Facebook account. Availability of gender data is shown in table 3.8.

These two gender columns were merged into one. For cases where u_gender and u_gender_fb are both available u_gender_fb value was used, because of the social incentive. It is possible to set gender to three possible values in the app: "male", "female" and "other". Facebook only provides values "male" and "female" for gender. Remaining missing values for the merged gender column were replaced with the value "unknown". Statistics for the merged gender column are shown in table 3.9

3.6 Evaluation Metrics for the Problem Setting

Although there are many standard metrics that have been introduced in section 2.3.1, only a combination of these metrics is ideal for the problem setting for the available data. Since the category structure doesn't have a fine granularity, some items can belong to more than one category. However in the application it is only possible to pick one

Gender	Sample count
female	1166051
male	446537
unknown	356668
other	8379

Table 3.9: Sample counts of each value in merged gender column

category. Because of this limitation users categorize same items in different categories as demonstrated with the coffee machine example in section 3.2. A coffee machine can semantically belong to both "Home & Garden" and "Electronics" categories. As result of this, performance metrics that only consider the label with the highest score do not represent the actual performance of a classifier trained on the available data.

For the following test set:

Title	Category
"Coffee Machine"	"Electronics"
"Coffee Machine"	"Home & Garden"

if the classifier predicts the category as "Electronics" for both samples, accuracy of the classifier will be 50% although actual performance is 100%. This problem can be mitigated by relaxing the top-1 metrics. However just using top-2 accuracy can also be misleading as well since the classifier can give the highest score to a wrong category for a coffee machine like "Fashion & Accessories" but still be considered correct.

Since success criteria depend on the performance of user support team measured in the data quality experiment in section 3.2, which considered the overlapping of the categories during marking of the items with wrong categories, the appropriate evaluation should take into consideration that for some items there can be a list of correct solutions. For this reason top-1 and top-2 accuracy will both be used. Given the results from section 3.2, for a dataset without samples from the "Other" category, the multimodal classifier can be considered to be on par with the performance of the user support team for full coverage if the top-2 accuracy is above 90.03% even if the top-1 accuracy is below 90.03%. A minimum unweighted average precision of 70.68% will need to be reached as well to ensure that the model performs well enough for each category, since accuracy can be high even if the model performs poorly on small categories for unbalanced datasets. This value has been determined by taking the unweighted average of precision for each category in category-specific analysis results for items categorized by user support team in table 3.3. For a system which increases accuracy by lowering coverage based on probability outputs of the classifier, top-1 accuracy of 90.03% and unweighted average precision of 70.68% must be reached for it to be considered on par with the user support team.

3.7 Summary

At first problem domain has been described. To be able to define the success criteria for the project, the performance of the user support team needed to be measured. This has been done by performing a data quality experiment, where average percentage of wrongly categorized items for two possible sources, items categorized by users and user support team, have been determined. Furthermore available data have been collected from the data warehouse and the media servers of the company. These data have been analyzed and preprocessed so that they are ready to be used as input data for training of the models in the next chapter. Finally, evaluation metrics for the problem setting have been defined based on the business goal and the results of the data quality experiment for use in the next chapter.

CHAPTER 4

Modeling and Evaluation

4.1 Modeling

In this chapter various models will be proposed and evaluated for the data mining project. In the first section development of the text classifier will be described. In addition to evaluation of text classifiers, various preprocessing techniques and input representations for text will be compared for the evaluated text classifiers. The second section is dedicated to the image classifier, where a pretrained image classifier will be retrained for the given task. Effects of freezing layers at different depths will be compared. Furthermore, in the third section early and late fusion techniques will be compared for the multimodal classifier. Lastly, two uncertainty functions for the best performing multimodal classifier will be evaluated. Uncertainty functions determine which predictions to accept based on output of the classifier to increase number of correct predictions by lowering the coverage.

4.2 Text Classifier

For comparison of the two fusion techniques a text classifier was needed to generate predictions for textual features. For categorization of items title and description are considered to be the main textual features. All other features except images that have been collected or extracted will be considered as additional features and will not be used for evaluation of preprocessing techniques or input representations. For late fusion additional features will be used as input for text classifier and for early fusion they will be used as input for the whole model. For the text classifier two machine learning techniques were evaluated:

- Linear SVM
- Multinomial Naive Bayes

For the implementation the following libraries were used:

- **scikit-learn 0.19.1** [Ped+11]: for transforming text into vector representations, model training and evaluation
- **PyStemmer 1.3.0** [Bou]: for the fast implementation of the Porter stemmer
- **gensim 3.2.0** [ŘS10]: for generating word embeddings

4.2.1 Representation of Text Data

Text features can be represented in various ways. One of the simple representations is bag-of-words representation, in which a text is represented as a numerical feature vector. The number of features for each sample is equal to the amount of unique words in the whole dataset, which is called vocabulary. Each word in a text is represented as a number. This number can be in its simplest form binary, 1 if a word is in the text and 0 if it is not. In the case of term-frequency-based representation, the value is equal to the appearance count of the word in the represented text.

In text data there can be a lot of words that appear in multiple samples belonging to different classes. Such words are usually less discriminative than words that occur in only a few samples. However simple term-frequency-based bag-of-words representation gives same weight to each word. Term frequency-inverse document frequency (in short tf-idf) is an extension of term-frequency-based representation which gives higher weights to rare words and lower weights to common words, which are less likely to be useful for discriminating between classes. Weighting is done with inverse document frequency, which is calculated as follows:

$$idf(t,d) = log\frac{n_d}{1 + df(d,t)} \tag{4.1}$$

where t is the term, d is the document, which is another way of saying text data of a sample, n_d is the total number of documents and $df(d,t)$ is the amount of documents that contain the term t. Equation for calculating tf-idf for a single term is as follows:

$$tf\text{-}idf(t,d) = tf(t,d) \cdot idf(t,d) \tag{4.2}$$

where $tf(t,d)$ is the term frequency of term t in doc d.

Another approach for representing text features is word embeddings, where words are mapped to certain coordinates in a high dimensional vector space by an unsupervised learning algorithm. The mapping is done in such a way that semantically similar words are closer to each other in the vector space. For this thesis, continuous bag-of-words [Mik+13] was used to generate the embeddings. In table 4.1 the 10 most similar words sorted in descending order according to cosine similarity measure for four words are

	iphone	shirt	adidas	wardrobe
1	iphone7	shir	addidas	wardrobes
2	iphones	shi	nike	cupboard
3	ipone	shirt!	fila	draws
4	phone	shirts	puma	drawers
5	iphon	shirttop	lonsdale	drawer
6	iphone6	tshirt	addias	bookcase
7	unlocked	shirtjumper	adiddas	dresser
8	iphobe	shirtshort	ellesse	sideboard
9	64gb	shirtdress	mckenzie	wadrobe
10	lphone	lewin	lacoste	bedroom

Table 4.1: The most similar words for four different words in the embedding space

shown. When looking at the nearest neighbours of common words in the embedding vector space, it can be seen that the typos and synonyms are nearest neighbours of the correct word.

To keep the input size constant the technique proposed by Kozareva [Koz15] was used, where mean of embeddings of all words in the title is used as input. Although in her work only title of an item is used, in this thesis three input representations were evaluated:

- Mean of embeddings of all words in title
- Mean of embeddings of all words in title and description
- Concatenation of mean of embeddings of all words in title and mean of embeddings of all words in description

To make our results comparable, context window of size 5 and vector size of 200 are used, like in Kozareva's work. Word embeddings have been scaled to values between 0 and 1 for the training of Naive Bayes classifier, because the multinomial distribution used for multinomial Naive Bayes implementation in scikit-learn can not contain negative values. In addition to custom trained word embeddings, pretrained word embeddings were also evaluated for the three input representations. Pretrained vectors trained on part of Google News dataset [Wor] were used, because of the identical context window size of 5 and the use of continuous bag-of-words architecture for training.

Cleaned data was split into training, validation and test sets. The split ratio was 80/10/10 for training, validation and test sets respectively. For training and evaluation of classifiers identical training and validation sets were used. The test set was kept for the evaluation of final model configuration at the end of the project.

Performance of the classifiers are shown in tables 4.2 and 4.3, where it can be seen that term-frequency representation is the best for multinomial Naive Bayes while linear SVM

Representation	top-1 accuracy	top-2 accuracy	precision
tf (binary)	83.73%	95.04%	81.41%
tf	**84.22%**	**95.36%**	80.84%
tf-idf	81.37%	92.39%	**85.12%**
Word embeddings (title only)	57.03%	75.19%	48.54%
Word embeddings (title & description)	44.33%	65.66%	26.88%
Word embeddings (concatenated title & description)	59.92%	78.23%	60.12%
Pretrained word embeddings (title only)	32.50%	64.96%	33.16%
Pretrained word embeddings (title & description)	29.97%	58.25%	24.58%
Pretrained word embeddings (concatenated title & description)	34.53%	65.70%	32.84%

Table 4.2: Performance of multinomial Naive Bayes with different input representations

performs best with tf-idf representation. It is also worth noting that both classifiers performed worse with word embeddings than with other representations. Because this might have been caused by the dense representation in a vector space with lower dimensions, an SVM with a non-linear kernel was also considered for training. However the training time was too long and experiments with a limit on training process delivered even worse results than the ones with linear SVM. Because of this limitation, a multi-layer feed-forward neural network with two hidden layers (with 64 neurons per hidden layer, number of neurons and hidden layers were arbitrarily chosen) was trained instead, to verify if non-linear models deliver good results with word embeddings. Only custom word embeddings were used for this evaluation since the classifiers trained on pretrained word embeddings performed worse than the ones trained on custom word embeddings. As shown in table 4.4, although the results of these non-linear models were better than results of the linear SVM and Naive Bayes trained on word embeddings, they weren't better than tf-idf representation combined with a linear SVM. It can be hypothesized that the slight performance drop comes from the information loss caused by taking the mean of word embeddings which dilutes the important words for classification. Better results for the concatenated means of title & description in comparison to the representation without concatenation provide a supporting evidence.

4.2.2 Preprocessing Noisy Text Data

The main problem with textual features in the collected data is noise. The type of noise that is especially dominant in the given problem setting is usage of different cases for the

Representation	top-1 accuracy	top-2 accuracy	precision
tf (binary)	85.7%	95.23%	82.15%
tf	85.64%	95.16%	82.07%
tf-idf	**86.69%**	**95.98%**	**83.98%**
Word embeddings (title only)	81.27%	89.77%	71.98%
Word embeddings (title & description)	82.85%	92.85%	78.71%
Word embeddings (concatenated title & description)	83.61%	93.74%	80.08%
Pretrained word embeddings (title only)	75.16%	84.70%	66.26%
Pretrained word embeddings (title & description)	76.55%	87.55%	68.01%
Pretrained word embeddings (concatenated title & description)	78.26%	87.65%	68.92%

Table 4.3: Performance of linear SVM with different input representations

Representation	top-1 accuracy	top-2 accuracy	precision
Title only	84.34%	94.95%	79.58%
Title & description	85.58%	95.69%	81.26%
Concatenated title & description	86.07%	95.91%	81.69%

Table 4.4: Performance of MLP with custom word embeddings

same words like "iPhone"/"iphone". Additionally descriptions of items contain a lot of information that are not relevant for classification and only increase dimensionality, like URLs, phone numbers and stopwords.

Although basic data cleaning has been discussed in section 3.5, **RQ2** about preprocessing techniques against noisy text data will be investigated in this section, since performance of classifiers will be used to justify usefulness of these techniques.

There are many standard techniques that can be used to lower the negative effect of noise for text classification. For the text classifier, the following preprocessing techniques were evaluated:

- **Lowercase**: All tokens are lowercased.
- **Stopword removal**: A pre-defined list of common words like "the" are removed.
- **Normalization of price related tokens, URLs and phone numbers**: Special

Preprocessing	top-1 accuracy	top-2 accuracy	precision	dimensionality
Baseline	84.22%	95.36%	80.84%	424700
Lowercase	84.32%	95.44%	80.49%	314118
Stopwords	**84.45%**	**95.49%**	**81.14%**	424392
Normalization	84.22%	95.36%	80.56%	397538
Stemming	83.86%	95.11%	80.66%	406618
All	84.1%	95.35%	80.36%	277616

Table 4.5: Preprocessing results for multinomial Naive Bayes classifier

Preprocessing	top-1 accuracy	top-2 accuracy	precision	dimensionality
Baseline	86.69%	95.98%	83.98%	424700
Lowercase	**86.8%**	**96.13%**	**84.14%**	314118
Stopwords	86.62%	95.93%	83.87%	424392
Normalization	86.68%	95.98%	83.92%	397538
Stemming	86.42%	95.79%	83.63%	406618
All	86.29%	95.85%	83.45%	277616

Table 4.6: Preprocessing results for linear SVM

tokens are replaced with a common token. E.g. "http://www.example.com" is replaced with "**URL**".

- **Stemming**: Tokens are reduced to simpler forms based on certain rules. For English words, suffixes like "ing" and "s" are removed from the tokens. In this work Porter stemmer [Por80] is used.

For determining the effectiveness of these techniques, they have been applied to title and description of samples and classifiers were trained using the representation that achieved the highest top-1 accuracy on the validation set. In tables 4.5 and 4.6 the results for multinomial Naive Bayes and linear SVM are shown respectively. Although the effects are marginal, a clear trend is visible, that just using all preprocessing techniques does not help. For multinomial Naive Bayes, stopword removal improves the top-1 accuracy the most, while this performs worse than baseline for linear SVM. Linear SVM with tf-idf representation and lowercased tokens performed best, already fulfilling success criteria defined in section 3.6 for a system with full coverage.

4.2.3 Additional Features

In item categorization for e-commerce the research is not focused on feature engineering. In relevant papers that have been reviewed in literature research, for text-based classification only titles of items are used. However there are other features that can improve

classification performance. In this section advantages of using additional features that have been collected and extracted for text-based categorization are evaluated for the linear SVM with tf-idf representation and lowercased tokens. These features are:

- Title length
- Description length
- Price
- User age
- Image count
- User gender

Five numerical features were standardized by removing the mean and scaling to unit variance. The only categorical feature in the additional features, user gender, was one hot encoded. One hot encoding adds, for a feature with nominal values, one column for each possible value of this feature. These columns have values of either 1 (the sample has the corresponding value for the encoded feature) or 0 (the sample has a different value). For user gender this transformation looks like the following:

user id	user gender
1	male
2	female
3	other
4	female

user id	male	female	other
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

These features have been concatenated with the sparse matrix, which contains the tf-idf vectors for each sample. Performance metrics are shown in table 4.7.

Inputs	top-1 accuracy	top-2 accuracy	precision
Title+Description	86.8%	96.13%	84.14%
Title+Description+Additional features	86.92%	96.21%	84.3%

Table 4.7: Performance of linear SVM with additional features

Furthermore, tuning was done to find optimal value for the penalty hyperparameter C, penalty parameter of the error term. If C is set to a high value, a smaller margin will be used for the hyperplane and vice versa for the smaller values of C. $C = 0.5$ performed the best out of seven different values. C values that have been evaluated and their associated performance are shown in table 4.8.

C	top-1 accuracy	top-2 accuracy	precision
0.1	86.85%	96.19%	84.34%
0.25	**87.01%**	96.25%	84.50%
0.5	**87.01%**	**96.26%**	**84.52%**
1	86.91%	96.20%	84.26%
5	86.16%	95.74%	83.47%
10	85.82%	95.38%	82.77%
100	82.87%	93.65%	80.58%

Table 4.8: Evaluation of the C hyperparameter for linear SVM with additional features

4.3 Image Classifier

For image modality the ResNet-50 architecture proposed by He, Zhang, Ren, and Sun [He+16a] is used. Reasons for choosing the architecture are increased speed of convergence to a good solution because of the identity connections, high accuracy and availability of pretrained weights on ImageNet dataset. Instead of training the neural network from scratch, a pretrained ResNet-50 will be retrained for the given task. The retraining of a classifier is also called transfer learning, because the part of useful knowledge that has been gained from another task is transferred to another classifier, which is trained to succeed on a different task. For example features like edges in images are present in all images containing one or more objects. Initial layers of convolutional networks that are activated for certain edges can be also used for another task without a significant performance decrease. These layers are frozen during transfer learning, meaning that the weights of these layers are not updated during training. The results of this comparison will be used to answer first part of **RQ4** regarding the effectiveness of transfer learning. The second part of **RQ4** will be answered in the next chapter, which deals with investigation of performance differences between a retrained model versus a pretrained network on ImageNet as image component in a multimodal classifier. Scripts for training were implemented using TensorFlow library (version 1.8.0) [Aba+16].

To investigate the effectiveness of transfer learning with ResNet-50 for the given task, five different models were trained using the same architecture but with different number of frozen layers. As shown in table 4.9, ResNet architectures can be divided into five different sections according to the points where the output size changes. These points were used to define freeze depth for the models. E.g. conv5_x means that all layers including layers in conv5_x were frozen. Five models will be referred to with their layer name from this point forward. The relevant configuration for ResNet-50 is under the column named "50-layer", where each cell shows the size of filters and amount of filters in the convolutional and pooling layers of the network. E.g. "$3 \times 3, 64$" means a convolutional layer with 64 filters with the size of 3×3 and "3×3 max pool, stride 2" means a max pooling layer with patch size of 3×3 and stride of 2.

Because it was not possible to fit all images in memory, images in the training set were loaded as segments, each segment containing 2^{14} images. Reason for choosing a power of two for segment size was to have complete mini-batches for each segment. Mini-batch size for training was 64. To make training faster, the next segment was loaded asynchronously while the model was getting trained with a segment on GPU. Adam optimizer [KB15] with suggested hyperparameters ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) as in the associated paper was used to update the trainable weights. Models were trained for five epochs.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 4.9: ResNet architectures (taken from [He+16a]).

Training loss is shown in figure 4.1. Blue lines parallel to y-axis indicate epochs. Validation loss is shown in figure 4.2. During the training all models except conv5_x overfit on training set after initial epochs.

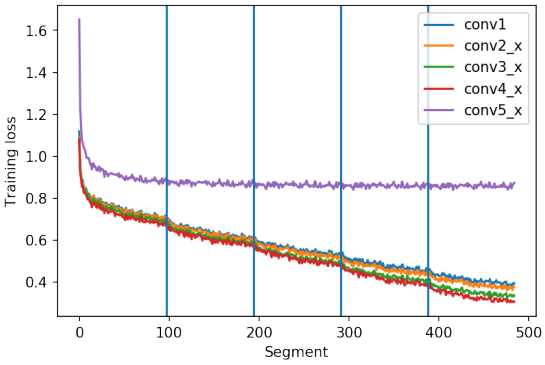

Figure 4.1: Training loss of image models

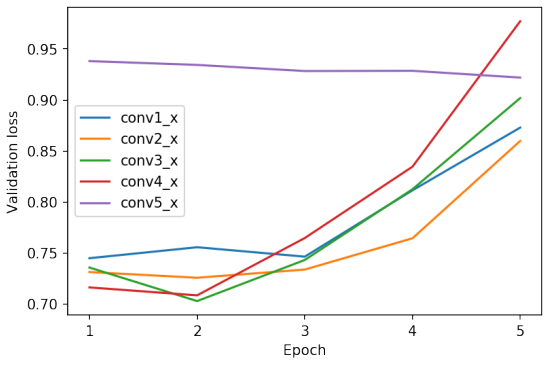

Figure 4.2: Validation loss of image models

Already after the first epoch performance metrics of all models on validation set were close to the best score achieved by these models in the later epochs. Top-1 and top-2 accuracy

and unweighted average precision on validation set are shown in figures 4.3, 4.4 and 4.5 respectively. conv5_x model kept improving only marginally reaching 69.11% top-1 accuracy after five epochs. conv3_x and conv4_x reached their best accuracy of 76.32% and 76.35% respectively on validation set and overfitted with each epoch after that point. conv1_x and conv2_x models reached their best top-1 accuracy score on validation set after third and fourth epoch with 75.69% and 75.8% respectively. Interestingly they did not reach a better top-1 accuracy than conv3_x or conv4_x although they had more trainable weights. A possible reason might be the granularity difference between ImageNet dataset and the dataset for the given task. Categories for the given task contain objects with many different features while ImageNet categories are more specific, allowing models to learn robust low level features. These trends were also same for top-2 accuracy and precision on validation set.

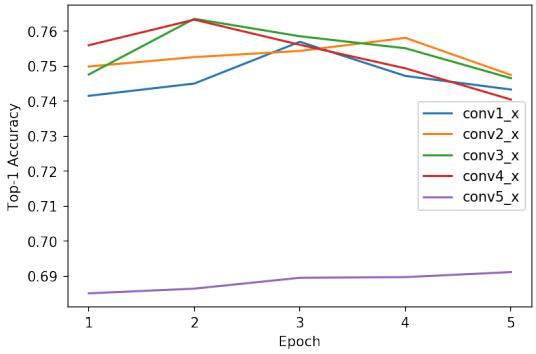

Figure 4.3: Top-1 accuracy of image models on validation set

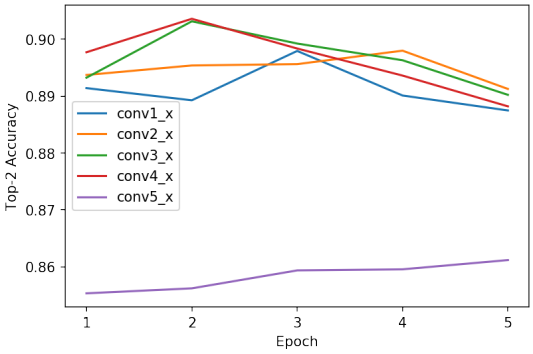

Figure 4.4: Top-2 accuracy of image models on validation set

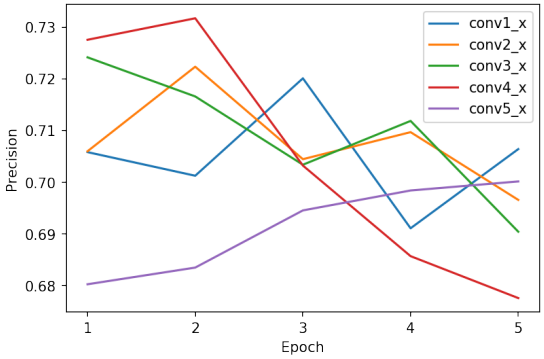

Figure 4.5: Macro-averaged precision of image models on validation set

Since performances of models were similar, average training time per epoch also became important for comparison. Each model needed a different amount of training time on average, getting shorter with increased amount of frozen layers in the model. Average training time per epoch is shown in table 4.10 in [HOURS:MINUTES:SECONDS] format.

Model	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	Average
conv1	12:57:40	13:33:40	13:38:50	13:20:01	13:20:28	13:22:08
conv2_x	11:22:23	11:14:48	11:19:31	10:51:27	11:04:03	11:10:26
conv3_x	9:18:33	9:14:17	9:18:35	9:11:05	9:09:08	9:14:20
conv4_x	7:15:53	7:18:27	7:18:24	7:22:27	7:31:52	7:21:25
conv5_x	6:31:23	6:27:02	6:31:16	6:38:04	6:37:19	6:33:01

Table 4.10: Training time for each epoch and their average per model

In conclusion, transfer learning with ResNet-50 was investigated for the images of an e-commerce company in this section. Best top-1 accuracy score was achieved by conv3_x but the second best model conv4_x was only behind 0.03% in terms of top-1 accuracy. Considering also the training time, top-2 accuracy and unweighted average precision score on validation set, it can be said that for the given dataset, conv4_x model had the best freeze depth for transfer learning. This leads to the hypothesis that having the last group of convolutional layers in ResNet-50 architecture trainable during training is enough for the model to adapt well to a new dataset while shortening the training time by a huge margin. Further research is needed to evaluate whether this rule also applies to other datasets as well.

4.4 Modality Fusion

In this chapter early and late modality fusion methods are evaluated for the given problem setting. Since performance of the retrained image model and the pretrained image model on ImageNet need to be compared as well, four different classifiers in total will be compared in this section. Based on the best results of evaluated classifiers and feature representations for text and image modalities, following multimodal classifiers were trained:

Early Fusion

early_retrained

- Text input: Lowercased text features as tf-idf vectors concatenated with additional features
- Image input: Output of flattened average-pooling layer before the final feed-forward layer in retrained ResNet-50

early_imagenet

- Text input: Lowercased text features as tf-idf vectors concatenated with additional features

- Image input: Output of flattened average-pooling layer before the final feed-forward layer in ResNet-50 pretrained on 1000 ImageNet classes

Late Fusion

late_retrained

- Text input: Output of linear SVM for each class with lowercased text features as tf-idf vectors and additional features as input
- Image input: Output of retrained ResNet-50 for each class

late_imagenet

- Text input: Output of linear SVM for each class with lowercased text features as tf-idf vectors and additional features as input
- Image input: Output of ResNet-50 pretrained on 1000 ImageNet classes

A multiclass perceptron with a softmax output layer has been trained to predict the correct class using the inputs listed above. As optimizer Adam algorithm was used with the suggested hyperparameters. Relevant performance metrics of models using the described fusion techniques are shown in figures 4.6, 4.7, 4.8 and 4.9.

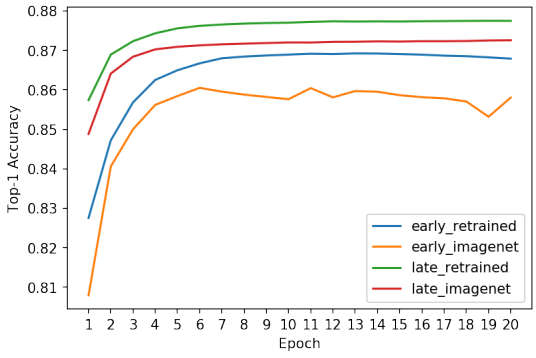

Figure 4.6: Top-1 accuracy of fusion models on validation set

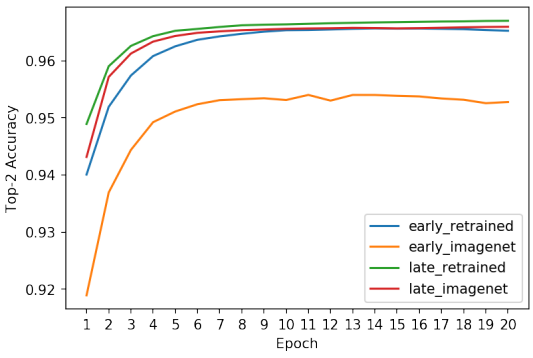

Figure 4.7: Top-2 accuracy of fusion models on validation set

As it can be seen in figures for all metrics, late_retrained configuration performed the best, which is the answer for **RQ5**. Confusion matrix for this model is shown in figure 4.10. This confusion matrix was normalized with sample counts of each class because of the class imbalance. The errors happen between certain categories, which suggest that these errors are caused by items that can belong to two or more categories. Because late_trained configuration performed better than all others, it will be used in

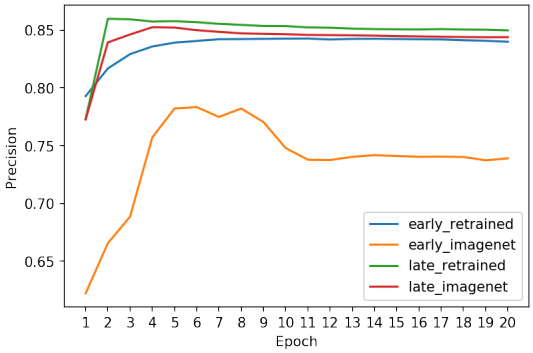

Figure 4.8: Macro-averaged precision of fusion models on validation set

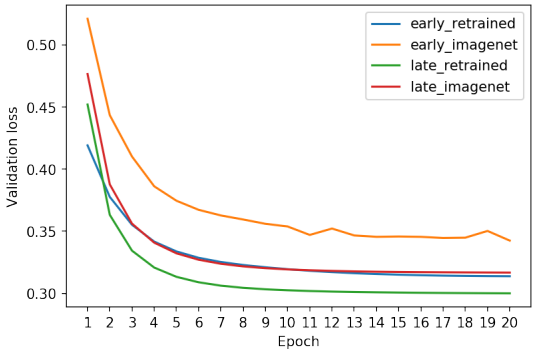

Figure 4.9: Loss of fusion models on validation set

the next section, in which uncertainty functions will be compared to deal with items that can belong to two or more categories.

4.5 Uncertainty Functions

Uncertainty functions determine whether the predictions of a model should be accepted or not. By accepting only predictions where the classifier is confident, it is possible to increase accuracy by lowering coverage. To increase the accuracy of late_retrained fusion model and answer **RQ6**, two uncertainty functions were evaluated:

- **Difference between first and second highest prediction probability**: Prediction is accepted if the difference between first and the second highest prediction probability is higher than a decision value.
- **Prediction probability threshold**: Prediction is accepted if the highest prediction probability is higher than a decision value.

First, outputs of the late_retrained multimodal classifier for each sample were obtained. Using these class probabilities, performance metrics and coverage were calculated for both uncertainty functions for different decision values ranging from 0 to 1 with step size 0.01. Performance metric versus coverage plots for these functions are shown in figures 4.11, 4.12 and 4.13.

Both functions reached equal top-1 accuracy and precision scores. Although by a small margin, prediction probability threshold function reached better top-2 accuracy than difference between first and second highest prediction probability function for coverages up to 50%.

Prediction probability threshold is a marginally better choice for the given problem setting, which is the answer for **RQ6**. Using prediction threshold function, 90.1% top-1

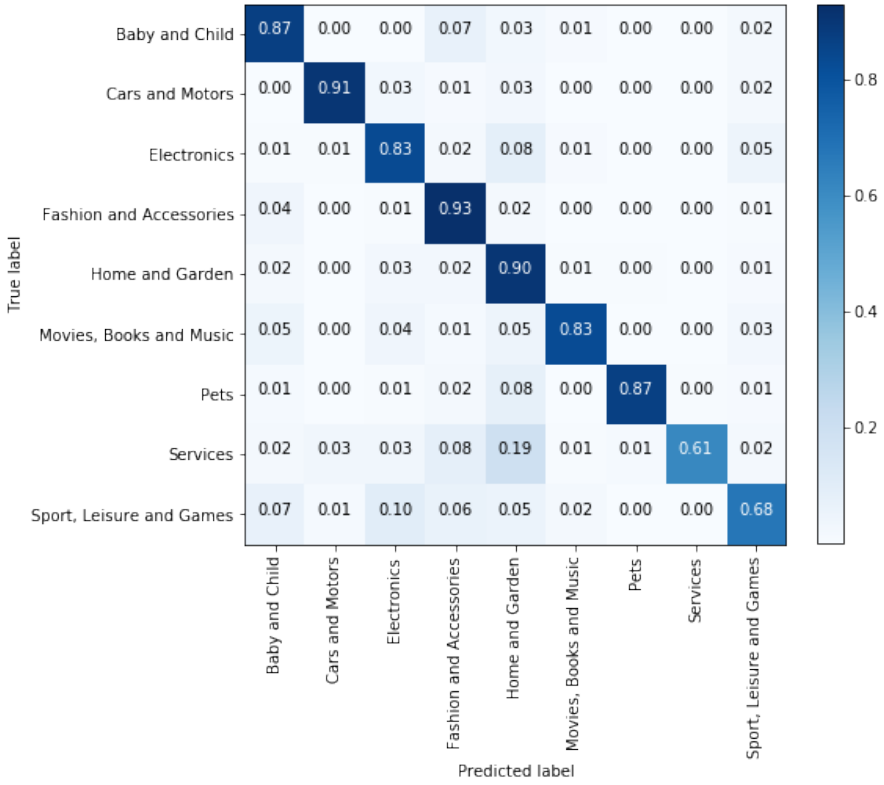

Figure 4.10: Normalized confusion matrix of the best fusion model (late_retrained)

accuracy can be achieved for 94,28% coverage with decision value of 0.54, which fulfils the success criteria for the system with accuracy-coverage tradeoff defined in section 3.6.

4.6 Final Evaluation

Since this classifier will not have any more hyperparameter adjustments, it was evaluated on the test set with prediction threshold decision function with decision value set to 0.54. Results, which fulfil the success criteria for the project, are shown in table 4.11.

top-1 accuracy	top-2 accuracy	precision	coverage
90.23%	97.63%	87.16%	94.23%

Table 4.11: Performance metrics and coverage on test set

Parameters of the whole architecture are as follows:

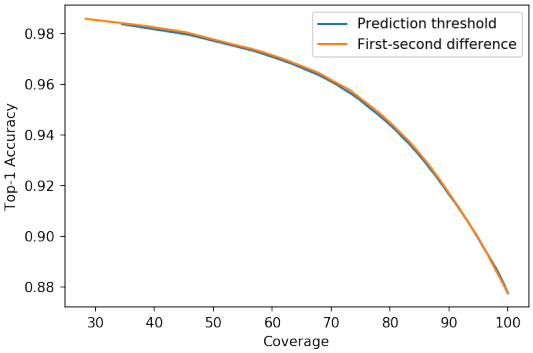

Figure 4.11: Top-1 accuracy with uncertainty functions for different coverages

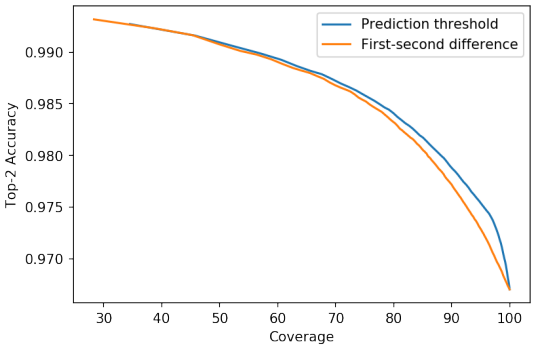

Figure 4.12: Top-2 accuracy with uncertainty functions for different coverages

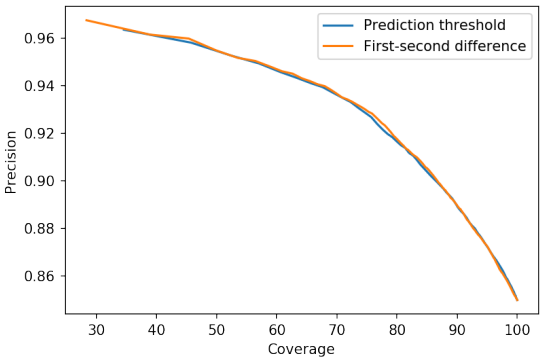

Figure 4.13: Precision with uncertainty functions for different coverages

Text classifier

- **Features**: Title, description, title length, description length, price, image count, user age, user gender
- **Preprocessing**: Lowercase for title and description, standardization for numerical features, one hot encoding for user gender
- **Classifier**: Linear SVM (C=0.5)

Image classifier

- **Features**: First image of an item
- **Preprocessing**: Subtraction of per-pixel mean of ImageNet images, resize with shorter side equal to 224, center-crop

- **Classifier**: Retrained ResNet-50 architecture (freeze depth=conv4_x, optimizer=Adam($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$))

Fusion

- **Features**: Output of the text classifier and output of the image classifier
- **Preprocessing**: Concatenation of the text and image classifier outputs
- **Classifier**: Multiclass perceptron with a softmax output layer
- **Postprocessing**: Prediction probability threshold as the uncertainty function with the decision value of 0.54

Analysis of Results in Business Setting

In 2018, 265292 items were listed without a category every month on average in Great Britain. Given the fact that the multimodal classifier reached the performance of the user support team categorizing such items by hand, it can take over this task. Items can be assigned to the category "Other", if the decision value of 0.54 can't be reached. Another option would be to partly automate categorization by leaving items to the user support team to classify them by hand if decision value of 0.54 can't be reached. Even in this case the manual work would be significantly reduced.

The multimodal classifier can also be used for assisting users during listing. However, considering the performance of the text classifier is not bad and prediction time for the multimodal classifier is significantly higher because of the image processing, it would be a better option to use the text classifier for user assistance to keep server response time low. The multimodal classifier is a better fit for already listed items, where they can be processed in batches in the background.

The classifier can be monitored by using the business intelligence tool utilized by the company. To check if there is a significant change in the ratio of items that the classifier can't classify, a monthly report can be created for the ratio between the number of items that couldn't be classified with a high enough certainty and the total number of items sent to the classifier. Another possibility would be to evaluate the classifier every month using the items that have been listed with a category in that month. Since users can still pick a category during listing, new training data will be available for the retraining of the multimodal classifier in the future, which will make it possible to combat topic drift.

4.7 Summary

In this section, the modeling and evaluation phase of the multimodal classifier was described. The development was done for each modality separately. After building the best classifiers for text and image modalities, four fusion model configurations were evaluated.

For the text modality linear SVM and Naive Bayes classifiers were compared using various text representations. After finding out the best text representations for each text classifier, preprocessing techniques were evaluated. After determining that linear SVM outperformed Naive Bayes classifier, effectiveness of using additional features with linear SVM was evaluated. Development of the text classifier was finished with tuning of the hyperparameter C.

For the image classifier, effectiveness of transfer learning for ResNet-50 architecture was investigated. For the given problem, training only the layers after conv3_x layer block was enough to get the best results.

After training text and image classifiers, early and late fusion techniques were compared. In addition to the retrained image classifier, configurations with a ResNet-50 image classifier trained on ImageNet dataset were also evaluated. Out of four configurations the late fusion classifier with the retrained image classifier performed the best.

Furthermore, to reach the performance of the user support team, uncertainty functions were evaluated. After finding the ideal decision value for the uncertainty function, the multimodal classifier was evaluated on the test set. All the evaluation metrics were above the performance of the user support team. Last but not least, possible deployment and monitoring strategies were discussed.

CHAPTER 5

Conclusions & Future Work

In this work, a multimodal classifier has been built for the item categorization task in an e-commerce problem setting. All research questions that have been listed in section 1.2 were answered satisfactorily.

RQ1 and **RQ3**, which dealt with the analysis of the problem domain, were answered by performing an experiment and analyzing the results. In this experiment accuracy and precision of categorization done by the user support team and users of the app developed by an e-commerce company were measured. Five indexers had to go through a list of categorized items and mark them if they had the wrong category. By looking at the number of indexers that marked an item as incorrect for each item, it was shown that categories are overlapping. Most items have only been marked by one or two indexers instead of all five. To deal with the inherent overlapping of categories, a combination of metrics were used to define the success criteria for the project. Additionally, two uncertainty functions were evaluated to increase accuracy by trading coverage for it, which is also a way of filtering out cases where two categories can be both correct.

RQ2, RQ4, RQ5 and **RQ6** that dealt with text preprocessing, transfer learning for image classification, modality fusion and uncertainty functions respectively, were answered by analysis of various techniques for the given problem setting. Techniques for text and image modalities have been investigated separately.

For text, input representations and preprocessing techniques have been investigated by comparing classification performance of models using these representations and techniques. Because the main focus of the work was on usage of a convolutional neural network in a multimodal context, only two learning algorithms have been evaluated for text-based models; Naive Bayes and linear SVM. Linear SVM with tf-idf representation performed best, which was further used for multimodal models. Although word embeddings trained on the collected text data captured the noise like typos well, performance of models using the mean of these embeddings as input performed worse than models using other input

representations. Further investigation is needed to see if this representation can lead to better results using a different model which does not lose information by taking the mean of embeddings.

For image, transfer learning with ResNet-50 architecture was investigated thoroughly. Five models were retrained, which had different numbers of frozen layers. Blocks of layers to freeze were determined by looking at the points in the architecture where the output size changes. Results on validation set showed that having one trainable convolutional block is enough to achieve the best result for the given problem setting. It remains to be investigated in future work as to whether this holds true for other datasets with the same configuration of ResNet-50 training as in this work.

In addition to investigating techniques for text and image models, early and late fusion techniques were compared. To answer the second part of **RQ4**, for each fusion technique two models were trained, one with the best image model retrained with collected data and one with an image model pretrained on ImageNet data. Late fusion with the best retrained image model performed best out of the four fusion models. For early and late fusion, the models with the retrained image model performed better than models with an image model pretrained on ImageNet in terms of top-1 accuracy.

Last but not least two uncertainty functions were evaluated to boost the performance of the best performing fusion model. Although the improvement was marginal, prediction probability threshold performed better for top-2 accuracy than difference between first and second highest prediction probability as the uncertainty function. For 94,28% coverage the multimodal classifier reached the success criteria for a system with accuracy-coverage trade-off on the validation set that has been defined in section 3.6.

After finding out the best configuration for the multimodal classifier, it was evaluated on the test set. Performance of the model stayed the same, fulfilling the success criteria. It reached top-1 accuracy of 90.23% and macro-averaged precision of 87.16% for 94.23% coverage. In addition to the test set evaluation, deployment and monitoring options for the project were discussed.

In summary, the developed multimodal classifier is able to take over a repetitive task from the user support team. This will enable them to focus more on tasks that humans excel at, such as helping users with their problems in the application. By automating the categorization, a significant gain is achieved from a business perspective.

List of Figures

List of Tables

List of Algorithms

Bibliography

[Aba+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[Abe10] Shigeo Abe. *Support vector machines for pattern classification*. Springer, 2010.

[Bou] Richard Boulton. *PyStemmer*. `https://github.com/snowballstem/pystemmer`. Accessed: 2018-05-06.

[Cha+00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. *CRISP-DM 1.0 Step-by-step data mining guide*. Tech. rep. The CRISP-DM consortium, 2000.

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[FM82] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[He+16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[He+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks". In: *Proc. of the European Conference on Computer Vision, Amsterdam*. Springer. 2016, pp. 630–645.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *Proc. of the International Conference on Machine Learning, Lille*. 2015, pp. 448–456.

[Kan+11] Anitha Kannan, Partha Pratim Talukdar, Nikhil Rasiwasia, and Qifa Ke. "Improving product classification using images". In: *Proc. of the 2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, pp. 310–319.

[Kar] Andrej Karpathy. *Stanford University CS231n*. `http://cs231n.github.io/convolutional-networks/`. Accessed: 2018-10-15.

[KB15] Diederik P Kingma and Jimmy Lei Ba. "Adam: A method for stochastic optimization". In: *Proc. of the 3rd International Conference on Learning Representations*. 2015.

[Koz15] Zornitsa Kozareva. "Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce". In: *Proc. of the 2015 Conference of the North American Chapter of the ACL*. 2015, pp. 1329–1333.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Proc. of the Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[LeC+01] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Intelligent Signal Processing*. IEEE Press, 2001, pp. 306–351.

[Mik+13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[Oqu+14] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. "Learning and transferring mid-level image representations using convolutional neural networks". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1717–1724.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Por80] Martin F Porter. "An algorithm for suffix stripping". In: *Program* 14.3 (1980), pp. 130–137.

[RHT01] Irina Rish, Joseph Hellerstein, and Jayram Thathachar. "An analysis of data characteristics that affect naive Bayes performance". In: *IBM TJ Watson Research Center* 30 (2001).

[RHW+88] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003.

[Roj13] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[Ros58] Frank Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[ŘS10] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. In: *Proc. of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, 2010, pp. 45–50.

[Sci] *Scikit-learn documentation for Naive Bayes*. `http://scikit-learn.org/stable/modules/naive_bayes.html`. Accessed: 2018-11-3.

[SRS12] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. “Large-scale item categorization for e-commerce”. In: *Proc. of the 21st ACM International Conference on Information and Knowledge Management*. ACM. 2012, pp. 595–604.

[Sun+14] Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. “Chimera: Large-scale classification using machine learning, rules, and crowdsourcing”. In: *Proc. of the VLDB Endowment*. Vol. 7. 13. VLDB Endowment. 2014, pp. 1529–1540.

[SZ14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[Sze+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going deeper with convolutions”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[Wor] *word2vec*. `https://code.google.com/archive/p/word2vec/`. Accessed: 2018-10-15.

[Yos+14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[ZF14] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *Proc. of the European Conference on Computer Vision, Zurich*. Springer. 2014, pp. 818–833.