Classify Swedish bank transactions with early and late fusion techniques

LOVISA B. SKEPPE

KTH Computer Science
and Communication

Master's Degree Project
Stockholm, Sweden May 2014

TRITA-EE 2014:xxx

Abstract

Categorising bank transactions to predefined categories are essential for getting a good overview of ones personal finance. Tink provides a mobile app for automatic categorisation of bank transactions. Tink's categorisation approach is a clustering technique with longest prefix match based on merchant.

This thesis will examine if a machine learning model can learn to classify transactions based on its purchase, what was bought, instead of merchant.

This thesis classifies bank transactions in a supervised learning setting by exploring early and late fusion schemes on three types of modalities (text, amount, date) found in Swedish bank transactions. Experiments are carried out with Naive Bayes, Support Vector Machines and Decision Trees. The different fusion schemes are compared with no fusion, learned on only one modality, and stacked classification, learning models in a pipe-lined fashion.

The early fusion concatenation schemes shows all worse performance than no fusion on the text modality. The late fusion experiments on the other hand shows no impact of modality fusion.

Suggestions are made to change the feedback loop from user, to get more data labeled by users, which would potentially boost the other modalities importance.

Sammanfattning

Klassificiera svenska banktransaktioner med tidig och sen fusion

Att sköta sin privatekonomi med hjälp av kategorisering gör nog många människor omedvetet, en försöker helt enkelt få en känsla på vad en lägger sina pengar på.

För att kunna ge full översikt på hur ens privatekonomi ser ut, har Tink skapat en mobilapplikation för att automatiskt kategorisera banktransaktioner. Detta görs just nu med klustering och längsta prefix matchning på försäljningsställe.

Kategoriseringen av banktransaktioner ger användaren en direkt återkoppling på hur pengaflödet ser ut samt till vad och när dessa köp görs.

Den här uppstasen kommer att undersöka om en maskininlärningsmodel kan lära sig att klassificera banktransaktioner baserat på köp istället för försäljningsställe. Genom att undersöka två olika fusionerings scheman på tre typer av modaliteter funna i banktransaktioner (text, pris och datum), ska vi försöka uttröna dessa modaliteters påverkan på klassificering. De olika scheman är jämförda med *ingen fusionering*, dvs inlärning på endast en modalitet, och *travad klassificering*, dvs inlärning med flera efterföljande modeller.

Experimenten är gjorda med *supervised-learning* och inlärningsmodellerna är Naive Bayes, Support Vector Machines samt Beslutsträd.

Experimenten visar på att klassificering på text, alltså försäljningsställe ger bäst resultat i jämförelse med alla de andra experimenten. I de *tidiga fusions* experimenten visar alla modalitet-sammanslagningar sämre resultat än ingen fusion på bara text. De *sena fusions* experimenten visar å andra sidan ingen skillnad alls efter fusionering med modaliteterna pris och datum.

Förslag på förbättad klassificering på köp antas öka, alltså modaliteterna pris och datum bör vara mer betydande, om mer data märkt av användare användes i träning.

Contents

Chapter 1

Introduction

1.1 Background

Creating budgets and keeping track of ones personal finance is not something new. Dividing income to essential expenses and savings for future unexpectedness is a necessary in our daily life to prevent personal ruin.

Some people manage to do this in their head or by hunches, but others needs help to track where their money goes.

The Tink approach is to provide a personal finance tool to categorize and visualize all transactions in a predefined hierarchy of 45 categories, like *restaurants*, *public-transport* or *rent*.

Categorizing these bank transactions is not a unique problem either and there are many examples of applications that do this both manual and automatic. When categorizing bank transactions manually, people tend to loose interest quite fast because of the administration and the often repetitive task it means (peoples spending/income pattern are quite predictable).

The automatic applications on the other side helps remove the repetitive task of manually classification, but the side effect are that they are often based on the merchant (the description) instead of the purchase (amount + description), meaning that the need for manually correcting the classifications are inevitable to have the correct information of ones personal finance.

So, it is the knowledge about the purchase who would give the user the best indication on where the money goes, thus categorizing on purchase would be the most desired goal for both users and service provider.

1.2 Tink

Tink is a Swedish app company providing a service to keep track of a users personal finance, across different credit card and account providers. The company was founded 2012 and has at the writing of this thesis, 180k number of users where around 10k are daily active on the app. Each day 800k transactions are handled by Tink servers thus the speed of classification on newly arrived transactions are essential for providing this tool.

At the moment Tink categorises transactions with clustering and longest prefix match.

1.3 Problem Specification

The motive for this thesis is to explore if a model can learn about the specific purchase behind the transaction, by looking at all of a transactions information (text, amount, date) instead of only the merchant (text).

The goal is to teach a machine learning model to classify bank transactions with a better accuracy (higher precision and recall) than Tink's system does now and also facilitate for the user to make a manual decision if the model has failed to decide a category (label ranking, see Definition 2.1).

The problem is situated on a global level, learning nothing from a single users shopping habits.

1.4 Approach

This thesis will exploit late fusion (see Section 2.1) and early fusion (see Section 2.1) schemes on bank transactions, meaning the different data parts of a bank transaction will be seen as different modalities. For Swedish bank transactions the modalities are description (text), amount (numbers) and date.

We will conduct two experiments where the first experiment shall exploit early fusion properties, meaning that all modalities of the data are represented in the same fashion and concatenated in different ways. To compare the early fusion results but also to learn properties from the different modalities, four tests will be done on "no fusion" meaning no concatenations are made and each modality are trained and tested individually but with the same experimental settings as for early fusion.

The second experiment will look at a late fusion scheme, where individual models will learn a modality separately, then fusion the result from each classifier which will train a final classifier. As with the first experiment, to compare this result a simultaneously test with no fusion in feature space (see Definition 2.1) and no fusion in concept space (stacking, pipe-lined learning models) in order to conduct appropriate comparisons.

I will test three different supervised learning models in the first experiment to find the best one for each modality, the findings from the no fusion tests will be used in the second experiment and the model choices done there.

I will not explore more than essential settings and optimisations for these models. The motives for testing the three algorithms are both to find out if there is a difference in categorisation accuracy based on modality properties or if there are some model specific properties that are desirable for other parts of a learning system, like label ranking or manual model corrections.

1.4.1 Limitations

This thesis will only consider bank transactions labeled *expenses* below 10k SEK. This is the categorisation step that Tink wants to improve. However, there is nothing in this thesis solution proposal who says it cannot extend the model to higher amount and more categories in income and transfer segment. Although this thesis will not test the relevance of number of categories for the model performance, an extension to cover the whole category space of Tink is possibly a straightforward procedure.

1.5 Outline

The following chapter will cover related work and the basic theory needed for this thesis. Chapter 3 will present the experimental setup and implementation followed by result, discussion and conclusion. The reason why the text modality are the most discussed in this thesis is because it is the one modality having the most differentiating properties of all modalities, which will be clarified.

Chapter 2

Theory

2.1 Definitions

- **Instance/document/exemplar/data point** Uses synonymously and means one unit of data in the data set, in this thesis it means a bank transaction.
- **Corpus/Vocabulary** A list of all tokens from the descriptions found in a pre-processing step of the training set.
- **Feature/Attribute** A specific part of a data instance. If the data is a vector, a feature is one dimension in that vector.
- **Feature Space** The space built up of all feature dimensions. If an instance contains features outside of this space, those features will not be used and removed.
- **Early fusion** "Fusion scheme that integrates unimodal features before learning concept" [1].
- **Late fusion** Fusion scheme that first reduces unimodal features to separately learned concept scores, then these scores are integrated to learn concepts [1].
- **Label ranking** Label ranking or sometimes referred to as soft categorisation, where a ranked list of all categories are presented instead of one predicted category from the decision function.
- **n-gram** A n-gram is all possible n-sized part of a text segment, including white space. For example the text "Lovisas belåningsbyrå" would be chopped into 16 6-grams.

```
Lovisas belåningsbyrå :

Lovisa
 ovisas
  visas
   isas b
```

```
sas be
 as bel
  s belå
    belån
    beláni
     elånin
      låning
       ånings
        ningsb
         ingsby
          ngsbyr
           gsbyrå
```

2.2 Related work

Since the propagation of Internet and people's increasing activity and content creation explosion, various application based on machine learning has emerged to analyse and understand these content flows.

One important application is text analysis and text categorisation (TC), where its used for spam detection [2][3][4], sentiment analysis [5] and categorizing articles into predefined categories such as news, sport, culture and politics [6][7].

But it is not only text flows that needs to be understood and categorized, also video and image classification has developed for the same purposes [1][8].

Lately it has become clear that classification models performs better or worse on different kind of modalities and depending on how and when these modalities are fusioned, the better classification performance you get [1][9].

Fusion techniques are studied for instance in biometrics classification [10], robotics [11] and semantic video analysis [1].

2.3 Data Representation - Multimodality

For a machine to quantify a text or an image, it has to pre-process the data so that it can easily be interpreted and understood. This pre-processing step is highly application dependent and crucial for model performance [12][13].

Continous data can be seen as a type of distribution or it can be discretisized by dividing the data in intervals [14].

For text it is non-trivial and an ongoing area of research. A standard model to represent text is the *Bag-of-Word* model (BoW), where the idea is that the meaning of a text comes from its words, regardless of their position in the document, thus cutting out all the words from a text, reorganize them and then tape them together would in the BoW model be seen as the exact same text [2][13][15].

In the BoW model documents are vectorized, meaning a word is representing one dimension of the document vector, this is sometimes referred to as surface representation [16]. The BoW model can be extended to a more generalized vector model where a vectors dimension does not necessarily mean a word. It

can be an n-gram of words or parts of a word or a discretisized version of another type of data modality that can represent one dimension of a vector.

The more distinct words and sentences a document set has, the larger the feature space gets. The features contain weights saying something about that particular feature's importance. For example, *Binary weighting*, *Term Frequency weights*, *normalized term frequency* and *Term Frequency- Inverse Document Frequency* (tf-idf) are different weighting schemes [13].

2.4 Text Classification

Text classification or text categorizing is the specific Information Retrieval problem where text documents are to be assigned to pre-defined categories [2][13]. Most related work in text classification focuses on larger document sizes, like the Reuters document set [17] and 20NewsGroup [18]. However, because of applications like Google (search engine) and Twitter (micro-blogs) there has been later studies in text classification where the document lengths has shrunk, meaning previous techniques has to be revisited and tuned in order to find better methods who represent the information in a text [7][16][19]. Regardless of text size Jackson and Moulinier list some important aspects on the data to consider when conducting text classification: *granularity*; How fine do the categories divide the document space?, *dimensionality*; How many features are needed to represent a document?, *exclusivity*; How many categories does a document belong to? and *topicality*; How many topics does a document touch? [12].

Text Classification often suffers under *the curse of dimensionality* meaning the number of training data points needs to increase along with the number of features, making learning algorithms slow and intractable [14]. This has lead up to many studies on data representation, feature selection and dimension reduction [13][20]. Metzler et al. evaluating different techniques for measuring similarity in short segments of text and Yih et al. seeks to improve these techniques [16][21]. They explore how to expand a search query representation by query an external source, enriching the text segments with more relevant words to the topic [16]. Phan et al. sees problem with web-search in real-time applications due to time constraints. Instead they have examined the use of an external source in training [7]. Sriram et al. did on the contrary not try to extend information but to use meta data as features in order to make better classification accuracy of tweets for filtering purposes [19]. Dumais et al. explored the possibility to do hiearchial classification to minimize the percepts error if an instance that is *sport:other* are categorized as *garden:other* it is preferable to miss-classify the instance in the sports bucket [6].

2.5 Supervised Machine Learning

Machine Learning is one technique to solve a classification problem. Depending on the characteristics of the problem different learning approaches are more appropriate than others [13]. This thesis will only consider supervised learning, meaning learning with labeled exemplars.

Learning is essential for great performance and having a ground truth to an exemplar serves to find reliable patterns in the data. But there are always some

problems to take measures for, when has the model learned enough patterns without learning too much? This is the famous phrase 'over fitting' in the context, and is when the model sees patterns that are not there. When over fitting occurs the model cannot generelize, make good guesses on previously unseen data.

2.5.1 Active Learning and labeled data availability

Supervised learning on data often lacks the availability of sufficient amount of labeled data. Active learning is a technique where the learner itself actively request the data points it need to improve classification [22]. There has been a lot of research in the area of active learning to minimise the need for humans to manually label data when the availability of data is low, for example pool based learning [23] and coactive learning [24].

Although manual interactions with a classification process has to be considered to be expensive and time consuming, some human interactions are however needed. Jackson and Moulliniere suggests that deciding how much human expertise are needed contra the sophistication of the system depends on the complexity of the text classification problem [12].

Other applications that makes use of active learning is search engines, detecting which documents the user chose serves as an indicator of how well the search engines document ranking works corresponding to a search query, called implicit feedback [25].

If human labeling is inevitable, the quality of what to consider when labeling should be established. What could be missed about a transaction if a person not responsible for that purchase is labeling that transaction?

2.5.2 Learning Models

There are many algorithms to choose from when conducting supervised classification.

For applications using machine learning it is not always the best accuracy of a model that will be the decisive flip of choice, therefor it is important to also look for properties of a model that can help in terms of ranking decision, modify parameters etc [12]. The following three models are each one from these three types; probabilistic classifiers, rule-based classifiers and linear classifiers. Choosing a model is based on several things which are explained in [26][27].

Naive Bayes

Naive Bayes, NB, often serves as a base line when it comes to text classification. The reason for this is that the NB is simple and fast to implement [28]. It follow the classical Bayes argument, see Equation 2.1, that a collection of n documents $D_1...D_n$ each belong to one of the categories, $C_1...C_k$. Thus a a-posterior (conditional category probability, $P(C|D)$) can be derived from a a-prior distribution of a category, $P(C)$, and a conditional probability of a document given the category, $P(D|C)$. It is the probabilities $P(C)$ and $P(D|C)$ that are learned in training.

$$P(C|D) = P(D|C) \cdot P(C) \tag{2.1}$$

From the BoW model a document D is a set of features: $t_1...t_m$ and the naivety of the algorithm is that all features are seen independently, which is a strong assumption about the data [26], however leading to the extention of Equation 2.1 to Equation 2.2.

$$P(C|D) = \prod_{i=1}^{m} P(t_i|C) \cdot P(C) \tag{2.2}$$

There are two variants of NB which are the multivariate Bernoulli model which sees the features as binary and thus ignores term frequency. The other is the multinominal model and takes into account the frequency of words but do not consider absent of features [26]. Callum and Nigam compares the differences of the multivariate and the multinominal NB and Metsis discuss which bayes to choose for what problem [3][29].

A systematic bias of NB is noted by Rennie et al. by showing that when the number of training instances differs among categories, the NB model tends to disfavor the lowest one. They suggest a solution by training a model with the same amount of exemplars per category [28], meaning the probability for a category, $P(C)$ is the same for each category.

Another problem with NB that is inherited by its independent assumption is called double counting, which means that if the BoW model is applied without n-grams (see Definition 2.1), concepts built by more than one word like, "ICA NARA", will get higher probabilities than concepts only including one word which can lead to unfavourable result [28].

However, positive arguments for NB is that a models category parameters can be efficiently computed in parallel and that the posterior vector can be used straight forward for label ranking or so called, soft categorisation [13].

Decision Trees

A decision tree is a rule based classifier that splits the feature space based on rules it has found in training data. Trees are straight forward and easy to understand, see Figure 2.1 because it follows a natural flow of decisions that can be turned into logical disjunctiones and simple **if .. then .. else** rules [14].

Building a tree is a greedy activity, allthough different optimisation can improve building time [13]. Using a single attribute split, starting at the root the algorithm chooses the most informative feature at each step, the feature with the highest entropy, gini-index or information gain [14][26].

A decision tree is efficient to use since look-up is $O(logN)$, where N is the number of datapoints. The problems with Decision Trees is that they can easily interpret noise as patterns, this can be prevented with pruning techniques and ensemble learning with forests [14].

Support Vector Machines

Support Vector Machines (SVM) are a quite new model founded by Vladimir N. Vapnik and later extended in 1995 by Cortes and Vapnik to include non-separable training data [30].

SVM's are linear classifiers that aims to find maximum margin hyperplanes that separates the data in its corresponding classes. By utilising a so called 'kernel-trick' an SVM can separate high dimensional data in a non-linear way,

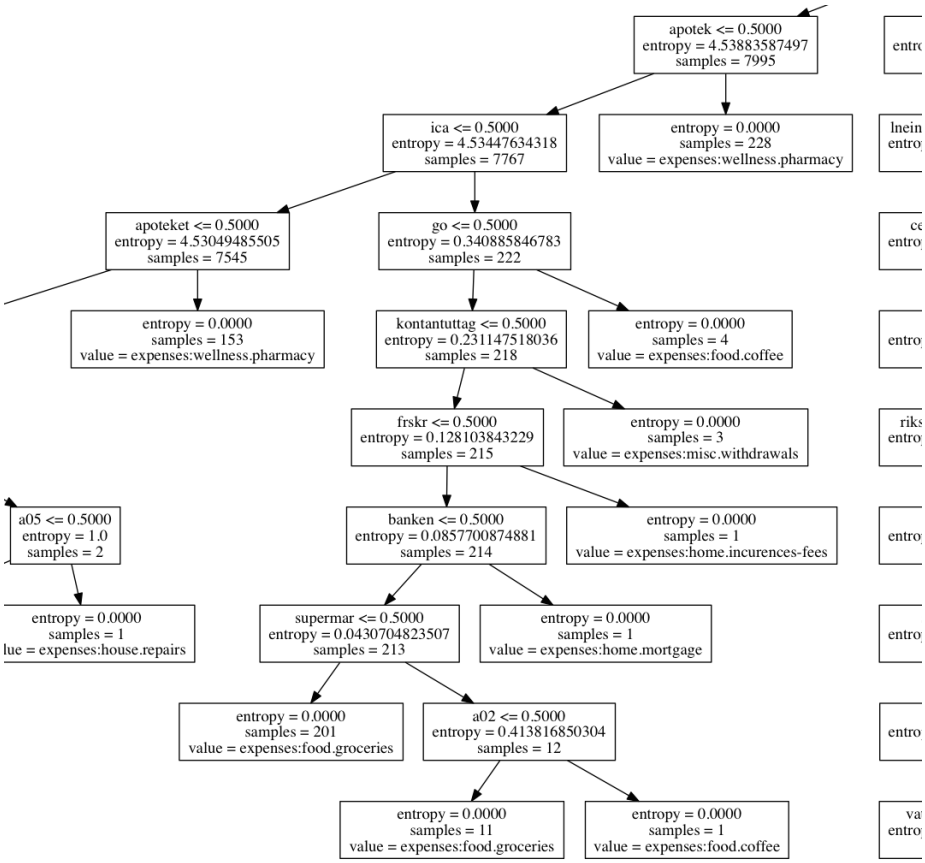

Figure 2.1: Decision Tree trained on Tink data. If a feature is present (eg, "ica"), the decision tree will take the right arrow if it is not present it will take the left arrow. This figure is generated from experiment 1 see Section 3.3.

depending on the kernel, see Figure 2.2 to see the decision boundary for a linear kernel and a rbf-kernel. Its an optimization problem and the SVM will learn from training data the weight vector, $\mathbf{w}$, and the bias, b, for each hyperplane so to maximize the margin, M, this means minimizing $\mathbf{w} \cdot \mathbf{w}$ under the constraint shown in Equation 2.3 where t_i is the target and x_i is the instance vector for instance i.

$$t_i(\mathbf{w} \cdot x_i + b) \geq 1 \qquad \forall i \tag{2.3}$$

The kernel-trick is used when the data cannot be linearly separable, the trick is to extend the dimension space but without the heavy computation it requires [14]. Because SVM's are binary classifiers they will split the data in a 'one-vs-all' fashion, the extension for a SVM to become a multi-category classifier is the creation of n hyperplanes for n category problems. There are much literature on how SVM works, what kernel to chose and how to tune parameters. Besides [30] a more comprehensible description for a beginner about SVM's can be found in [14].

SVM have been seen to perform very well in text classification context, one of the reasons is that the model handle sparse data gracefully [31]. However, learning a SVM is a computational heavy process and takes around O(n^2·features), where n is the number of documents in training, and is inherently hard to parallelize, so if the number of exemplars needed for training is high or the time requirements for training are tight, an SVM is not a good option [14].

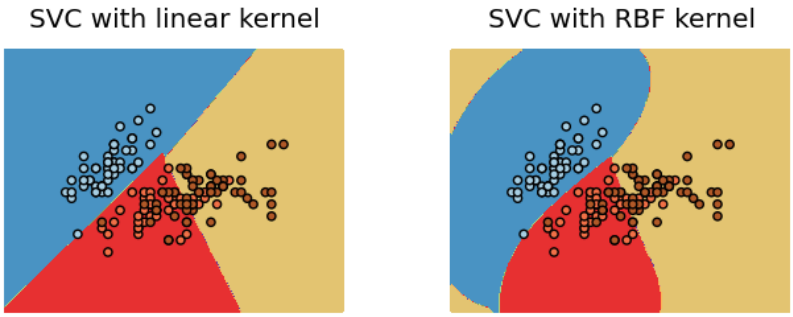

Figure 2.2: 3-category classification on the iris data set with a linear kernel and a rbf kernel.

Source: (http://scikit-learn.org/stable/modules/svm.html)

2.6 Fusion Schemes

Experimenting with data fusion can shortly be described to capture information about a concept from multiple sensors and to fusion these sensors signals to detect that concept more often than with only one sensor [9][10][32].

Interpreted to machine learning this can be seen as a form of ensemble learning, training multiple, possibly weaker classifiers on different parts of the data and then combine their result in order to improve final classification [9][14]. The

literature is not solely agree on why some fusion schemes performs better than other but Kittler et al. has found empirically that the sum rule is most resilient to estimation error, other fusion rules are explained thoroughly in [9][10].

Fusioning data can take place on several levels in data processing, Joshi et al. states five levels for fusioning of multimodal biometrics; sensor-, feature-, match-score-, rank- and decision level [10]. By the definitions stated in Section 2.1, late fusion and early fusion can be explained were the former fuses modalities in feature space, feature-level, while the latter fuses modalities in semantic space [1].

2.6.1 Early fusion

Fusioning at feature level or so called early fusion (see Definition 2.1), needs to balance the best data representation for all modalities features. From Equation 2.4 and 2.5 x, y and z comes from their individual modality sources X, Y and Z, transformed in the same transformation function ϕ and concatenated to feature vector $\mathbf{F}$ so that all features belongs to the same feature space H.

Because the features has to be represented in the same way, potential information loss can be inferred. However, the advantage with early fusion is it only needs one learning phase [1].

$$x \in X, y \in Y, z \in Z \tag{2.4}$$

$$\mathbf{F} = [\phi(x), \phi(y), \phi(z)] \in H \tag{2.5}$$

2.6.2 Late fusion

The advantages with late fusion is then consequently to meet the disadvantages from early fusion. By choosing to fusion at a later stage, models can be chosen that will best fit a specific modality's representation.

In Equation 2.6 each modalities are transformed to belong to each corresponding feature space. Each models outcome y will be combined by a fusion rule θ and output a representation in feature space H_4.

$$\phi_1(x) = y_1, \phi_2(y) = y_2, \phi_3(z) = y_3 \in H_1, H_2, H_3 \tag{2.6}$$

$$\theta(y_1, y_2, y_3) \in H_4 \tag{2.7}$$

The idea with late fusion is also based on the finding that different models miss-classify different data points, based on model inherit bias [12]. The disadvantages with late fusion is "the potential loss of correlation in mixed feature space" [1]. For example, modalities that are conditional dependent on each other will not be learned.

Chapter 3

Implementation

3.1 Data

Table 3.1: Bank transaction example data

	Description	Amount	Date	Category	User modified	First Category
1	PIZZA PLANET	-99	2013-12-15 12:00:00	restaurants	0	restaurants
2	FOLKBAREN AB	-56	2012-04-27 12:00:00	restaurants	0	restaurants
3	FOLKBAREN	-62	2013-11-07 12:00:00	bars	1	restaurants
4	FOLKBAREN	-220	2013-04-09 12:00:00	restaurants	0	restaurants
5	Pressbyran 5173 Karl	-18	2012-10-07 12:00:00	coffee	0	NULL
6	Pressbyran 8179	-53	2013-01-31 12:00:00	alcohol-tobacco	1	coffee
7	Pressbyran 8650	-36	2013-10-27 12:00:00	public-transport	1	coffee
8	PRESSBYRAN 420	-378	2013-12-29 12:00:00	public-transport	1	coffee
9	Pressbyran 5190 Mari	-790	2012-11-23 12:00:00	public-transport	1	coffee
10	STATOIL VARBERG	-23	2012-10-17 12:00:00	coffee	1	NULL
11	STATOIL TUMBA 2	-30	2013-11-29 12:00:00	coffee	1	car
12	STATOIL RÅSUNDA	-689	2012-06-30 12:00:00	car	0	NULL
13	STATOIL SANNARP	-22	2013-11-18 12:00:00	car	0	NULL
14	Systembolaget Jönköpin	-299	2012-11-28 12:00:00	groceries	1	NULL
15	EKLANDA PIZZERIA	-260	2012-10-05 12:00:00	groceries	1	NULL
16	Vero Mat & Cafe	-214	2013-11-16 12:00:00	restaurants	1	coffee
17	BALLBREAKER KUN	-95	2013-05-03 12:00:00	restaurants	1	culture
18	BULLENS CAFE	-65	2013-11-04 12:00:00	restaurants	1	coffee
19	BULLENS CAFE	-60	2013-12-03 12:00:00	coffee	0	NULL

Table 3.1 shows 19 transactions from Tink, where **Category** is the final category the transaction has right now and **First Category** is the category it had before it changed to **Category**. **User modified** means that a user actively has changed that transactions category.

3.2 Experimental setup

By choosing to see a transactions description, amount and date as three different modalities experiment with the two different fusion schemes, early fusion and late fusion will be made. Early fusion treats the three different modalities as binary vectors and simply concatenated before classification. Different concate-

nation combination will be examined to see how the different modalities behave in relation to each other.

The second scheme is late fusion where each modality train one model separately. The result from each classifier is then added together and serves as input vector to a new classifier. The classifier that performs best on respective modality from the early fusion tests will be used in the late fusion experiments.

3.2.1 Requirements & Characteristics

Tink's requirements are a fast and scalable solution to classify a large amount of transactions with high precision and high recall.

The data availability are high in Sweden but non-existent in non yet explored markets.

Precision and recall are not measured by Tink on their existing system. However, Tink's own estimation of its performance are 82% classification accuracy, based on transactions spanning over both income and expenses and on amount over 10k SEK.

An estimation on my data set is 85% classification accuracy of 338.654 transactions. The estimation is based on the assumption that if a user has changed the category, the user are right and Tink is wrong.

Tink also requires a model that rather makes no decisions (do not categorize) than making wrong decisions, meaning false positives has to minimised. This is due to usability issues or reversed *trust bias*, giving the impression that a user cannot trust the systems categorisations. For example if a transaction's description and amount are "gröna lund", 500 SEK this should not be categorized *home.electronics*.

3.2.2 Data set

Data are collected from Tink's Swedish systems.

The data set will only include *expenses* under 10,000 SEK (there are also *income* and *transfer* and *expenses* over 10,000 SEK). See some data instance examples in Table 3.1

There are 45 categories in the expense segment divided on 8 first level categories, see Figure 3.1, 10 of these are 'other' and are not included in this thesis tests, neither are *gifts* and *outlays*. There is also a category called *uncategorized*, this category on the other hand has to be trained since some transactions should always be uncategorized, for example transactions with description *KLARNA* and *PREL. KORTKÖP*.

If a category is labeled *uncategorized* by the system this will be noted by the user who can then manually label that transaction, when that happens, the new label will be sent to Tink who mark it as categorized by user and its new category, see Table 3.1. A total number of 35 categories are used in all tests and considered existing in a flat hierarchy, see Figure 3.2. Although there exists a category hierarchy, this is not exploited in this thesis since previous studies shows unsatisfactory results [6] although it is added to future research proposals.

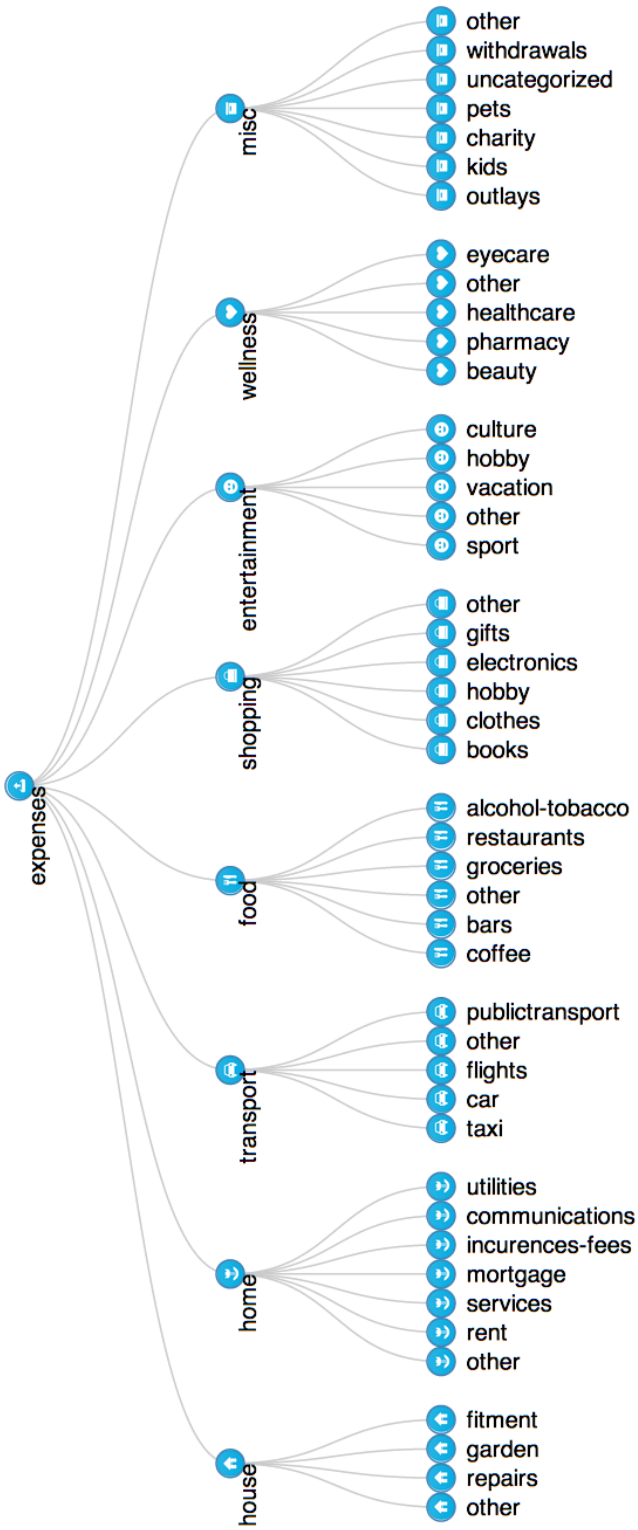

Figure 3.1: The categorisation tree currently maintained by Tink.

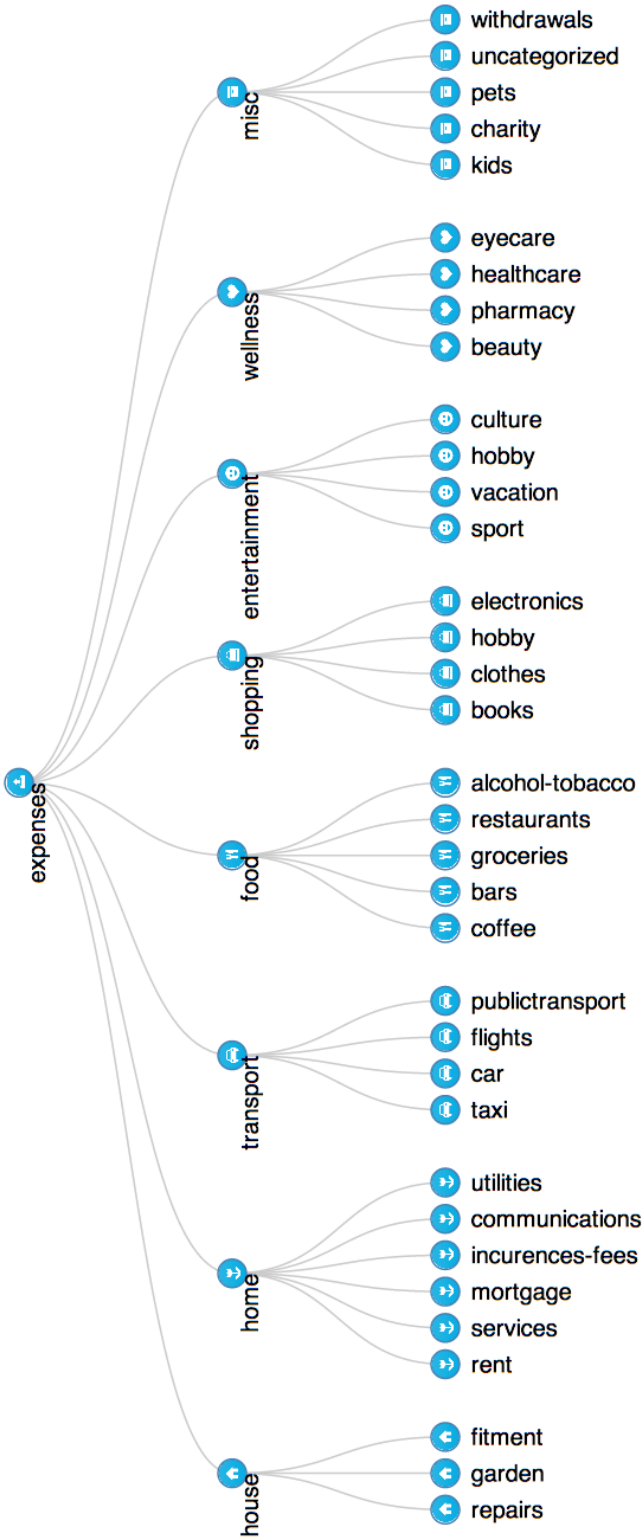

Figure 3.2: The modified categorisation tree used in this thesis.

3.2.3 Data representation

The information given from a bank transaction is a short description of the merchant (5-25 letters depending on the bank), amount and a date when the purchase was registered by the bank (no time is given by the bank or registered by Tink).

There is no semantics in the description[1], it is a name of the merchant and on some there is also a location[2] which could be helpful in some use cases, however, in this thesis it will be omitted and left to future research. The topology of the data is therefore low [12]. The only text a model can learn from a description is therefore the name of the merchant, fortunately some types of merchants has named themselves after its business eg. "PIZZA PLANET". Depending on what the merchant sell or do the exclusivity in the data are quite high [12]. Meaning the BoW model can be applied since it ignores grammatical structure.

Because the description is short and does not vary in size, ordinary vector feature weighting like normalized term frequency cannot be relied on. The tf-idf weight approach is also in some sense wrong since it gives terms occurring in many documents low weight, but we want terms that occurs in many document. For example, if many transactions with one occurrence of "HM" in each (low term frequency), and all transactions are categorized to clothes, then "HM" is a good word for the clothes category to learn. However, as with the word 'ab' which occurs in transactions represented by all categories (high document frequency), this is non-discriminant and should not be learned by any category. For merchants that exists in many areas, like Ica (Ica the bank, Ica publishers, Ica grocery store etc) then each category should learn the term "Ica". To address the problem with high frequency word across categories, a stop-word list that is manually created can instead be used, a stop-word list is a list that removes word from the feature space before learning. Because of the special characteristics of the Swedish banks description, the stop-word list is very short, around 10 in size despite the list of location, see Section 3.2.4.

We are dealing with high dimensional data although the size of the feature vector will not go beyond the number of unique merchant and their different representation for different banks in Sweden.

3.2.4 Dimension Reduction

To reduce the feature dimension a stop-word list and stemming is applied.

Stemming words reduces the feature space additionally by transforming words according to its grammatical root, for example words in plural are stemmed to its singular correspondence. The stemming library used in this thesis is from the open source Snowball project with Swedish as language [33].

The stop-word list contains geographical locations in Sweden and small non-informative words, like 'ab', 'kr' and 'www'. Removing locations is not only to reduce the feature space but also because the model should not find patterns on descriptions from the same geographic location. A McDonalds restaurant in Säffle are not to be considered similar to Åhlens in Säffle. The location

[1]It is different formats on description in other countries and it is possible that it could change in the future in Sweden too.

[2]It is the merchant who decides what the description would say.

information should however be registered somewhere else and not completely thrown away but this is out of the scope for this thesis.

3.2.5 Training and Testing

All comparisons are made on the same non overlapping training and testing data set. In the early fusion experiments the different concatenation schemes are cross validated separately but the data points are the same.

All experiments are tested on exemplars that has not been seen in training.

The early fusion setup uses a non overlapping 5-fold Stratified cross validation of non overlapping data to set the training/test split to 80/20. The reason behind a stratified sampling is to address the inheritent bias in NB [28].

The late fusion schemes training and testing are a little bit different. It is instead a nested k-fold cross validation because there are two levels of learning models. So the data splits are 3x3 stratified cross validation, meaning that a final split of 33%/44%/22% training/testing&training/finalTesting is used to ensure that the final testing set is 22% of the whole data set.

3.2.6 System setting

The code is written in Python 2.7 with the open source library Scikit-Learn.

Scikit-learn provides all different kind of tools for machine learning applications like, algorithms for classification, clustering and regression, pre-processing tools and much more [34].

The implementation is executed on a Mac OS X 10.9.2 2GHz Intel Core i7 and 8GB memory.

Important methods from scikit-learn that has been used are:

- cross_validation.StratifiedKFold()[3] - This cross-validation object is a variation of KFold, which returns stratified folds. The folds are made by preserving the percentage of samples for each class.
- metrics.precision_recall_fscore_support()[4] - Compute precision, recall, F-measure and support for each class.
- CountVectorizer()[5] - Represents a corpus as a feature vector.
- predict_proba(X)[6][7][8] - Predicts a probability vector on all classes for X.

[3] http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedKFold.html#sklearn.cross_validation.StratifiedKFold

[4] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

[5] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[6] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB.predict_proba

[7] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict_proba

[8] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.predict_proba

Naive-Bayes - Bernoulli Bayes The Naive Bayes classifier is a Bernoulli Bayes which is a multivariate Bernoulli Bayes and it uses a Laplace smoother by default, to prevent multiplying with zero if an attribute is missing. This thesis only uses the scikit-learn BernouliNB with default prameters.[9][10]

Decision Tree Scikit implements an optimised version of the CART algorithm [14] for decision trees.

This thesis has used the default parameter settings for decision trees where the split criteria is measured on gini-index [14]. Scikit does not implement pruning, but a overfitted tree could possibly be controlled by setting the parameters: minimum-split and minimum-leafnode. This has been omitted in this thesis. [11]

SVM - SVC with rbf kernel The kernel choice and parameter settings are crucial for a good SVM performance. In this thesis there has been no further investigation in the SVM optimisation than fixing a RBF-kernel and performed a GridSearch to find the best parameters for C and gamma. Where the C parameter determines the smoothness of the decision surfaces and the gamma parameter determines the importance of each training example. The final SVM settings used in this thesis is a RBF-kernel with C=100 and gamma=0.01.

The probability parameter are set to TRUE, meaning in Scikit-learn that a probability vector per class can be used when calling predict_proba(). The probabilities are calculated using Platt's scaling [35].[12]

If SVM is the choice of further usage, this settings would preferable be tuned more carefully, and also be tested on multiple kernels.

3.3 Experiment 1 - Early vs no fusion

As stated in Definition 2.1, early fusion is when combining multiple modalities (different data sources) into one representation before classification. The drawbacks with this is the potential removal of information in the pre-process step. Looking at the nature of the other modalities, fitting them in a vector model just as they are is unreasonable, at least for the first one, *amount* + *date* (10.000 + 12*31 additional features).

The amount mode are therefore discretisized on 25 intervals with varying value [5, 10, 11, 12, 13, 14, 15, 25, 30, 35, 40, 50, 60, 70, 80, 100, 110, 140, 200, 250, 300, 400, 500, 1000, 10000]. The reasons behind the interval splits are based on binning all the transactions amount and even out the distribution, this is open for improvements.

To binarize the text, the description has first been pre-processed by replacing all non alpha characters with white space, (& and -) are replaced with nothing to emphasis that the words are really one word. Then the words are stemmed and transformed to lowercase letters.

[9] http://scikit-learn.org/stable/modules/naive_bayes.html

[10] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html

[11] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[12] http://scikit-learn.org/stable/modules/svm.html#scores-and-probabilities

Table 3.2: Concatenation schemes for early fusion

No fusion
Text
Amount
Weekday
Day of Month
Concatenation Schemes
Text + Amount
Text + Weekday
Text + Day of month
Text + Amount + Weekday
Text + Amount + Day of Month
Text + Amount + Weekday + Day of Month
Text + Weekday + Day of Month

The date is also transformed to its corresponding day of the week and a different feature for the day of the month. Leading to a date range of monday to sunday (7 features) and a different month range between m1-m31 (31 features), the day of the month is concatenated with an 'm' to easily distinguish that feature from amount.

Depending on the test and what modalities are used, a corpus is created on all the newly concatenated instances in the test set and fitted to a feature vector where each feature corresponds to a word in the corpus.

The different concatenation schemes are displayed in Table 3.2.

3.4 Experiment 2 - Late vs stacked and no fusion

For the late fusion schemes, separate classifiers has been trained on separate modalities see Figure 3.3. The text modality are trained just like the concatenation scheme for only Text in experiment 1, see Table 3.2. Thus the usage of SVM are chosen because of its performance in the early fusion experiments.

The amount and date modalities are trained with an NB because no classifier outperforms the other in the no fusion experiments and this is chosen because of its speed.

The amount modality are rounded to closest one (float to int) and the date modality has two features, one is the day in month (int) and the second is the month (int).

To compare the result from the late fusion scheme, the same tests has been conducted on the first SVM of the text modality as for the second SVM, to see what difference there is in late vs early fusion. But because of the smaller training set than for early fusion for the first SVM, a second comparison test has been made. A late fusion classifier that ignores to fuse after the first training, thus the final SVM will only learn probability vectors from one modality, like the early vs no fusion, resulting in a kind of stacked training step, see Figure 3.4.

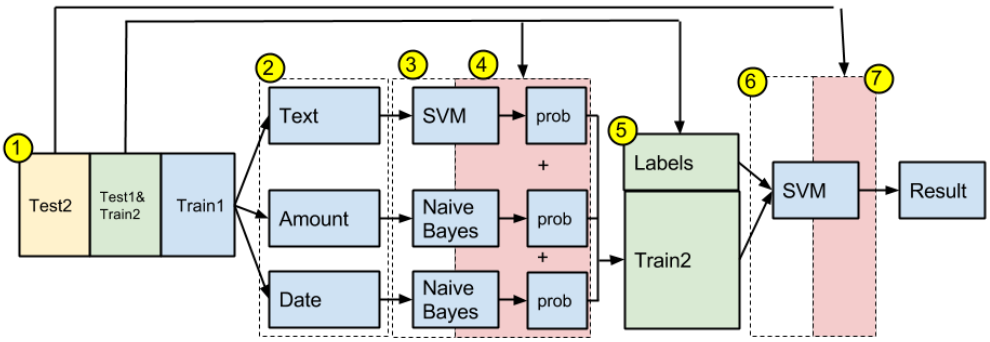

Figure 3.3: **Late fusion Scheme**. (1) Data from Tink System split on Train1/Test1&Train2/Test2. (2) Feature Extraction. (3) Train each classifier separately with training data set 1, *Train1*. (4) Test models with test data set 1, *Test1*, fusion result by adding each probability vector. (5) The fusioned probability vector is now training data set 2, *Train2*, for the final classifier. (6) Train model with *Train2*. (7) Test final classifier on test data set 2, *Test2*

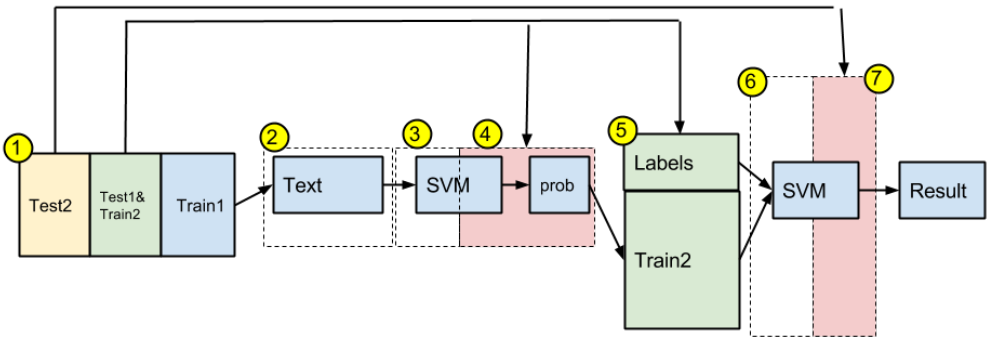

Figure 3.4: **Stacked Scheme**. (1) Data from Tink System split on Train1/Test1&Train2/Test2. (2) Feature Extraction. (3) Train a classifier with training data set 1, *Train1*. (4) Test model with test data set 1, *Test1*. (5) The resulting probability vector is now training data set 2, *Train2*, for the final classifier. (6) Train model with *Train2*. (7) Test final classifier on test data set 2, *Test2*

Chapter 4

Results

4.1 Experiment 1 - Early vs no fusion

From Table 4.1 and 4.2 we can see that all three models perform the best when trained and tested on only the text modality (no fusion). The SVM and the Decision Tree shows similar performance but the SVM is better in both precision and recall.

The Text+Weekday concatenation scheme performs better than the other concatenation schemes, this could simply be explained because the weekday modality concatenated only consist of 7 features.

The no fusion schemes; Day of Month, Weekday and Amount (no fusion), shows worse or close to random prediction. This is interpreted from the knowledge that all training data are labeled based on text and not on amount or date (see Section 4.3).

Table 4.1: Average precision in percent for early fusion. It is a 5-fold stratified sampling with 300 samples from each category.

No fusion	SVM	NB	Tree
Text	**93.4**	**90.5**	**93.1**
Day of Month	3.5	3.6	3.4
Weekday	0.8	0.8	0.8
Amount	6.1	6.3	6.0
Concatenation Scheme			
Text + Amount + Weekday	88.0	83.7	81.7
Text + Amount + Weekday + Day of Month	86.8	81.5	78.8
Text + Amount	89.7	85.7	87.9
Text + Weekday	91.5	87.8	91.8
Text + Day of month	88.7	84.3	87.8
Text + Amount + Day of Month	87.5	82.4	80.4
Text + Weekday + Day of Month	87.9	83.4	85.0

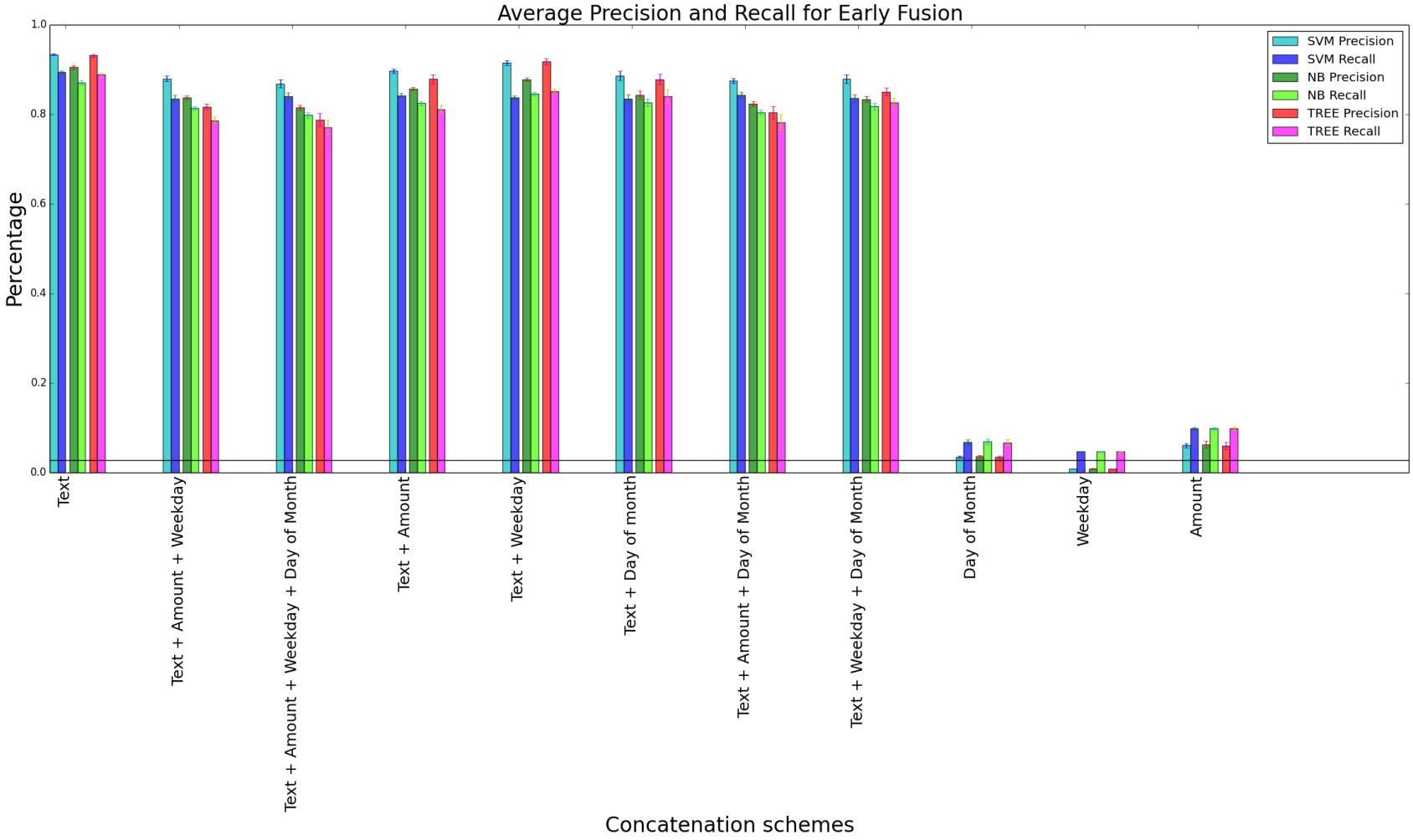

Figure 4.1: Average precision and recall for early fusion. All schemes are presented in Table 3.2 and discussed in Section 3.3. The horizontal line represent random precision and recall.

Table 4.2: Average recall for early fusion. It is a 5-fold stratified sampling with 300 samples from each category.

No fusion	SVM	NB	Tree
Text	**89.5**	**87.1**	**88.9**
Day of Month	6.8	6.9	6.7
Weekday	4.7	4.7	4.7
Amount	9.8	9.8	9.9
Concatenation Scheme			
Text + Amount + Weekday	83.5	81.4	78.6
Text + Amount + Weekday + Day of Month	84.0	79.8	77.1
Text + Amount	84.2	82.5	81.1
Text + Weekday	83.7	84.6	85.1
Text + Day of month	83.5	82.6	84.1
Text + Amount + Day of Month	84.3	80.4	78.2
Text + Weekday + Day of Month	83.5	81.8	82.6

4.2 Experiment 2 - Late vs stacked and no fusion

From Table 4.3 and Figure 4.2 we can see that the stacked SVM performs exactly the same as late fusion. This means that the impact of the amount and date modalities are non. Both stacked SVM and late fusion performs marginally better than no fusion (singe SVM on text modality) this is interpreted as dependent on the size of the training data. Comparing to the early fusion experiment in Section 4.1, stacked SVM and late fusion performs with almost the same average precision and recall as for no fusion on only the text modality.

Table 4.3: Average precision and recall for Experiment 2, measured in percent.

Test	Precision	Recall
Late fusion	92.5	83.6
Stacked SVM	92.5	83.6
No fusion	91.1	84.5

We can also see some anomalies in Figure 4.3 on the categories *culture* for no fusion and *restaurants* for stacked SVM and late fusion. This is because of the one-vs-all idea of an SVM, so when there are instances with features not existing in feature space one category will become a sinkhole, it is interesting that the stacked SVM learn this in its final predict, however it moves the sinkhole to another category. For Naive Bayes we can target the sinkhole to the appropriate category *uncategorized* by setting a threshold on prediction with high uncertainty. Similar sinkholes can be found in the early fusion experiments too, see Table 4.4 for the *culture* category.

4.3 Discussion of experiment result

In the results we can see that the no fusion scheme on the text modality are superior to all other experiments.

The reason for this result can be explained that all transactions in the data are labeled by the Tink system, meaning it is labeled by only regarding information from text. Thus the patterns to be learned will only be found in the text modality.

The same arguments are found in the late fusion experiments where we can see that fusion the two modalities amount and date shows no affect at all. It is also slightly worse than the early fusion experiment on text. However, comparing the stacked model and the late fusioned model we can see that they perform exactly the same, compare this to the early fusion experiments this means that at least late fusion does not decrease performance. This could be useful if we can show that as soon as individual modality performance rise, so does the fusioned result, thus the programmer can easily automate an individual threshold to switch on and off late fusion depending on the individual modality results and the overall fusioned result.

If the data labeling procedure should be different, with more exemplars that has been labeled by users, this should be seen improve model performance on

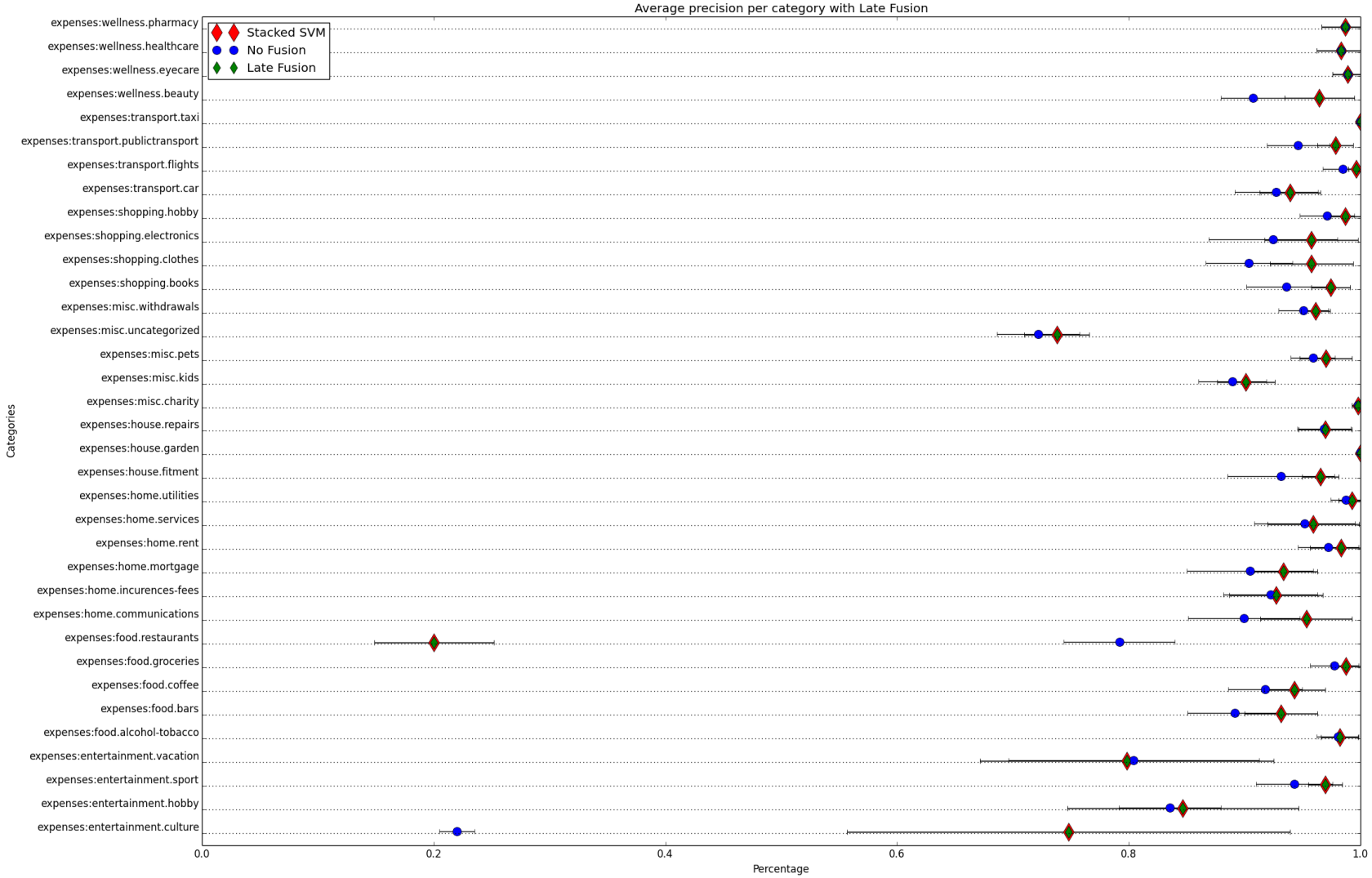

Figure 4.2: Average precision per category for late fusion. The standard deviation is represented as a horizontal line on each data point.

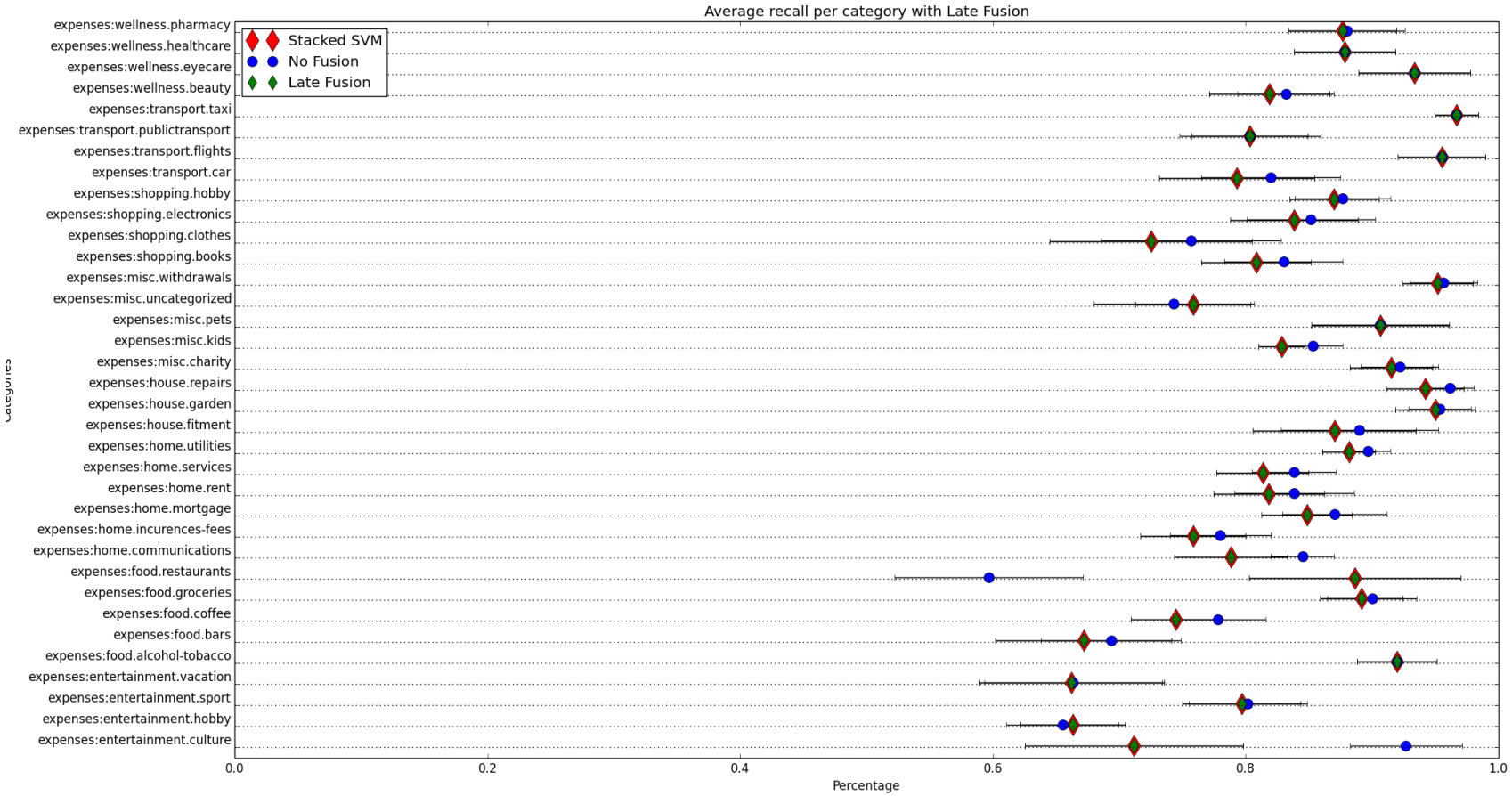

Figure 4.3: Average recall per category for late fusion. The standard deviation is represented as a horizontal line on each data point.

only one modality, then this could indicate an improvements of both early and late fusion combination.

Because of the fact that the text modality performs so well a more thoroughly study has been made on the text modality in Section 4.4 where the data labeled by users are tested as well to possibly motivate a change of data sampling for learning.

4.4 Case study of the Text Modality

The Results 4.1 and 4.2 shows strong indication of a no use of other modalities then text for learning bank transactions. However, the data used for training and testing has only a low percentage of user modified categories. This result has led to a hypothesis that model performance would boost if trained on data that has been categoriesd by a human, preferable a human that knows the purchase behind the transaction. Users can always re-label a category in the Tink application. There could be a range of reasons why a re-categorisation happens. One reason is if the Tink system was uncertain and categorized to **uncategorized**. Another reason could be more personally, you do not want that transaction to be in the "wrong" category. If you are a smoker or a drinker, you may want to manipulate the statistics so you do not get it in plain text how expensive the bad habit is, see Table 3.1 row 14.

For some categories it is not crystal clear in what category purchases should be filed under, maybe different people have different categorisation strategies.

Another reason is that you correct the system by changing the category to where it is supposed to be, based on the purchase, meaning Tink's prediction was wrong, see Table 3.1 row 3 for an example.

However, the number of user labeled data is much less than Tink labeled data. I believe three reasons can explain this. Either users are satisfied with its transactions categorisation or unsatisfied but cannot be bother to manually change every transaction that is wrong or finally what is referred to as *trust bias*, Trust bias is often seen in search engines ranking, meaning users trust the system to produce good ranking so the user only check the first top result, which has to be countered for when measuring clicks. In this setting, trust bias could happen if Tink shows well performance in the initial step with the user, leading to that the user then trust the categorisation and does not control Tink's future categorisation. However, in the Tink setting this trust bias can just as well be reversed, if a user detects an anomaly in the categorisation, its trust towards the system will be lost, leading to the users need to control the system or worse, the user stops using the service.

A proposed solution would be to *force* a number or fraction of transactions to be manually categorised by users, even if they should correctly be categorised by Tink's system. With that solution you will get a stream of labeled data that would potentially be right categorised based on the purchase, (this could be a vulnerability, shilling-attack).

But how many and which transactions are needed to be categorised in order to improve each categories precision and recall?

Looking at the table 4.4 it is not number of transactions that improves precision, neither number of features. So it has to be that some features exists more often in the sampling of 300 transactions per category in the Tink labeled

Table 4.4: Difference between user labeled data and Tink labeled data. The columns SVM User Labeled and SVM Tink Labeled are precision on an SVM performance on only text modality of user labeled and Tink labeled data respectively, measured in percent. Column **User Labeled** is number of instances that are user labeled, remember that the number of Tink labeled instances are 300 on all categories. Column **User Labeled Features** and **Tink Labeled features** is the number of features found in the respective data set

Categories	SVM User Labeled	SVM Tink Labeled	User Labeled	User Labeled Features	Tink Labeled features
expenses:entertainment.culture	37	31	289	254	138
expenses:entertainment.hobby	30	86	197	171	135
expenses:entertainment.sport	74	96	193	148	102
expenses:entertainment.vacation	49	92	157	166	177
expenses:food.alcohol-tobacco	37	98	300 or more	286	104
expenses:food.bars	40	92	300 or more	288	157
expenses:food.coffee	44	94	300 or more	288	202
expenses:food.groceries	71	99	300 or more	231	221
expenses:food.restaurants	27	83	300 or more	384	318
expenses:home.communications	63	94	300 or more	173	102
expenses:home.incurences-fees	73	94	300 or more	185	137
expenses:home.mortgage	39	96	300 or more	117	66
expenses:home.rent	61	97	273	130	101
expenses:home.services	21	98	95	100	96
expenses:home.utilities	60	99	147	96	69
expenses:house.fitment	46	97	160	101	94
expenses:house.garden	80	100	8	8	50
expenses:house.repairs	42	97	48	55	68
expenses:misc.charity	70	99	43	36	48
expenses:misc.kids	49	93	218	157	103
expenses:misc.pets	82	98	50	45	65
expenses:misc.uncategorized	32	71	300 or more	140	94
expenses:misc.withdrawals	43	95	113	93	51
expenses:shopping.books	52	95	173	110	85
expenses:shopping.clothes	39	94	300 or more	203	167
expenses:shopping.electronics	28	96	185	97	87
expenses:shopping.hobby	36	99	35	33	81
expenses:transport.car	53	95	300 or more	176	148
expenses:transport.flights	0	100	7	14	25
expenses:transport.publictransport	68	97	289	152	129
expenses:transport.taxi	97	100	30	25	33
expenses:wellness.beauty	33	100	81	93	138
expenses:wellness.eyecare	43	99	27	30	42
expenses:wellness.healthcare	84	99	61	62	49
expenses:wellness.pharmacy	88	99	45	46	132

Feature	Frequency in user labeled data	Frequency in Tink labeled data
pub	2	4
and	2	4
kung	0	5
leary	0	5
subway	1	5
the	2	5
sushi	1	6
pizz	4	7
mc	1	8
donald	1	8
bar	5	9
pizzeri	2	10
restaurang	5	16
king	1	17
burg	1	17
max	1	19
mcdonald	8	54

Table 4.5: Top occurrence features found in the restaurant category.

data set includes more transactions with specific features.

When examinining the data more thoroughly, some merchants are more common than other (see Table 4.5), *max*, *pizzeri*, *mcdonald*, etc. These features has to be considered ultra important to classify, since they occur most frequently. However, a merchant seen twice in the data is just as important to learn, because categorising low frequent merchants would potentially lead to higher satisfaction for a single user.

Because of the date and amount modality, enough data has to be sampled from each category that accurately represent that categories amount distribution and date distribution, this can only be done in a supervised learning environment if the training data are labeled to account for these modalities too.

Another argument that would help both the collection of user labeled data and possible user experience is the usage of ranked labels below a certain treshold. Classification as an application for humans are suppose to help humans divide the information in manageable areas, categories. But what should a system do when it is uncertain of its decision? A proposal is to make use of the users (user feedback) who can manually categorize that data instance and send that back into the system for the system to improve. Even though humans are good categorizer, a hiearchial categorisation tree of 45 categories means 45 decisions to make, then the application is no longer a helpful tool. However, if the system and the user could have an easy dialog where the system is frank stating its uncertainty and ask the user to help on its top three candidates for a solution. Then the user can help the system in one single button press with a reduced decision space.

Chapter 5

Conclusion

5.1 Summary

Understanding the whole process of an application using a machine learned model is not a silver bullet. All application has its tweaks depending on the data, the availability of the data and the nature of the problem the model should solve.

Depending on where an application is in its life cycle, different machine learning approaches could be taken. In the stage where this thesis has been made, a lot of labeled data has been available, leading to a proposed solution of supervised learning is applicable. All three models shows good performance in both precision and recall, but the SVM and Decision Tree has a slightly higher accuracy over all.

What argues for the Naive Bayes model is that it is parallelizable and by default probabilistic meaning it enables the straight forward usage of thresholding a models uncertainty and returning a ranked list of suggested categories.

It is possibly to return a ranked result from SVM and Trees too in Scikit-learn, but if the algorithms are written from scratch, the Naive Bayes are to be recommended due to its implementation simplicity.

This thesis found no improvements of modality fusion in neither early nor late fusion. However, in the late fusion experiments the fusion step did not decrease performance which could indicate that an implementation in production could be possible with the fusion step to automatically be switch on after a certain validity threshold has been reach by individual modality performance and the overall fusioned performance.

The experiments shows that bank transactions are very well suited for machine learning purposes and that a supervised learning model can give acceptable results on test data. However, the thesis also notice the same problems as for many similar applications that deals with high dimensional data the importance of having enough training and testing data per feature.

This thesis propose a change in the user feedback flow, utilizing the users as experts of its own spending. By introducing a more controlled user categorisation flow the learning system could both counter for the curse of dimensionality and to collect more training data that has been classified on possibly all of its modalities. Since Tink's application has a large user group this cannot be seen as

expensive nor inefficient compare to other applications without the availability of human experts.

5.2 Future Work

To change the feedback loop and learning the *right* features, an interesting approach to investigate would be online-learning with user feedback or pool-based active learning. This thesis investigates learning on a global level, however, learning patterns on an individual level could be interesting, seeing transactions as a stream with spending patterns changing over time.

Another suggestion for further research is exploring the possible conditional dependencies on individual features. Given feature 'Pressbyrån', what is the probability of a category when the amount is 15 SEK?

Less labeled data are something Tink has to deal with if they open on other markets, then techniques like different clustering methods, training a model from completely different data tested on a small labeled set or infuse more data in the training stage would be interesting to see. However, measures for validating these techniques has to be developed.

Another idea to investigate is the fact that this thesis chopped important data like merchant location. If users where to add its location to the application, classification could potentially improve knowing the location of a transaction.

Bibliography

[1] Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. "Early versus Late Fusion in Semantic Video Analysis". In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*. MULTIMEDIA '05. Hilton, Singapore: ACM, 2005, pp. 399–402. ISBN: 1-59593-044-2. DOI: 10.1145/1101149.1101236. URL: http://doi.acm.org/10.1145/1101149.1101236.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 2nd ed. Prentice Hall. ISBN: 0131873210.

[3] Vangelis Metsis Telecommunications and Vangelis Metsis. "Spam Filtering with Naive Bayes – Which Naive Bayes?" In: *Third Conference on Email and Anti-Spam (CEAS*. 2006.

[4] Haiyi Zhang and Di Li. "Naive Bayes Text Classifier". In: *Granular Computing, 2007. GRC 2007. IEEE International Conference on*. 2007, pp. 708–708. DOI: 10.1109/GrC.2007.40.

[5] Bo Pang and Lillian Lee. "Opinion Mining and Sentiment Analysis". In: *Found. Trends Inf. Retr.* 2.1-2 (Jan. 2008), pp. 1–135. ISSN: 1554-0669. DOI: 10.1561/1500000011. URL: http://dx.doi.org/10.1561/1500000011.

[6] Susan Dumais and Hao Chen. "Hierarchical Classification of Web Content". In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '00. Athens, Greece: ACM, 2000, pp. 256–263. ISBN: 1-58113-226-3. DOI: 10.1145/345508.345593. URL: http://doi.acm.org/10.1145/345508.345593.

[7] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. "Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections". In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: ACM, 2008, pp. 91–100. ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367510. URL: http://doi.acm.org/10.1145/1367497.1367510.

[8] T. Deselaers, L. Pimenidis, and H. Ney. "Bag-of-visual-words models for adult image classification and filtering". In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. 2008, pp. 1–4. DOI: 10.1109/ICPR.2008.4761366.

[9] Josef Kittler et al. “On Combining Classifiers”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 20.3 (Mar. 1998), pp. 226–239. ISSN: 0162-8828. DOI: 10.1109/34.667881. URL: http://dx.doi.org/10.1109/34.667881.

[10] Tejas Joshi, Somnath Dey, and Debasis Samanta. “Multimodal Biometrics: State of the Art in Fusion Techniques”. In: *Int. J. Biometrics* 1.4 (July 2009), pp. 393–417. ISSN: 1755-8301. DOI: 10.1504/IJBM.2009.027303. URL: http://dx.doi.org/10.1504/IJBM.2009.027303.

[11] Mongi A. Abidi and Rafael C. Gonzalez. *Data Fusion in Robotics and Machine Intelligence*. San Diego, CA, USA: Academic Press Professional, Inc., 1992. ISBN: 0-12-042120-8.

[12] Peter Jackson and Isabelle Moulinier. *Natural Language Processing for Online Applications : Text Retrieval, Extraction and Categorization*. Amsterdam, NLD: John Benjamins Publishing Company, 2007. ISBN: 9789027249920.

[13] Fabrizio Sebastiani. “Machine Learning in Automated Text Categorization”. In: *ACM Comput. Surv.* 34.1 (Mar. 2002), pp. 1–47. ISSN: 0360-0300. DOI: 10.1145/505282.505283. URL: http://doi.acm.org/10.1145/505282.505283.

[14] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. 1st. Chapman & Hall/CRC, 2009. ISBN: 1420067184, 9781420067187.

[15] G. Salton, A. Wong, and C. S. Yang. “A Vector Space Model for Automatic Indexing”. In: *Commun. ACM* 18.11 (Nov. 1975), pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: http://doi.acm.org/10.1145/361219.361220.

[16] Christopher Meek Donald Metzler Susan Dumais. “Similarity Measures for Short Segments of Text”. In: 4425 (), pp. 16–27.

[17] *The Reuters 21578 Data Set*. [Online; accessed 19-05-2014]. URL: http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[18] *The 20 News Group Data Set*. [Online; accessed 19-05-2014]. URL: http://qwone.com/~jason/20Newsgroups/.

[19] Bharath Sriram et al. “Short Text Classification in Twitter to Improve Information Filtering”. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’10. Geneva, Switzerland: ACM, 2010, pp. 841–842. ISBN: 978-1-4503-0153-4. DOI: 10.1145/1835449.1835643. URL: http://doi.acm.org/10.1145/1835449.1835643.

[20] Yiming Yang and Jan O. Pedersen. “A Comparative Study on Feature Selection in Text Categorization”. In: *Proceedings of the Fourteenth International Conference on Machine Learning*. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420. ISBN: 1-55860-486-3. URL: http://dl.acm.org/citation.cfm?id=645526.657137.

[21] Wen-Tau Yih and Christopher Meek. “Improving similarity measures for short segments of text”. In: *AAAI*. Vol. 7. 2007, pp. 1489–1494.

[22] Simon Tong and Daphne Koller. "Support Vector Machine Active Learning with Applications to Text Classification". In: *J. Mach. Learn. Res.* 2 (Mar. 2002), pp. 45–66. ISSN: 1532-4435. DOI: 10.1162/153244302760185243. URL: http://dx.doi.org/10.1162/153244302760185243.

[23] Andrew McCallum, Kamal Nigam, et al. "Employing EM and Pool-Based Active Learning for Text Classification." In: *ICML*. Vol. 98. 1998, pp. 350–358.

[24] Pannaga Shivaswamy and Thorsten Joachims. "Online Structured Prediction via Coactive Learning". In: *CoRR* abs/1205.4213 (2012).

[25] Thorsten Joachims and Filip Radlinski. "Search Engines That Learn from Implicit Feedback". In: *Computer* 40.8 (Aug. 2007), pp. 34–40. ISSN: 0018-9162. DOI: 10.1109/MC.2007.289. URL: http://dx.doi.org/10.1109/MC.2007.289.

[26] CharuC. Aggarwal and ChengXiang Zhai. "A Survey of Text Classification Algorithms". English. In: *Mining Text Data*. Ed. by Charu C. Aggarwal and ChengXiang Zhai. Springer US, 2012, pp. 163–222. ISBN: 978-1-4614-3222-7. DOI: 10.1007/978-1-4614-3223-4_6. URL: http://dx.doi.org/10.1007/978-1-4614-3223-4_6.

[27] Yiming Yang and Xin Liu. "A re-examination of text categorization methods". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 42–49.

[28] Jason D Rennie et al. "Tackling the poor assumptions of naive bayes text classifiers". In: *ICML*. Vol. 3. Washington DC). 2003, pp. 616–623.

[29] Andrew McCallum and Kamal Nigam. "A comparison of event models for Naive Bayes text classification". In: *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*. AAAI Press, 1998, pp. 41–48.

[30] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: http://dx.doi.org/10.1023/A:1022627411411.

[31] Thorsten Joachims. "Text categorization with Support Vector Machines: Learning with many relevant features". In: *Machine Learning: ECML-98*. Ed. by Claire Nédellec and Céline Rouveirol. Vol. 1398. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 137–142. ISBN: 978-3-540-64417-0. DOI: 10.1007/BFb0026683. URL: http://dx.doi.org/10.1007/BFb0026683.

[32] I. R. Goodman, Ronald P. Mahler, and Hung T. Nguyen. *Mathematics of Data Fusion*. Norwell, MA, USA: Kluwer Academic Publishers, 1997. ISBN: 0792346742.

[33] *Snowball tartaurus*. [Online; accessed 19-05-2014]. URL: http://snowball.tartarus.org/algorithms/swedish/stemmer.html.

[34] *Scikit-learn*. [Online; accessed 19-05-2014]. URL: http://scikit-learn.org/stable/.

[35] John C. Platt. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”. In: *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.