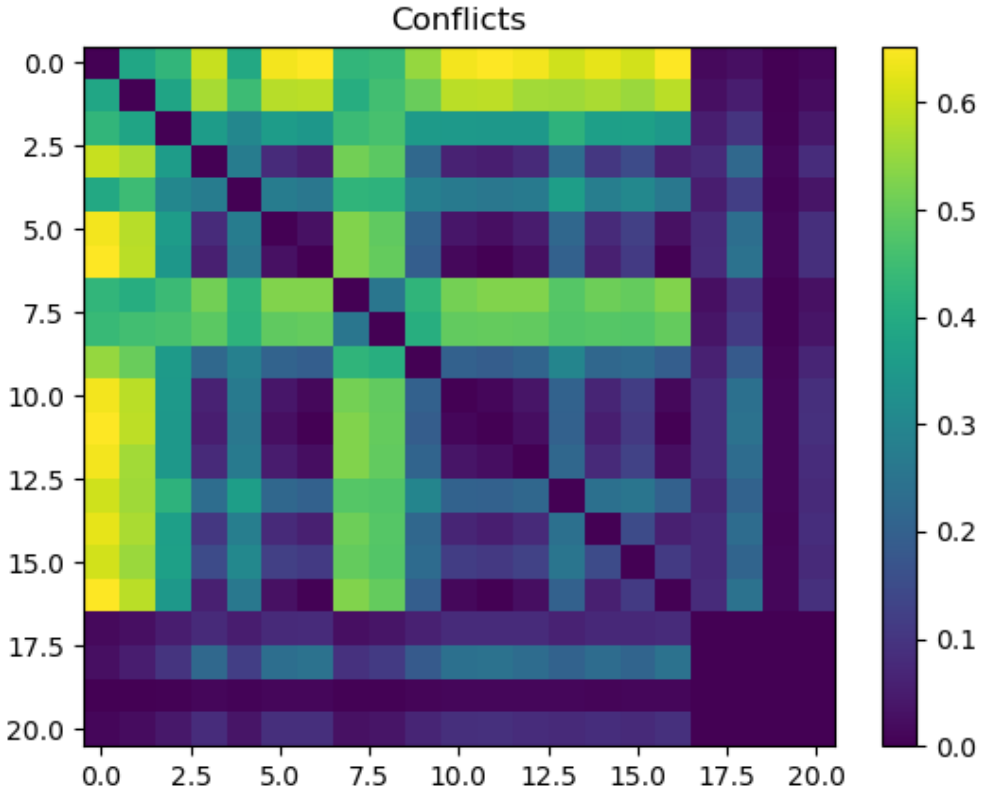
CHALMERS
UNIVERSITY OF TECHNOLOGY



Weakly Supervised Deep Learning Classification

Concept Classification from Electronic Health Records without Ground Truth Data

Master's thesis in Complex Adaptive Systems

Carl Claesson
Fredrik Johnsson

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Weakly Supervised Deep Learning Classification

Concept Classification from Electronic Health Records without Ground Truth Data

Carl Claesson
Fredrik Johnsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2021

Weakly Supervised Deep Learning Classification
Concept Classification from Electronic Health Records without Ground Truth Data
Carl Claesson
Fredrik Johnsson

Supervisor: Magnus Kjellberg, Sahlgrenska University Hospital
Examiner: Giovanni Volpe, Physics

Master's Thesis 2021
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualisation of a conflict matrix for a set of labeling functions.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Weakly Supervised Deep Learning Classification
Concept Classification from Electronic Health Records without Ground Truth Data
Carl Claesson
Fredrik Johnsson
Department of Physics
Chalmers University of Technology

Abstract

The usage of increasingly large and complex sets of data is rapidly gaining traction within healthcare and life sciences. To handle these datasets prompts for more sophisticated methods. A key such method is Artificial Intelligence, AI. There are numerous examples of successful application of AI in health care, especially in diagnostic disciplines, e.g., automatic analysis of X-ray images, treatment recommendations and monitoring adherence [25]. In some of these disciplines, AI have been demonstrated to be able to outperform humans. AI is therefore receiving more and more attention as a way to increase efficiency and safety in healthcare.
A key hindrance to the adoption of such systems is the large quantities of labeled data required to train deep learning models. One proposed method of overcoming this annotation bottleneck is *weak supervision*, or *data programming*, where the data annotation is done using *labeling functions*. These labeling functions are used to translate the expert domain knowledge of the annotator using statistical models into "denoised" or probabilistic labels that can be used to train deep learning algorithms without the use of ground truth data provided by an expert annotator.
This thesis investigates the Weak Supervision method for concept classification from electronic health records. We describe the development of a *distant supervision* method, where the external medical database MeSH is used to create labeling functions for different *phenotypes* (concepts) from the MIMIC-III database [20]. These labeling functions are then used to create probabilistic labels for a few different deep learning models to train on. A deep CNN model trained on the probabilistic labels from the labeling functions achieves a $f1$-score of 0.93 on the test set and is clearly able to generalize beyond the probabilistic labels it is trained on. It can be concluded that weak supervision seems to be a promising approach for NLP problems within the medical field that could potentially drastically decrease the need for expert annotations, which is both time-consuming and expensive.

Keywords: weak supervision, deep learning, machine learning, NLP.

Acknowledgements

We would like to thank our supervisor Magnus Kjellberg at Sahlgrenska Univerity Hospital for introducing us to the emerging field of weak supervision and dataprogramming, and guiding us throughout this project. Without the many discussions with him, this project would not have been possible.

Carl Claesson & Fredrik Johnsson, Gothenburg, June 2021

Contents

List of Figures

List of Tables

Listings

1
Introduction

Usage of AI.methods in healthcare is getting traction as ways of improving healthcare quality as well as increase efficiency [7][11]. There are numerous examples of successful application of AI in health care, especially in diagnostic disciplines, e.g., automatic analysis of X-ray images, treatment recommendations and monitoring adherence, or extracting key information from large volumes of scientific literature to create databases of genome wide association studies [25]. Physician level accuracy has been achieved in tasks such as the identification of melanoma, spinal analysis with magnetic resonance imaging, cardiovascular risk, and diabetic retinopathy.
One central piece of information is the clinical record for each patient [16]. The clinical record is composed of structured and unstructured, i.e., free-text, fields and describes all interactions with the patient, such as measurements, diagnoses, interventions. This information can be used for generating predictive machine learning models for selected and relevant patient outcomes, e.g., diagnosis, risk of mortality, risk of early readmission, or length of stay. Such models can provide a basis for generating decision support systems that could aid medical workers in their clinical work.
In order to leverage the data contained in the unstructured parts of the clinical record using algorithms, the information contained therein needs to be extracted. The task of extracting critical information from unstructured data in clinical narratives is referred to as *concept extraction*, also known as "Named Entity Recognition and Classification", and has been a prominent research problem for decades and is an enabling technology for gaining access to critical information such as prescribed drugs, various symptoms, and diagnoses. There are traditional, NLP-based tools for these tasks (also known as automated medical language processing systems) such as CLAMP, cTakes, and MetaMap [39]. These tools work by utilizing the semantics of a given narrative can be inferred by taking into account the relative co-occurrence of various words related to the concept to be extracted. For instance, the authors of [22] propose a semi-supervised learning algorithm where a sequential discriminative classifier for extracting mentions of medical problems.
Deep learning models have been shown to outperform, for example, traditional *NLP* or pure rule-based algorithms in extraction of concepts from unstructured data [29]. However, where these models are trained in the *supervised learning* setting, the large quantities of annotated data constitute a severe bottleneck in deep learning pipelines [47]. One method to overcome this bottleneck is to use crowdsourcing, where large groups of annotators can produce predictions that outscore that of a single annotator [23]. In some contexts, such as the field of medicine, this method is infeasible to, e.g., extensive knowledge required to make predictions or privacy

concerns.
Another possible approach to alleviate this problem is called *weak supervision*, or *data programming*, where the labeled data is created algorithmically by leveraging domain heuristics and estimating the accuracies of various possibly noisy sources creating supervision signals that can be used to train supervised learning algorithms. Thereby reducing costs by achieving performances that are comparable to the results achieved by years of expert hand labeling [50][43][10].

1.1 Background

This thesis is written in collaboration with Sahlgrenska University Hospital. Sahlgrenska University Hospital aims to be at the forefront in the development and adoption of AI algorithms, both for use in research and development and for clinical decision support systems. The costs and time required for creating the annotated datasets required to train supervised learning algorithms have been identified as a critical problem hampering the adoption of such algorithms. We explore the paradigm of *Weak Supervision* to overcome the annotation bottleneck. The paradigm is evaluated on concept extraction tasks from the MIMIC-III dataset [20].
Previous work has shown some promise in using weak supervision methods in a clinical context by using NLP-based rules developed with the assistance of a physician to generate weakly labeled data points. Without access to physician consultation, we explore the use of external databases and integrating these noisy signals using the paradigm of *data programming*, and its state-of-the-art implementation *Snorkel* [42]. We construct labeling functions that encapsulate various supervision signals and leverage these to construct probabilistic training labels by untangling the labeling function dependency structures using an automatic selection of generative models for statistical inference.

1.1.1 Aim

We want to know if weak supervision and data programming is a feasible approach to concept extraction tasks. Therefore, our general research question is:

Is weak supervision a suitable approach for annotating large sets of unstructured data in the medical domain?

Since the primary motivation for being able to create large annotated datasets rapidly is the huge data requirements of deep learning models, we want to know how these models perform when trained on datasets created without ground truth data. Therefore our specified research question is:

Is weak supervision a feasible approach for annotation of datasets with enough fidelity that it can be used to train deep learning classification models without ground truth labels?

Since the model used to create annotations is a classifier itself, we want to know if the labels produced by such a model have good enough quality that a deep learning model can generalize beyond the quality of the generated training labels. We aim to answer this question by implementing a weak supervision pipeline and training end models using the generated annotations. This pipeline and these models can be validated by comparing to ground truth annotations made by a human annotator.

1.2 Delimitations

The motivation for using weak supervision is to be able to generate training sets without ground truth labels. However, expert labeled data or expert domain knowledge required to create such data is required for the evaluation of these models. Therefore, we are restricted to datasets where annotations are available. However, the datasets where annotations are made by experts are generally small, so comparing the huge datasets produced by weak supervision and data programming is generally infeasible. The use of cross-validation further limits the sizes of expert-annotated labels available and, combined with the heavy class imbalance of the classification tasks considered, this further limits the range of available comparison baselines.

A large number of dataset modalities are available for the evaluation of weak supervision in general, such as text, images, and time series. Furthermore, there are numerous ways the supervision signals from these modalities can be used to train end models. We restrict ourselves to unstructured data, i.e., text, from electronic health records. The weak supervision signals are used to make classifications from these documents alone, rather than, for instance, using the signal to supervise image processing.

There are further techniques encompassed by the data programming paradigm, which aims to be a holistic approach to manage, curate, and create datasets for supervised learning. We focus on the core concept: the creation of probabilistic labels utilizing labeling functions. We do not consider, for example, data augmentation by means of learning class invariant transformation rules and its associated data programming concept: transformation functions. Neither do we consider slicing functions: a way to monitor critical slices of the training set known to produce erroneous results.

The enormous space of neural network architectures available for the evaluation of the generated labels is much too large to test and validate exhaustively. Therefore, we select one model which we know from experiments that seems to work well. Another model is selected as a baseline to validate the former. We then evaluate our training set against a third model to give indications for suggestions for further work. Nevertheless, we do not provide a comprehensive study of which architecture might be most suitable for the domain at hand. The space of hyperparameters involved in creating these models is also much too computationally large to exhaustively probe. The configurations used in this thesis are, therefore, those that have been found to work "best" from experimentation.

State-of-the-art performance is also not the aim of this project. Therefore transformers were not considered even though it would be interesting to test fine-tuned

versions of a pre-trained model such as BioBERT [26] or ClinicalBERT [19] if there was enough time.

2

Theory

In this chapter, the theory needed to understand the central concepts of the project is presented. Crucially, we describe the theory of *Weak Supervision* in section 2.1. In this section, we describe how labels are generated by creating a generative model (described in section 2.1.2) that is automatically selected, working on the outputs of Labeling functions (described in section 2.1.1). Understanding the denoising process, described in section 2.1.5, requires some understanding of probabilistic graphical models. Therefore, we give a brief introduction to graph theory and probabilistic graphical models in section 2.1.3.

The class of machine learning models that are trained in this paradigm are that of deep learning or neural network models. Therefore, we give a brief introduction to the subject of neural networks in section 2.3. The scoring metrics used to evaluate our models are described in section 2.2.

The order the information is presented is in order to its relevance to the project. The reader is recommended to start at Section 2.3 if they do not have a solid background in deep learning and neural networks.

2.1 Weak Supervision

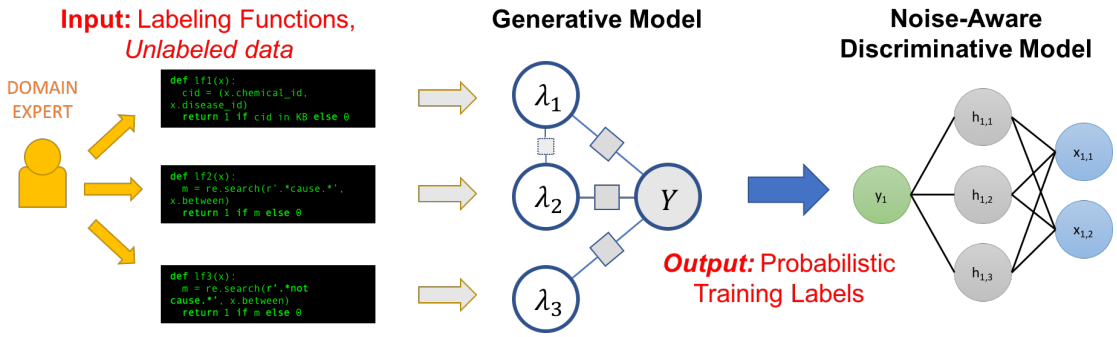


Figure 2.1: Illustration of the weak supervision workflow [41]. On the leftmost side, a domain expert has some intuitions about how a given dataset should be annotated. These intuitions are used to write *labeling functions*. The votes of these labeling functions are then denoised using a generative graphical model (here seen as a factor graph with the label Y as latent). The outputs of this generative model, i.e. the denoised labels can then be used to train a Noise-Aware discriminative model.

The main idea behind *Weak Supervision* is that in the absence of strong supervision sources (i.e., labels created by an expert annotator), weaker (or noisier) sources of information can be used to train ML algorithms. This includes techniques such as distant supervision [34] and co-training [3]
A high-level description of this workflow is seen in figure 2.1. The labeling functions are written based on domain knowledge or other sources of information about that annotation process. The labeling functions then are applied to the unlabeled data to produce noisy labels which may or may not have complex interdependencies. We then model the label accuracies and their interdependencies in order to learn the appropriate weightings of these labels and thereby denoising them. These denoised labels hereby referred to as *silver labels*, can then be used as training data for a discriminative end model. We call the labels produced by an expert annotator *gold labels*.
In the standard supervised learning context, here called fully supervised learning, complete class information, in the form of labeled examples, are used to train a model that in the prediction stage can infer the class of new *unlabeled* examples. Where complete class information is unavailable, several techniques have been proposed to train models without complete class information. A taxonomy is proposed by [17] to classify such nonstandard supervision problems.

- *Full-supervision* - For each example, complete class information is provided.
- *Unsupervision* - No class information is provided with examples.
- *Semi-supervision* - Part of the examples are provided fully supervised. The rest are unsupervised.
- *Positive-unlabeled* - Part of the examples are provided fully supervised, all of them with the same categorization. The rest are unsupervised.
- *Candidate labels* - For each example, a set of class labels is provided. In this set, the class labels(s) that compose the real categorization of the example are included.
- *Probabilistic labels* - For each example, the probability of belonging to each class label is provided. This probability distribution is expected to assign a high probability to the real label(s).
- *Incomplete* - For each example, a subset of the labels that compose its real categorization is provided).
- *Noisy labels* - For each example, complete class information is provided, although its correctness is not guaranteed.
- *Crowd* - For each example, many different non-expert annotators provide their (noisy) categorization.
- *Mutual label constraints* - For each group of examples, an explicit relationship between their class labels is provided (e.g., all the examples have the same categorization).
- *Candidate labeling vectors* - For each group of examples, a set of labeling vectors (including the real one) is provided. A labeling vector provides a class label for each example of a group.
- *Label proportions* - For each group of examples, the proportion of examples belonging to each class label is provided.

The framework for weak supervision that this thesis explores is that of *Snorkel* and

Data programming. In this framework, noisy labels are provided in the form of outputs of *Labeling Functions*. No part of the examples is provided as supervised, which means this scheme does not fall under that of *Semi-supervision*, but instead that of *Unsupervision*. The labels that are generated are probabilistic, meaning that labels cannot be written as belonging to the set $\{0, 1\}$ but instead takes on continuous values between 0 and 1, denoting the probability that a given data point belongs to a certain class. Models trained on this additional information are said to be noise-aware.

2.1.1 Labeling Functions

The main entry point, and first class citizen of weak supervision is the *Labeling Function*. The purpose of labeling functions is to model expert domain knowledge (seen in the leftmost part of Figure 2.1. The expert annotator will then usually be able to point at some feature of the data, and based on that feature, write a labeling function that captures this insight.
To do this, we write *labeling functions* $\lambda : X \to Y \cup \{\emptyset\}$. The empty set is to allow for the Labeling Functions to abstain, i.e. neither predict 1 or -1. Suppose we have n (unlabeled) data points $x_i \in X$ and m label functions. From this we compute a matrix of labeling function outputs:

$$\begin{pmatrix} \lambda_1(x_1) & \lambda_1(x_2) & \dots & \lambda_1(x_n) \\ \lambda_2(x_1) & \ddots & & \\ \vdots & & \ddots & \\ \lambda_m(x_1) & & & \lambda_m(x_n) \end{pmatrix} = \Lambda \in (Y \cup \{\emptyset\})^{m \times n} \tag{2.1}$$

If we restrict our attention to binary classification tasks then
$\{Y \cup \{\emptyset\}\}^{m \times n} = \{-1, 0, 1\}^{m \times n}$, which can be seen as positive, negative or abstaining. Which element is regarded as positive, negative or abstaining is a matter of implementation. Now we aim to estimate the probability of Y given X, without any reference to ground truth. We do, however, have access to the matrix of labeling function outputs. Intuitively, the accuracies of the labels can be computed using a Majority Vote scheme. Suppose, for instance, that for a given point X, 9 out of 10 labeling functions vote positive. Then we can estimate that the probability of Y being positive given X is 90%. Since we usually have conflicts between the labeling function outputs, and some of them are more predictive than others, this approach may prove too coarse to produce labels that are sufficiently accurate to be used as training data for an end model. Instead, an underlying graphical model is assumed to model these dependencies of labeling functions. This method of "learning" the underlying structure of a graphical model without prior assumptions about how the variables are interrelated is called *structure learning*.
While we cannot observe the "true" labels directly, we do observe the entire set of datapoints $x_i \in X$. By feeding $x_i \in X$ into a pre-trained classifier we can write labeling functions based on higher-order attributes extracted from this pre-trained classifier. In the context of the unstructured data found in electronic health records, this comes down to using NLP, or *Natural Language Processing* -methods, e.g., part-of-speech tagging, text categorization, named entity recognition and sentiment

analysis. For instance, the sentiment of a sentence may be used as a basis for the decision of a labeling function, and extracted entities may be matched with knowledge of these entities using an external database, with linking done with for example UMLS, or Unified Medical Language System. In listing 2.1 we provide examples of the usage of such higher-order attributes in the context of electronic health records.

Listing 2.1: Example of labeling functions

```
def LF_is_negated(x):
    return NEGATIVE if 'negated' in x.pain.props else ABSTAIN

def LF_is_hypothetical(x):
    return NEGATIVE if 'hypothetical' in x.pain.props else ABSTAIN

def LF_is_historical(x):
    if 'hist' not in x.pain.props:
        return ABSTAIN
    return NEGATIVE if x.pain.props['hist'] == 1 else POSITIVE
```

Labeling functions are not the only approach that can be used to manipulate data in the Data Programming paradigm and Snorkel. The vision for Snorkel is to be seen as a comprehensive system for the management of training data. For this reason, there is also functionality for defining class invariant (i.e., label preserving) transformations. These are called *transformation functions*[44]. In the context of NLP, such transformations are, e.g., synonymy relations. In the context of image processing, an example of a class invariant transformation would be a combination of rotations, translations, inversions, and scaling. By defining a set of such transformations, Snorkel is able learn how to compose chains of such transformations, and by doing so extending the dataset by means of *data augmentation*.

There may be subsets of the training data where the predictions are more critical[4]. For instance, some diseases may be more harmful and may thus warrant more attention. Optimizing the end model with respect to global metrics such as f_1-score may then be too coarse. The Data Programming paradigm provides abstractions for specifying such critical slices of the data, thus allowing for more attention to be given to subsets of the data where the outcomes are deemed more important and allows for monitoring the training results with respect to the end models predictions for these slices. The main entry point for this technique is *slicing functions*, and they are written similarly to how labeling functions are written.

2.1.2 Structure Learning and Generative Models

Given the outputs of the labeling functions, we adopt a graphical model in order to calculate our label probabilities. However, the structure of this model has to be specified. Manually specifying this graphical model is arduous, time-consuming, and may require domain knowledge not readily available at the time of writing labeling functions. Model selection instead is made algorithmically. Automating the learning of the structure of a Bayesian network is called structure learning[8].

With the labeling functions as inputs we want to synthesize noisy labels $\bar{y}_i \in \bar{Y}$ where $i = 1, ...m$ [49]. This is done by using the label matrix as input to estimate latent variables in a probabilistic graphical model. For this reason this is called a generative model[38]
. This (conditionally independent) probabilistic model takes the form:

$$p_\theta(\Lambda, Y) \propto \exp\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\theta_j^{Acc}\phi_j^{Acc}(\Lambda_i, y_i)\right) \tag{2.2}$$

[2] where ϕ^{Acc} are accuracy dependencies:

$$\phi_j^{Acc}(\Lambda, y_i) := y_i\Lambda_{ij} \tag{2.3}$$

so that the outputs of the labeling functions are conditionally independent given the true label. and θ_j are the accuracy of each labeling function. Now we can estimate parameters θ by minimizing the negative log marginal likelihood $p_\theta(\bar{\Lambda})$

$$\arg\min_\theta -\log\sum_Y p_\theta(\bar{\Lambda}, Y) \tag{2.4}$$

which we can solve using stochastic gradient descent.
The conditional independence assumption in the model outlined above may not be practical, since correlations naturally arise between labeling functions. This can be amended by generalizing the above model. The conditionally independent model is then generalized as a factor graph with additional dependencies.

$$p_\theta(\Lambda, Y) \propto \exp\left(\sum_{i=1}^{m}\sum_{t\in T}\sum_{s\in S_t}\theta_s^t\phi_s^t(\Lambda_i, y_i)\right) \tag{2.5}$$

S_t is a set of index tuples and T is a set of dependency types. For instance the correlation dependency is defined as

$$\phi_{jk}^{Corr}(\Lambda_i, y_i) := \mathbb{I}\{\Lambda_{ij} = \Lambda_{ik}\} \tag{2.6}$$

where $\mathbb{I}$ is the indicator function. We can also define conjunction dependencies: $\phi_{jk}^{And}(\Lambda_i, y_i) := \mathbb{I}\{\Lambda_{ij} = y_i \wedge \Lambda_{ik} = y_i\}$.
Sparsity is induced by introducing a regularization parameter ϵ and an L_1-regularization term into the objective function. The learning objective is then:

$$\arg\min_\theta -\log p_\theta(\bar{\Lambda_j}, \bar{\Lambda}_{\setminus j}) + ||\epsilon||_1 = \tag{2.7}$$

$$\arg\min_\theta \sum_{i=1}^{m}\log\sum_{y_i}\log p_\theta(\bar{\Lambda_j}, y_i, \bar{\Lambda}_{\setminus j}) + ||\epsilon||_1 \tag{2.8}$$

The parameters can be computed using gradient descent. The gradient of the learning objective can be computed in closed form as:

$$-\frac{\partial\log p(\hat{\Lambda}_j|\hat{\Lambda}_{\setminus j})}{\partial\theta_s^t} = \alpha - \beta \tag{2.9}$$

where

$$\alpha = \sum_{i=1}^{m} \sum_{\Lambda_{ij}, y_i} p_\theta(\Lambda_{ij}, y_i | \hat{\Lambda}_{i\setminus j}) \phi_s^t((\Lambda_{ij}, \hat{\Lambda}_{i\setminus j}), y_i)$$

$$\beta = \sum_{i=1}^{m} \sum_{y_i} p(y_i | \hat{\Lambda}_i) \phi_s^t(\hat{\Lambda}_i, y_i).$$

This suggests an algorithm for structure learning, using Gibbs sampling[13]: Gibbs sampling is a Markov Chain Monte Carlo Technique (or a special case of Metropolis-Hastings sampling) used to carry out Bayesian inference overcoming the problem of having to integrate high dimensional functions and the name Markov Chain Monte Carlo is derived from the usage of the previous sample values to generate the next samples, thus creating a Markov Chain.

Algorithm 1: Structure Learning for Data Programming

input : Observations $\bar{\Lambda} \in \{-1, 0, 1\}^{m \times n}$, threshold ϵ, distribution p with parameters θ, initial parameters θ^0, step size η, epoch count T, truncation frequency K

output:

$D \leftarrow \emptyset$;
for *j = 1 to n* **do**
  $\theta \leftarrow \theta^0$;
  for $\tau = 1$ *to* $\mathcal{T}$ **do**
    for *i = 1 to m* **do**
      for θ_s^t *in* θ **do**
        $\alpha \leftarrow \sum_{\Lambda_{ij}, y_i} p(\Lambda_{ij}, y_i | \bar{\Lambda}_{i\setminus j}) \phi_s^t((\Lambda_{ij}, \bar{\Lambda}_{i\setminus j}), y_i)$;
        $\beta \leftarrow \sum_{y_i} p(y_i | \bar{\Lambda}_i) \phi_s^t(\bar{\Lambda}, y_i)$;
        $\theta_s^t \leftarrow \theta_s^t - \eta(\alpha - \beta)$;
      end
      if $\tau m + i \mod K$ *is* 0 **then**
        for $\theta_s^t \in \theta$ *where* $\theta_s^t > 0$ **do**
          $\theta_s^t \leftarrow \max\{0, \theta_s^t - K\eta\epsilon\}$
        end
        for $\theta_s^t \in \theta$ *where* $\theta_s^t < 0$ **do**
          $\theta_s^t \leftarrow \min\{0, \theta_s^t + K\eta\epsilon\}$
        end
      end
    end
  end
  for θ_s^t *in* θ *where* $j \in s$ **do**
    if $|\theta_s^t| > \epsilon$ **then**
      $D \leftarrow D \cup \{(s, t)\}$;
    end
  end
end
return D

This approach for structure learning involves selecting an appropriate selection

threshold ϵ, where high epsilon will lead to more dependencies added to the model. This, in turn, leads to a higher model complexity, which can potentially lead to computationally intractable models, whereas a low selection threshold will miss potentially important connections between supervision sources. Therefore a parameter sweep is done over different selection thresholds, and then the threshold is chosen at the elbow point for the number of edges.
In some cases, the majority vote scheme works MV fine, in which case this model is selected; otherwise, the generative model GM is selected. The algorithm for finding the selecting the appropriate model is outlined in Algorithm 2.

Algorithm 2: Modeling Strategy Optimizer

input : Label matrix $\Lambda \in (Y \cup \{\emptyset\})^{m \times m}$, advantage tolerance, γ, structure search resolution η
output: Modeling strategy
if $\tilde{A}^*(\Lambda) < \gamma$ **then**
  return MV;
end
$Structures \leftarrow []$;
for *i from 1 to* $\frac{1}{2\eta}$ **do**
  $\epsilon \leftarrow i * n$;
  $E \leftarrow LearnStructure(\Lambda, \epsilon)$;
  Structures.append(|E|, ϵ) ;
end
$\epsilon \leftarrow$ SelectElbowPoint(Structures) ;
return GM_ϵ

So, in summary: the structure learning algorithm uses a combination of stochastic gradient descent interleaved with Gibbs sampling, with a threshold ϵ representing a trade-off between computational costs and label model performance. A parameter sweep is done over a set of different ϵ, and the "optimal" trade-off is chosen as the elbow point of the number of correlations as a function of the correlation threshold. Should this generative model perform better than the majority vote model MV, then GM is returned, otherwise MV. Having selected a model, we can now perform the inference task of calculating our probabilistic labels. This can be done using, e.g., Gibbs sampling [43]. To ensure efficiency when running on very large datasets, a matrix completion style algorithm is used, which only requires running SGD M^2 times, where M is the number of cliques of the dependent labeling functions, creating a much more scalable algorithm. To describe this approach, we need to introduce a few key graph-theoretic notions.

2.1.3 Probabilistic Graphical Models and Graph Theory

In the context of weakly supervised machine learning, the number of data points and labeling functions may be very large. Therefore, achieving favorable complexity bounds is critical for making the data programming approach feasible. One approach to achieve better complexity bounds than the algorithms outlined above can be derived by exploiting some mild assumptions and key insights derived therefrom. This *matrix completion* algorithm requires, however, necessitates a few more graph

theoretic concepts and the notion of probabilistic graphical models.
Probabilistic graphical models or structured probabilistic models are a way of describing probability distributions by using the language of graph theory. These probabilistic graphical models are mainly divided into directed and undirected models. Directed models are, as the name implies, models where the edges of the underlying graph are directed, and conversely, undirected models are models where the edges are undirected. Directed models are mainly used where causality relations are evident (as in X causes Y - modeled as a directed edge between X and Y, and X and Y are random variables, seen as nodes in the graph). Where such knowledge is unavailable, or where the direction of interactions is otherwise hard to ascertain, undirected models, otherwise known as Markov Random Fields, are a popular alternative. We focus here on undirected models since this is the language in which the matrix-completion algorithm is phrased in section 2.1.5.
On a more technical level, a Markov Random Field is an undirected graph. For each clique C in the graph, a factor $\phi(C)$, constrained as strictly nonnegative, is defined to measure the random variables' affinity for the joint state defined by the factor. Observed variables are usually denoted by shaded nodes, and unobserved (or latent) variables are unshaded.
The expression for the (unnormalized) probability distribution is then divided by a *partition function* Z, a concept borrowed from statistical physics ensuring that the distribution integrates to unity.
The edges in the graph denote direct "interaction" between the random variables, and the absence of which denotes "non-interaction". Two nodes may interact indirectly if there is a path between the nodes where all nodes connecting the two are unshaded.
Suppose there is ambiguity as to whether each clique has a factor whose scope encompasses the entire clique. This ambiguity can be resolved by drawing the MRF as a factor graph, where the factors are included as square nodes, creating a bipartite graph where the scope of each factor is clearly denoted by its edges to its constituent variables.
We summarise the above by providing the formal definitions from [24] below.
Definition 2.1.1 (graph)**.** A graph G = G(V,E) a data structure consisting of a set of nodes V and a set of edges, E. A pair of nodes X_i, X_j can be connected by a directed edge $X_i \to X_j$ or an undirected edge $X_i - X_j$.
Definition 2.1.2 (path)**.** We say that $X_i, ...X_k$ form a path in the graph G(V,E) if for every i=1,...k-1 we have that either $X_i \to X_j$ or $X_i - X_j$. A path is directed if, for at least one i, we have $X_i \to X_j$.
Definition 2.1.3 (trail)**.** We say that $X_i, ...X_k$ form a trail in the graph G(V,E) if, for every i=1,...,k-1 we have that $X_i \rightleftharpoons X_j$
Definition 2.1.4 (loop)**.** A loop in G is a trail $X_1, ...X_k$ where $X_1 = X_k$. A graph is singly connected if it contains no loops. A node in a singly connected graph is called a leaf if it has exactly one adjacent node.
Definition 2.1.5 (chordal graph)**.** Let $X_1 - X_2 - ...X_k - X_1$ be a loop in the graph; a chord in the loop is an edge connecting X_i and X_j for two nonconsecutive nodes X_i, X_j. An undirected graph H is said to be chordal if any loop $X_1 - X_2 - ...X_k - X_1$ has a chord. Chordal graphs are also called triangulated.
Definition 2.1.6 (induced subgraph)**.** let $K = (X, E)$ and let $\boldsymbol{X} \supset X$ We define the

induced subgraph $K[X]$ to be the graph (X, E') where E' are all edges $X \rightleftharpoons Y \in E$ such that $X, Y \in \boldsymbol{X}$.

Definition 2.1.7 (clique)**.** A subgraph over X is complete if every two nodes in $\boldsymbol{X}$ are connected by some edge. The set X is often called a clique; we say that a clique $\boldsymbol{X}$ is maximal if for any superset of nodes $\boldsymbol{Y} \supset \boldsymbol{X}$, $\boldsymbol{Y}$ is not a clique.

Definition 2.1.8 (sepset)**.** Let H be a connected undirected graph, and let $C_1, ...C_k$ be the set of maximal cliques in H Let T be any tree structured graph whose nodes correspond to the maximal cliques $C_1, ...C_k$. Let C_i,C_j be two cliques in the tree that are directly connected by an edge; we define $S_{i,j} = C_i \cap C_j$ to be a sepset between C_i and C_j. Let $W_{<(i,j)}$ be all of the variables that appear in any clique on the C_i side of the edge. Thus each edge decomposes X into three disjoint sets: $W_{<(i,j)} - S_{(i,j)}$, $W_{<(j,i)} - S_{i,j}$ and $S_{i,j}$

With these definitions, we are now ready to define a junction tree:

Definition 2.1.9 (junction tree)**.** We say that a tree T is a clique tree (or junction tree) for H if:

- each node corresponds to a clique in H, and each maximal clique in H is a node in T;
- each sepset $S_{i,j}$ separates $W_{<(i,j)}$ and $W_{<(j,i)}$ in H.

.

Definition 2.1.10 (Markov Network)**.** A *Markov network* is a tuple $M = (G, \{\psi_{C_1}, ..., \psi_{C_L}\})$ where $G = (X, E)$ is an undirected graph with maximal cliques $C_1, ..., C_l$. The nodes X correspond to random variables and $\psi_{C_i} : val(C_i) \rightarrow \mathbb{R}_+$ are nonnegative functions, called *factors* or *potentials* over the maximal cliques of the graph G. The Markov Network defines a probability distribution according to

$$P_M(X_1, ..., X_N) = \frac{1}{Z} \prod_{l=1}^{L} \psi_{C_l}(C_l) \tag{2.10}$$

where Z is a normalization constant given as

$$Z = \sum_{x \in val(X)} \prod_{l=1}^{L} \psi_{C_l}(x(C_l)). \tag{2.11}$$

Often, Z is also referred to as *partition function.*

For an illustration of a Markov Network see figure 2.2.

2.1.4 Inference

The standard method for inference on undirected graphs is the Sum-Product algorithm. For exact inference, the sum-product algorithm is usually run on transformed trees where each node corresponds to a clique. So for exact inference, the MN is transformed into a junction tree or a clique tree by means of the following procedure:

- if the graph is directed, the directed graph is transformed into an undirected graph by means of *moralization.*
- Triangulation: The MN is transformed into a junction tree, also known as a chordal graph, by adding edges in such a way as to create minimal max-cliques.

- With the newly constructed chordal graph, A junction tree can be constructed. The cliques C_i of the junction tree are separated by separator sets $S = C_i \cap C_j$. By definition, this tree has to satisfy the *running intersection property*: any random variable present in two cliques, C_i,, C_j must also be in every clique connecting C_i, C_j.

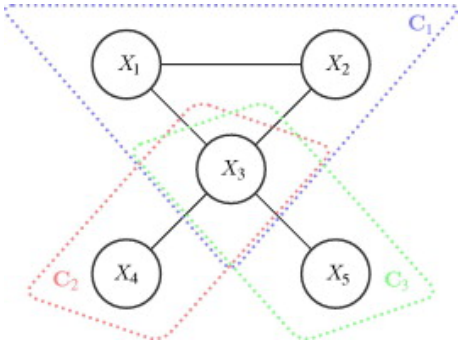


Figure 2.2: Illustration of a Markov Network [40]. The maximal cliques of this graphs (surronded by boxes) are $C_1 = \{X_1, X_2, X_3\}$, $C_2 = \{X_3, X_4\}$ and $C_3 = X_3, X_5$. The joint probability distribution of this model is given as $P_M = (X_1, ...X_5) = \frac{1}{Z}\psi_{C_1}(X_1, X_2, X_3)\psi_{C_2}(X_3, X_4)\psi_{C_3}(X_3, X_5)$.

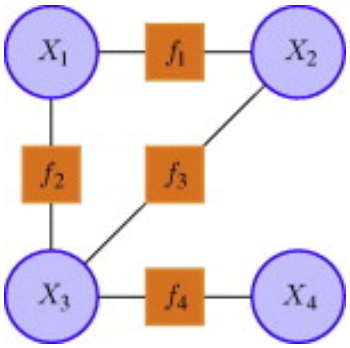


Figure 2.3: Illustration of a Factor Graph [40]. The graph illustrated is $F = (G, f_1, ...f_4)$. The nodes G are the union of the set of variable nodes $\{X_1, ..., X_4\}$ and the set of factor nodes $\{F_1, ...F_4\}$. The functions corresponding to these factor nodes are $f_1(X_1, X_2)$, $f_2(X_1, X_3)$, $f_3(X_2, X_3)$ and $f_4(X_3, X_4$. The represented joint probability density is $P_F(X_1, ..., X_4) = \frac{1}{Z} f_1(X_1, X_2) f_2(X_1, X_3) f_3(X_2, X_3) f_4(X_3, X_4)$ where Z is the normalization constant.

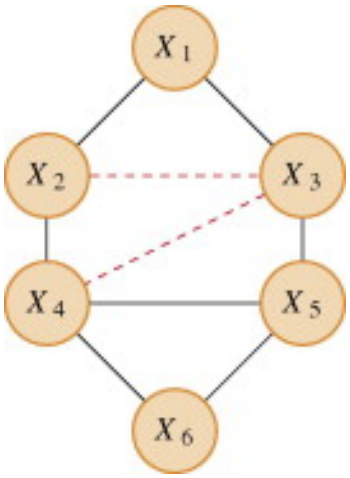


Figure 2.4: Illustration of a graph triangulation [40]. Graphs are transformed into triangulated, or chordal, by removing chords by adding edges. Optimal triangulation, meaning that the maximal clique is minimal is NP-hard.

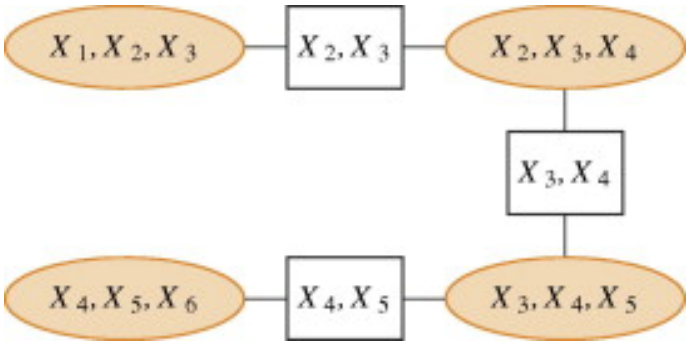


Figure 2.5: The junction tree created from the triangulated graph in 2.4 [40]. The separators are the square nodes: $\{X_2, X_3\}$, $\{X_3, X_4\}$ and $\{X_4, X_5\}$. The maximal cliques in the oval nodes are: $\{X_1, X_,2, X_3\}$, $\{X_2, X_3, X_4\}$, $\{X_4, X_5, X_6\}$ and $\{X_3, X_4, X_5\}$.

Definition 2.1.11 (Factor Graph)**.** A *factor graph* is a tuple $F = (G, \{f_1, ...f_L\})$,

- $G = (\boldsymbol{V}, \boldsymbol{E})$ is an undirected bipartite graph such that $\boldsymbol{V} = \boldsymbol{X} \cup \boldsymbol{F}$, where $\boldsymbol{X}$ and $\boldsymbol{F}$ are disjoint, and the nodes $\boldsymbol{X} = \{F_1, ..., F_L\}$ correspond to random variables. The nodes $\boldsymbol{X}$ are *variable* nodes, while the nodes $\boldsymbol{F}$ are *factor* nodes.
- Further, $f_1, ..., f_L$ are positive functions and the number of functions L equals the number of nodes in $\boldsymbol{F} = \{F_1, ..., F_L\}$. Each node F_i is associated with the corresponding function f_i, and each f_i is a function of the neighboring variables of F_i, i.e. $f_i = f_i(Nb_G(F_i))$. Here $Nb_G(F_i)$ are the neighbours of F_i in the graph G.

The factor graph F defines the joint probability distribution

$$P_F(X_1, ..., X_N) = \frac{1}{Z} \prod_{l=1}^{L} f_l(Nb_G(F_l)) \tag{2.12}$$

where Z is the normalization constant given as

$$Z = \sum_{x \in val(X)} \prod_{l=1}^{L} f_l(Nb_G(F_l)). \tag{2.13}$$

As each factor node corresponds to a factor the terms are used equivalently.

2.1.5 Denoising Procedure by Matrix-Completion

The previous Gibbs-sampling based approach for learning the parameters of the labeling model outlined above may not be fast enough for the large models typically used in applications. The version of Snorkel used in this thesis, therefore, uses a method of learning the parameters that leads to over 100 times faster runtimes[45]. To present this method, we introduce some notation: we call the datapoints $x \in X$ and the true labels $y \in Y$. For ease of presentation we consider here the binary setting, i.e. $Y = \{1, -1\}$. As before, we have labeling functions, that when applied to x, returns either POSITIVE, NEGATIVE, or ABSTAIN, phrased as $\lambda_j \in Y \cup \{\emptyset\}$, where $\emptyset$ denotes the ABSTAIN value as before. A graph of the *conditional dependency structure* is modeled as Markov random field with the associated graph $G_\lambda = (V, E)$ where $V = \{y, \lambda_1, \lambda_2, ..., \lambda_m\}$, meaning that when two labeling sources λ_i, λ_j are not independent conditioned on y, then there will be no edge between them. See definition 2.1.10
We introduce the sufficient statistics [12],

$$\psi(C, y_C) = \mathbb{1}\{\cap_{j \in C} V_i = (y_C)_i\} \tag{2.14}$$

as an indicator variable that the event of a clique $C \in C$ of labeling functions taking on a set of values y_C. Defining the vector $\psi(C) \in \{0, 1\}^{Y^C}$ then the parameters we which to learn are given by $\theta = E[\psi(C)]$.
One more assumption is made: G_λ is triangulated (for an illustration, see figure 2.4), admitting a junction tree representation (see figure 2.5, and definition 2.1.9) with singleton separator sets whose sets of cliques we denote $\hat{C}$. This corresponds to an assumption that some sets are highly correlated since they originate from the same sources (e.g., the same databases, heuristics, etc.). $\hat{C}$ can now be decomposed into separator sets of the junction tree $S = \{y\}$, i.e., nodes separating the maximal cliques, and only containing y, and the complementary set of observable cliques $O \subseteq \hat{C}$. We therefore have definitions: $O = \{C | y \notin C, C \in \hat{C}\}$, $S = \{y\}$
We can now write the generalized covariance matrix of the graphical model as:

$$Cov[\psi(O \cup S)] = \Sigma = \begin{bmatrix} \Sigma_O & \Sigma_{OS} \\ \Sigma_{OS}^T & \Sigma_S \end{bmatrix} \tag{2.15}$$

The right hand side terms are thusly given as an observable block $\Sigma_O = Cov[\psi(O)]$. $\Sigma_{OS} = Cov[\psi(O), \psi(S)]$ is an unobserved block and a function of the parameters we want to recover. Lastly we have $\Sigma_S = Cov[\psi(S)] = Cov[\psi(Y)]$ is a function of the class balance $P(Y)$.
K_O is the observable block of Σ. Assuming that the dependency graph is sparse implies that the inverse covariance matrix is graph-structured, meaning that when

there are no edges between two edges λ_i and λ_j, the corresponding term in the inverse covariance matrix will be 0[28]. Another way to say this is that every λ_i and λ_j are independently conditioned on all the other terms.
Using the block matrix inversion lemma [5] we compute the inverse of (2.15):

$$K = \Sigma^{-1} = \begin{bmatrix} K_O & K_{OS} \\ K_{OS}^T & K_S \end{bmatrix} = \begin{bmatrix} \Sigma_O^{-1} + \Sigma_O^{-1}\Sigma_{OS}c\Sigma_{OS}^T\Sigma^T\Sigma_O^{-1} & -c\Sigma_{OS}\Sigma_S \\ c\Sigma_{OS}^T\Sigma_O^{-1} & c \end{bmatrix} \tag{2.16}$$

where $c = (\Sigma_S - \Sigma_{OS}^T\Sigma_O^{-1}\Sigma_{OS})^{-1}$.
We therefore have

$$K_O = \Sigma_O^{-1} + c\Sigma_O^{-1}\Sigma_{OS}\Sigma_{OS}^T\Sigma_O^{-1} \tag{2.17}$$

By letting $z = \sqrt{c}\Sigma_O^{-1}\Sigma_{OS}$ we can rewrite equation (2.17) as

$$K_O = \Sigma_O^{-1} + zz^T \tag{2.18}$$

We define Ω as the set of indices (i, j) for which $(K_O)_{i,j} = 0$, as determined by G_λ. For any pair $(i, j) \in \Omega$ such that $i \neq j$, that is where no edge is present, the left hand side will be 0 [28].

$$0 = (\Sigma^{-1})_{i,j} + z_i z_j \text{ for } (i, j) \in \Omega \tag{2.19}$$

We can thus estimate z by rephrasing 2.19 as a matrix-completion problem:

$$\hat{z} = \arg\min_z ||\hat{\Sigma}_O^{-1} + zz^T||_\Omega \tag{2.20}$$

where $||.||_\Omega$ is the Frobenius norm. Other parameters can be recovered by straightforward computation:

$$\begin{aligned} c = (\Sigma_S - \Sigma_{OS}^T\Sigma_O^{-1}\Sigma_{OS})^{-1} &= (\Sigma_S - (c^{-1/2}\Sigma_O z)^T\Sigma_O^{-1}(c^{-1/2}\Sigma_O z))^{-1} \\ &= (\Sigma_S - c^{-1}z^T\Sigma_O z)^{-1} \implies c^{-1} = \Sigma_S - c^{-1}z^T\Sigma_O z \\ \implies c^{-1}(1 + z^T\Sigma_O z)) = \Sigma_S &\implies c = \Sigma_S^{-1}(1 + z^T\Sigma_O) \end{aligned}$$

Using the identity $Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$ we get:

$$\Sigma_{OS} + \mathbb{E}[\psi(O)]\,\mathbb{E}[\psi(S)]^T = \mathbb{E}[\psi(O)\psi(S)^T] \tag{2.21}$$

.
The class balance $P(Y)$ can be recovered using the subset of the sources that are conditionally independent, $\lambda_i \perp\!\!\!\perp \lambda_j | Y$. Denote the observerd overlaps matrix between labeling functions i and j as $A_{i,j} = \mathbb{E}[\psi_i\psi_j^T]$, we have :

$$\begin{aligned} A_{i,j} &= \mathbb{E}[(\psi_i)_k(\psi_j)_l] \\ &= P(\lambda_j = y_k, \lambda_j = y_l) \\ &= \sum_{y \in Y} P(\lambda_i = y_k, \lambda_j = y_l)P(y = y) \\ &= \sum_{y \in Y} P(\lambda_i = y_k | y = y)P(\lambda_j = y_l | y = y)P(y = y). \end{aligned}$$

By defining $(B_i)_{j,k} = P(\lambda_i = y_j | y = y_k)$ and the diagonal matrix $P_{i,i} = P(y = y_i)$ A can then be written as $A_{ij} B_i P B_j^T = \tilde{B}_i \tilde{B}_j$ where $\tilde{B}_i = B_i \sqrt{P}$ (since P is diagonal and positive semi-definite).
Using the law of total probability:

$$\sqrt{P(y)} \sum_{y \in Y \cup \emptyset} P(\lambda_i = y | y) = \sqrt{P(y)} \tag{2.22}$$

we can then estimate the class balance using that:

$$P = \text{diag}(\tilde{B}_i^T \vec{I})^2. \tag{2.23}$$

by first recovering $\tilde{B}_i$ and then from that recovering P. Putting all this together suggests an algorithm for estimating the source accuracies:

Algorithm 3: Source Accuracy Estimation for Multi-Task Weak Supervision

input : Observed labels $\mathbb{E}[\psi(O)]$ and covariance $\hat{\Sigma}_O$; correlation sparsity structure Ω

$\mathbb{E}[\psi(y)] \leftarrow \text{ClassBalance}(\mathbb{E}(\psi(O)], \hat{\Sigma}_O, \Omega)$;
$\hat{z} \leftarrow \arg\min_z ||\hat{\Sigma}_O^{-1} + zz^T||_\Omega$;
$\hat{c} \leftarrow \Sigma_S^{-1}(1 + \hat{z}^T \hat{\Sigma}_{OS})$;
$\hat{\Sigma}_{OS} \leftarrow \hat{\Sigma}_O \hat{z} / \sqrt{\hat{c}}$;
$\hat{\theta} \leftarrow \hat{\Sigma}_{OS} + \mathbb{E}[\psi(Y)]\,\mathbb{E}[\psi(O)]$;
return $\hat{\theta}$;

With these recovered parameters we can now estimate our probabilistic labels $\bar{Y} = P_\theta(y|\lambda)$ using the junction tree defined over G_λ:

$$p_\theta(y, \lambda) = \frac{\prod_{C \in \bar{C}} P(V_C)}{\prod_{S \in \mathcal{S}} P(V_S)} = \frac{\prod_{C \in \bar{C}} \theta_{(C,(y,\lambda_C))}}{\prod_{S \in \mathcal{S}} \theta_{(C,(y,\lambda_S))}} \tag{2.24}$$

with $V_C = \{V_i\}_{i \in C}$, $V_0 = y$ and $V_{i>0} = \lambda_i$. So, in summary: instead of learning the correlations between the labeling function output by means of stochastic gradient descent, the sought parameters are recovered by exploiting the block graph structure of the underlying covariance matrix of the graphical model; The graph is split into an observable part and an unobservable part. The parameters are then recovered by exploiting knowledge of the sparsity structure of this covariance matrix.

2.2 Scoring Metrics

For balanced datasets, it is common to simply use accuracy as a scoring metric to evaluate the performance of a classifier. However, for imbalanced datasets using accuracy can be misleading. For example, if the class balance is 9 to 1, an accuracy of 90 percent can be achieved by simply only predicting the most common class. Considering the *precision* and *recall*, defined in Table 2.1, is more suitable in the case of heavily imbalanced datasets.

	Definition
TP (True Positives)	Number of positives correctly classified
FN (False Negatives)	Number of positives incorrectly classified as negatives
FP (False Positives)	Number of negatives incorrectly classified as positives
TN (True Negatives)	Number of negatives correctly classified
precision	$TP/(TP+FP)$
recall	$TP/(TP+FN)$

Table 2.1: Definitions of scoring metrics

The f_1-score is defined as the harmonic mean of the precision and recall:

$$f_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 2 \cdot \frac{TP(2 \cdot TP + FP + FN)}{(TP + FP)(TP + FN)}. \tag{2.25}$$

This way having many false positives or false negatives is heavily penalized.

2.3 Deep Learning and Neural Networks

Tom M. Mitchell defined in 1997 a machine learning algorithm as
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [15]
This is a broad definition, and a lot of widely different algorithms could fit under it. Deep learning clearly does and is a specific area in machine learning using deep artificial neural networks, which are inspired by the biological neural networks in the brain.
The building blocks of neural networks are neurons consisting of weights, here denoted w, and biases, denoted θ. For input data x, the transfer function of a neuron is

$$\hat{y} = \sigma(w \cdot x + \theta) \tag{2.26}$$

where σ is called an *activation function* and is usually a (very) nonlinear function. If w, x and θ are interpreted as vectors then $\cdot$ is the scalar product A network consisting of on single such neuron is called a perceptron. Several connected layers of perceptrons are called a multi-layer perceptron, MLP. The MLP is an example of a feedforward network.
Layers between the input and output layer are called hidden layers. The presence of hidden layers is what allows the network to approximate linearly inseparable problems. MLPs with one or more hidden layers are called deep. The number of neurons in a given layer is called the layer width. When all neurons in a layer are connected to all other neurons in a consecutive layer, this layer is referred to as fully connected. Finding these weights and biases to achieve desired behavior is referred to as training. Training can be split into two broad classes: *Supervised learning*, where labels or ground truth are available. And *Unsupervised learning*, where no labels are available, and the task is usually to find structure in the data.

A slightly more abstract view of neural networks obtained by viewing each node $u^{(i)} = f(A^{(i)})$ as nodes u, associated with a function f, acting on the set of nodes $A^{(i)}$ that are parents of $u^{(i)}$ in the computation graph.
By viewing the neurons as nodes in a computation graph, we can view the problem of training as that of updating the the parameters associated with each node. When these weights are are updated by iteratively minimizing a loss function with respect to these parameters (or weights), this process is called backpropagation. The name backpropagation comes from the fact that one computes the derivative of a forward node $u^{(i)}$ going "backwards" in the computation graph using the chain rule through the parents of $u^{(i)}$, $\mathrm{Pa}(u^{(i)})$:

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}. \tag{2.27}$$

Or written non-recursively:

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{\substack{path(u^{(\pi_1)}, u^{(\pi_1)}, \ldots, u^{(\pi_1)}) \\ \text{from } \pi_1 = j \text{ to } \pi_t = n}} \prod_{k=2}^{t} \frac{\partial u^{(\pi_k)}}{\partial u^{(\pi_{k-1})}} \tag{2.28}$$

The standard activation functions, when used in conjunction with stochastic gradient descent backpropagation, tend to suffer from the problem of vanishing and exploding gradients. This can be seen by inspecting the products on the right-hand side of equation 2.28. This problem is caused by gradients in the update rules described above become vanishingly small or infinite when training either very deep networks or recurrent neural networks. A natural choice for amending this problem is to use ReLu as activation function $\sigma(x) = max(0, x)$[1], or generalizations such as leaky ReLU and ELU, combined with a softmax activation layer.

2.3.1 Convolutional Neural Networks

The use of convolutional neural networks is motivated by three key ideas: sparse interactions (called sparse connectivity), parameter sharing, and equivariant representations [15].

- *Sparse Interactions* - In contrast to the fully connected MLP, CNNs are connected by a *pooling layer* and a *receptive field*. Instead of having all neurons connected to all other neurons in the next layer. This leads to improvements in efficiency since meaningful features can be detected by looking only at small subsets of the input data.
- *Parameter Sharing* - also called tied weights. In a fully connected MLP, each neuron is evaluated once per forward pass. In a CNN, the kernel is applied at every position of the input (separated by stride) except for some edge points, e.g., padding. This helps to reduce the number of parameters in comparison to a fully connected MLP.
- *Equivariant Representations* - A function f is said to be equivariant with respect to a function g if $f(g(x)) = g(f(x))$. In this case, we have equivariance to translation: a translation of the input leads to a corresponding translation

in the output. Intuitively this is desirable since features should not be seen as different based on their position in the input.

There are many different ways to link these layers together other than fully connected layers. One example is the CNN or convolutional layer. The motivation for using these is that the implement the convolution operations, with known properties such as translational invariance: a network defined by a series of convolutional and pooling layers are not sensitive to the position of the "object" it is detecting. Furthermore, aggregating several such layers, where each layer learns to detect a feature, results in a network that learns features of features and so on, e.g., higher-level attributes of the data. In document classification, this would correspond to the network learning the semantic structure of the language used in the document.

2.3.2 Recurrent Neural Models

Another class of neural network models is recurrent neural networks, RNNs. In standard feedforward neural networks such as MLP and CNNs, the outputs are independent of previous inputs. By contrast, in an RNN, the output is dependent on previous inputs. This, combined with the possibility of using variable input sizes, makes RNNs ideal for treating sequences of inputs such as time series or texts, where for instance, sentences are not all of the same size.

Like CNNs, RNNs benefit from parameter sharing. However, where the convolutional operation allows for parameter sharing across "space", e.g., in an image processing, this would be across different parts of the picture, RNNs are able to use parameter sharing across time, i.e., across previous inputs.

The recurrency is achieved by storing information as a *hidden state*, which is then fed back into the network. This hidden can then be seen as the "memory" of the network. Viewed as a directed computation graph, this amounts to allowing for "loops" in the said computation graph.

RNNs can be trained using regular backpropagation, given that the recurrent weights are duplicated spatially in a process called unfolding. This mode of training is then called *Back Propagation Through Time*.

2.3.3 Stochastic Gradient Descent

Nearly all deep learning algorithms use *stochastic gradient descent* to minimize the loss function. Stochastic gradient descent, as the name implies, is a stochastic version of the iterative optimization algorithm gradient descent. The principle of gradient descent is quite simple. To minimize the objective function $f(x)$, keep moving x with small steps in the negative gradient direction. Gradient descent will minimize the objective function until a local minimum is reached. For deep learning algorithms, the loss function to minimize is often an additive loss over all training data samples. The time-complexity to calculate the loss thus grows linearly with the dataset size. In stochastic gradient descent, the gradient of a minibatch is used as an approximation of the gradient of the full training dataset. The size of the minibatch is often quite small but, most importantly, independent of the size of the dataset, meaning that the time complexity to perform one step of stochastic gradient descent

does not increase with the size of the dataset.[15]
A problem with gradient descent is that the algorithm will always get stuck in the first local minimum (unless the step size is unreasonably large), which might be far from the global minimum. While stochastic gradient descent also finds a local minimum and does not guarantee to find the global minimum, it can get out of shallow local minimums due to its stochastic nature.

2.3.4 Regularization

A deep learning algorithm may perform well on the training set it is trained on but under-perform on the validation and training set. This is usually due to *overfitting*, which is caused by the algorithm learning patterns only present in the training set, such as noise. There is a wide array of strategies available for the prevention of overfitting of deep learning models, known as *regularization*-strategies. A couple of these have already been discussed, such as the parameter sharing in Convolutional Neural Networks and minibatches in Stochastic Gradient Descent. We discuss the remaining and most commonly used regularization techniques.

Early Stopping

When training neural networks using a cross-validation scheme, the loss function for both the training and validation set is observed to decrease for the first few epochs. At some point, the training loss will stop decreasing, at which point the training is stopped. Now somewhere before the last epoch, the validation loss will have started to increase. The parameters are selected at the point where the training and validation error starts to diverge, and thus the point with the lowest validation error.

Parameter Norm Penalties

Overfitting can be prevented by imposing penalties in the loss function for high parameter values. This method is also known as weight decay. Most commonly the L_2 norm is used, adding a term $\frac{1}{2}||.||_2^2$ to the loss function, penalizing any deviation from zero. In other contexts, this method is also called *ridge regression* or *Tikhonov regularization*. Another norm for weight decay is the L_1 norm, $||.||_1$. Using this regularization scheme tends to create more sparse solutions.

Drop Out

Drop out involves sampling from an ensemble of subnetworks, where neurons are randomly removed during training. Only the neurons that are not removed are updated. Only sampling from subnetworks prevents co-adaptations between the neurons which has a regularizing effect.

3
Method

In this chapter, the methods used to obtain the results in Chapter 4 are presented. The datasets used, the data preprocessing pipeline, and the architectures of the end models are described. In order to answer the question: *Is weak supervision a feasible approach for annotation datasets with enough fidelity that it can be used to train deep learning classification models without ground truth labels?* Weak supervision label models used to generate probabilistic labels were created using *Snorkel* [42]. A few different end models were then trained on these probabilistic labels, which will hereafter be referred to as *silver labels*. *Gold labels* will be used to refer to expert annotated data.

3.1 Dataset

The data used during this project comes from the MIMIC-III database [20]. The MIMIC-III database is freely available, and all data is de-identified. However, to access the database, a formal request at physionet.org is needed. MIMIC-III is composed of health data from c. 60 000 intensive care unit admissions to a single hospital between 2001 and 2012 [20]. There are a lot of different data available such as laboratory measurements, prescriptions, hospital length of stay, diagnoses, patient information (de-identified), and more. During this project, only the discharge summaries in the form of text documents were considered. The discharge summaries are summaries of all laboratory tests, prescribed medications, patient history, and doctor's diagnosis. Annotations are not necessary for training. However, they are still needed for validation and testing when using weak supervision. A small subset of the MIMIC-III patients notes have been annotated with 15 different *phenotypes* (including *None* and *Unsure*) by clinical researchers [35]. The concepts to identity with weak supervision were chosen from the following phenotype definitions:

- *Advanced Cancer* - Cancers with very high mortality (pancreatic, esophageal, stomach/gastric, biliary, anaplastic)
- *Advanced Heart Disease* - Ejection fraction (EF) less than 30%, severe cardiomyopathy, severe aortic stenosis, any mention of heart transplant (considered for, set to receive, denied).
- *Advanced Lung Disease* - Pulmonary Function Test (PFT) results of Forced Expiratory Volume (FEV1) less than 50% of normal, or Forced Vital Capacity (FVC) less than 70%. Severe chronic obstructive pulmonary disease (COPD), which may be indicated by Gold Stage III-IV. Severe interstitial lung disease

(ILD).

- *Alcohol Abuse* - Recent alcohol abuse history which is an active problem at the time of admission, whether it is the primary cause of admission or not.
- *Chronic Neurological Dystrophies* - Chronic central nervous system (CNS) or spinal cord diseases, including: multiple sclerosis (MS), amyotrophic lateral sclerosis (ALS), muscular dystrophies, myasthenia gravis, Parkinson's Disease, epilepsy, stroke and cerebrovascular accident (CVA) with residual deficits, and various neuromuscular diseases or dystrophies.
- *Chronic Pain Fibromyalgia* - Any etiology of chronic pain (including fibromyalgia) requiring long-term opioid/narcotic medication to control.
- *Dementia* - Alzheimer's and other forms of dementia mentioned in the text.
- *Depression* - Diagnosis of depression, treatment of depression, presentation to the ICU with symptoms of depression including acts of self-harm or suicide.
- *Developmental Delay* - Includes congenital, genetic and idiopathic disabilities.
- *Non Adherence* - Temporary or permanent discontinuation of a treatment, including pharmaceuticals or appointments, without consulting a physician prior to doing so. This includes skipping dialysis appointments or leaving the hospital against medical advice. A patient who sees a physician to discuss adverse events associated with a medication may or may not constitute non-adherence depending on whether or not the treatment was ceased without the physician's consultation.
- *None* - True when no indication is apparent to the annotator.
- *Obesity* - Any mention of obesity as a consideration in the healthcare encounter. Abdominal obesity is not sufficient.
- *Other Substance Abuse* - Intravenous drug abuse, illicit drug use, accidental overdose of psychoactive or narcotic medications. Remote use of marijuana is not sufficient.
- *Schizophrenia and other Psychiatric Disorders* - Psychiatric disorders in DSM-5 classification, including schizophrenia, bipolar and anxiety disorders. Does not include depression.
- *Unsure* - Indicates ambiguity with regard to one or several of the other indications (including "None") on the part of the note annotator.

3.2 Distant Supervision

Since the motivation to use labeling functions is to heuristically model expert knowledge, some notion of "expertise" is needed. The natural way to do this would be to use the insights of the annotators trained in the subject field to write labeling functions that would correspond to how they would make a classification. However, without direct access to trained medical professionals, labeling functions cannot be written in this way. In order to not just simply use potentially uninformed ideas about how the classification was done, i.e., inspecting the data and trying to reverse engineer the thought process of the annotator, we implement a distant supervision scheme. Labeling functions are based on synonyms and subcategories of the phenotype to classify (definitions in Section 3.1) found in the MeSH database (Medical

Subject Headings).

Misspellings or spelling variants in the discharge summaries may occur, therefore, when checking for the occurrence of words fuzzy string matching were used. Specifically, we use the distance between two strings a, and b called the Levenshtein Distance[36], $lev(a, b)$ defined by:

$$lev(a,b) = \begin{cases} |a| & \text{if } |b| = 0 \\ |b| & \text{if } |a| = 0 \\ lev(tail(a), tail(b)) & \text{if } a[0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & \text{otherwise} \end{cases}. \tag{3.1}$$

If the distance is below a fixed threshold, the strings a and b are deemed the same. Here $tail(x)$ is the string of all the characters in x except the first one, and $x[0]$ denotes the first character of x.

3.2.1 The MeSH Database

MeSH, or Medical Subject Headings, is the US National Library of Medicine's controlled vocabulary or thesaurus used for indexing MEDLINE/PubMed[27][30][37][9][32]. It is hierarchically organized in tree structures called MeSH Tree Structures. The vocabulary is comprised of four kinds of terms:

- *Headings* - Also known as descriptors, or main headings, represents concepts found in the biomedical literature. Examples of such concepts include "Brain Edema", and "Kidney". The headings are organized in a tree with 16 main branches. Examples of these trees include: "Anatomy", "Organisms" or "Diseases".
- *Subheadings* - Also called qualifiers, are attached to MeSH the main headings. They describe a specific aspect of a concept such as "diagnosis" or "adverse effects". Subheadings are arranged in logical hierarchical groupings or families such as "therapeutic use → adverse effects, poisoning". Subheadings can be used to describe a specific aspect of a MeSH (main) heading. For instance, "Diagnosis of cough" is indexed as "Cough/diagnosis".
- *Supplementary Concept Records* - A separate thesaurus, or vocabulary, that contains the Supplementary Concept records that do not fit anywhere else, such as substance terms, records, virus and disease terms, etc. A couple of examples are: "Snyder Robins syndrome" and "cordycepin".
- *Publication Characteristics* - or types, is useful when looking for a particular kind of publication. The publication characteristic describes various publication types such as "letter", "Review" or "Randomized Controlled Trial".
- *Check tags* - A special class of headings that primarily covers species, historical time periods, etc.

The branches can have many levels of subheadings, with each subheading occupying a level in the branch hierarchy. Some terms are allowed to appear in more than one

branch of the tree. Indexing is coordinated in such a way that, for example, "The medical staff in teaching hospitals" is indexed as "Medical Staff, Hospital" and "Treatment of HIV infections with HIV protease inhibitors" is indexed as: "HIV Infections/drug therapy" or "HIV Protease Inhibitors/therapeutic use".

3.3 Choice of Phenotypes and Labeling Strategy

Only the phenotypes *Dementia*, *Obesity* and *Advanced Cancer* were considered one at a time (binary classification) during this thesis. The phenotype *Dementia* can be considered the main phenotype since the choice of which types of labeling functions to use was an iterative process of trial and error for this phenotype.

The two types of labeling functions used during this thesis are described in pseudocode in listing 3.1.

labeling_function_from_keyword_NEGATIVE labels every discharge summaries either POSITIVE or NEGATIVE depending on the presence of a specific keyword.

labeling_function_from_keyword_ABSTAIN labels every discharge summaries either POSITIVE or ABSTAIN depending on the presence of a specific keyword.

The thought process was that the most important keywords should label discharge summaries POSITIVE when they are present in the text and NEGATIVE if they are not present. While keywords that are sufficient but not necessary should label discharge summaries POSITIVE when they are present and ABSTAIN from labeling when they are not. The keywords used for the different phenotypes *Dementia*, *Obesity* and *Advanced Cancer* are presented in Table 3.1, 3.2 and 3.3 respectively.

Listing 3.1: Pseudocode of the 2 types of labeling functions used during this thesis. "@" denotes higher order functions.

```
def labeling_function_from_keyword_ABSTAIN(keyword, discharge_summary):
return @(discharge summary) {
        if keyword is (fuzzy) present in the parts
            Discharge Diagnosis, Secondary Diagnosis or
            Medical History of the discharge_summary:
            return POSITIVE
        else:
            return ABSTAIN
        }

def labeling_function_from_keyword_NEGATIVE(keyword, discharge_summary):
return @(discharge summary) {
        if keyword is (fuzzy) present in the parts
            Discharge Diagnosis, Secondary Diagnosis or
            Medical History of the discharge_summary:
            return POSITIVE
        else:
```

```
        return NEGATIVE
    }
```

3.3.1 Keywords Used for Phenotype *Dementia*

In Table 3.1, the keywords used to create labeling functions for the phenotype *Dementia* are presented. The keywords are heavily inspired by the subcategories found in MeSH for the search term "dementia". Sub-subcategories was not considered.

keywords_NEGATIVE	dementia
keywords_ABSTAIN	alzheimer, aids dementia complex, aphasia, creutzfeldt-jakob syndrome, vascular dementia, diffuse neurofibrillary tangles with calcification, frontotemporal lobar degeneration, huntington disease, kluver-bucy syndrome, lewy body disease

Table 3.1: Keywords used to create labeling functions for the phenotype *Dementia*

3.3.2 Keywords Used for Phenotype *Obesity*

In Table 3.2, the keywords used to create labeling functions for the phenotype *Obesity* are presented. The keywords are heavily inspired by the subcategories found in MeSH for the search term "obesity". Sub-subcategories was not considered. Using the definition of the phenotype *Dementia* in Section 3.1, "abdominal" was not used as a keyword.

keywords_NEGATIVE	obesity
keywords_ABSTAIN	hypoventilation syndrome, maternal, metabolically benign, morbid, pediatric, prader-willi syndrome

Table 3.2: Keywords used to create labeling functions for the phenotype *Obesity*

3.3.3 Keywords Used for Phenotype *Advanced Cancer*

In Table 3.3, the keywords used to create labeling functions for the phenotype *Advanced Cancer* are presented. The keywords are heavily inspired by the entry terms and subcategories found in MeSH for the search terms "pancreatic neoplasms", "esophageal neoplasms", "stomach neoplasms", "biliary neoplasms" and "thyroid carcinoma, anaplastic". Sub-subcategories were not considered.

keywords_NEGATIVE	cancer, neoplasm
keywords_ABSTAIN	pancreatic, adenoma, carcinoma, pancreatic intraductal, esophageal, esophageal squamous cell carcinoma, stomach, gastric, biliary, biliary tract, bile duct, gallbladder, anaplastic, thyroid

Table 3.3: Keywords used to create labeling functions for the phenotype *Advanced Cancer*

3.3.4 Labeling Model and Silver Labels

A *Snorkel* [42] labeling model where then trained on the resulting labeling functions for each phenotype. These labeling models where then used to generate the silver labels for the end models to train on. The performance of the labeling models on the test set for each phenotype, *Dementia*. *Obesity* and *Advanced Cancer*, can be seen in Chapter 4 in Figure 4.1, 4.2 and 4.3 respectively.

3.4 Management of Input Data

Modifications of the discharge summaries were needed in order to train the end models since the raw text format of the discharge summaries is not suitable as input. Thus preprocessing, tokenization, and word embedding was performed.

3.4.1 Preprocessing

Preprocessing of the input data was performed by cleaning the discharge summaries from unwanted characters, numbers, newlines, and punctuation, with the purpose of preparing the texts for tokenization. All uppercase letters were also switched to lowercase.

3.4.2 Tokenization

Tokenization is the process of splitting up text into smaller units called tokens. The tokens can, for example, represent words, characters, or subwords. Tokenization of the discharge summaries was performed in order to make it possible for the end models to handle the text inputs. In our case, the tokenization was performed with the built-in tokenizer in Keras. A vocabulary was built by considering the 25000 most commonly occurring words. Then the texts were transformed into sequences of integers where each integer represent a word in the vocabulary. All words not in vocabulary are represented by the same integer. Some information is lost this way, and all word tokenization methods have trouble handling out of vocabulary words. If handling out of vocabulary words is crucial, subword tokenization might be a better alternative. However, once again, the focus of this project is not a state-of-the-art performance for the end model.

3.4.3 Word Embedding

After the tokenization, all semantic similarities between words are lost. This is the reason our first layer in the end model is an embedding layer. Word embedding is representations of words that consider the semantic similarity by having similar representations for words that are semantically similar. By having an embedding layer, the embeddings are learned during training of the end model, which recovers the semantic structure. Another possible approach that could have been done instead is to use pre-trained word embeddings. For example Word2vec [33] pre-trained on clinical notes. When using pre-trained word embeddings, it is important to consider what kind of corpus the word embedding model was trained on. If there is a significant difference from the corpus of application, the word embeddings will not be very accurate.

3.5 End Model Architecture

To validate our approach, we use the generated silver labels on a few different neural network architectures. Since there is no definite theory on what model might be considered "optimal", we choose our models based on experimentation and a few rules of thumb.
Firstly, the high performance but also the high training data requirements of deep learning models motivates us to focus on deep learning models. We, therefore, exclude models such as Support Vector Machines and Random Forest from consideration. CNN models seem to perform well on text classification tasks, and therefore we choose to evaluate our silver labels on a CNN model.
We also validate the silver labels on a Logistic Regression model. The motivation is that its properties are well known with respect to explainability, and it has the property that it may be regarded as a classical (non-neural) model that can be implemented as a neural network model[46]. The Logistic Regression can thus be regarded as the simplest possible type of neural network model and ensures we are not overfitting with respect to model complexity when using more complex models.
Recurrent models are known to be able to exploit the sequential structure of texts[14]. Therefore, we also train a Bidirectional LSTM model. This choice is guided by a few heuristics: standard RNN cells tend to be forgetful, thus using LSTM to aid in this. The bidirectionality ensures that the models "see" both previous context and future context when making a decision.
Summaries of the architectures of the CNN, Logistic Regression and Bidirectional LSTM model can be seen in Figure 3.1, 3.2 and 3.3 respectively. The CNN model can be considered the main model of this thesis.
All models are trained using the binary cross-entropy loss, defined on outputs $\hat{y}_i$ and ground truths (or estimated probabilities) y_i:

$$L(\hat{y}, y) = -\sum_{i=1}^{n} \hat{y}_i \log y_i \tag{3.2}$$

using a vocabulary size of 25 000 and embedding dimension 100. The optimizer used by all the models is the standard Adam optimization algorithm.

3.5.1 CNN

In computer vision, the idea behind using convolutions is to transform adjacent pixels into single values, using filters that are learned [21]. Computation regions are placed in such a way as to cover the whole input space, with a distance between computation regions known as *stride*, with a low stride resulting in overlapping computation regions. Defining inputs as x, and $r_l(x)$ is a region of x at located at x, then the output of a convolutional layer can be written as

$$y = \sigma(W \cdot r_l(x) + b) \tag{3.3}$$

where W and b are weights and biases to be learned and σ is the activation function. This data is passed to a pooling layer, converting several "pixels" into one. This intuition works for text classification as well by considering adjacent words and combinations thereof. The main difference is that while in computer vision, the convolution operations are 2D, the convolutional operations used for text classification are 1D. A summary of the CNN model used in this thesis can be seen in figure 3.1.

Figure 3.1: The Convolutional Neural Network model (CNN) used in this thesis.

3.5.2 Logistic Regression

Logistic regression is reminiscent of linear regression, but uses a binomial response variable [48]. call the response variable Y, and the inputs x_i. The response variable $\pi_i = P(Y_i = 1)$ is then modelled as [6]

$$\pi_i = \frac{\exp(x_i^T \beta_i)}{1 + \exp(x_i^T \beta_i)} = \frac{1}{1 + \exp(-x_i^T \beta_i)}. \tag{3.4}$$

Parameters β_i are usually estimated using maximum likelihood estimation. Logistic regression can, however, be seen as a neural network model, with sigmoid activation functions and no hidden layers. Our neural implementation of Logistic Regression is illustrated in 3.2.

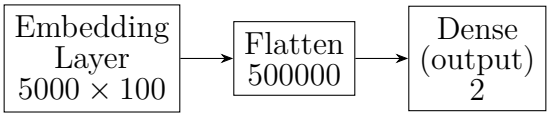


Figure 3.2: The Logistic Regression model used in this thesis.

3.5.3 Bidirectional LSTM

LSTM, or Long-short term memory, is a type of Recurrent Neural Network, RNN. RNNs can, through the use of recurrent links, learn to store information over time, making them a good candidate for sequential processing tasks. [18]. Conventional

RNNs trained using backpropagation through time, however, suffers from exploding and vanishing gradients and tend to be "forgetful" in this sense. LSTMs remedies this by introducing memory cells and gate units. A bidirectional LSTM trains two LSTMs. One is fed data from start to end, and the other LSTM is fed the data chunks in reversed order giving the model increased context when making classifications[31]. The update rules for a bidirectional lstm are given by [51]:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &\sigma(W_o \cdot [h_{t-1}, x_t] + b_0) \\
h_t &= o_t \circ \tanh(C_t).
\end{aligned}
$$

Here h_t is the hidden state at time t, and C_t is the cell state at time t. The weights, W_f, W_i, W_c, W_o and biases b_f, b_i, b_C, b_o are trainable parameters. $\circ$ denotes element-wise multiplication. The output is the hidden state at the last time step. The complete model is shown, along with output dimensions in figure 3.3.

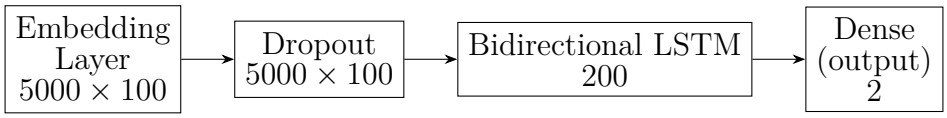


Figure 3.3: The Bidirectional LSTM model used in this thesis.

4

Results

In this chapter, the results obtained from the method described in Chapter 3 are presented. The result for the CNN model (architecture presented in 3.5.1) for phenotype *Dementia* is presented first since this can be considered the main result of the thesis. We compare this classification with that of two more phenotypes for evaluation. We also compare the classification made by the CNN with a Bidirectional LSTM model (presented in section 3.5.3) as well as a Logistic Regression model (presented in section 3.5.2).

4.1 Evaluation of the CNN model

In Figure 4.1 precision-recall curves are shown for the CNN model rained on varying training set sizes, both silver labels, generated with weak supervision, and gold labels created by expert annotators for the phenotype *Dementia*. The precision-recall curves for the labeling model used to generate the silver labels and a baseline no skill classifier are presented as well for comparison.

The model trained on 250 gold labels has significantly worse performance than the other models, however clearly better than the baseline no skill classifier. This indicates that the CNN is a suitable model for the given task. However, as expected needs a larger amount of data to be able to achieve its full potential.

We observe that all models trained on silver labels generalize beyond the labeling model. This was expected, at least for the models with a large number of silver labels, since otherwise, the whole concept of weak supervision would fall apart. However, it also gives us a clear indicator that the implementation of the method works as intended. Exactly how well the model would be able to generalize beyond the silver labels was not known. The difference between a f_1-score of 0.74 for the labeling model and a f_1-score of 0.93 for the CNN model trained on the silver labels annotated by the labeling model is quite a significant improvement.

From Figure 4.1 we can also see that the highest f_1-score of 0.93 was achieved by the CNN trained on 10 000 silver labels. The differences in f_1-score between the models trained on silver labels are pretty small even though the amount of training data varies greatly. It is a bit surprising that the model trained on only 2500 silver labels performs equally well as the model trained on 50000 silver labels, f_1-score of 0.91 and almost as good as the best model, f_1-score of 0.93.

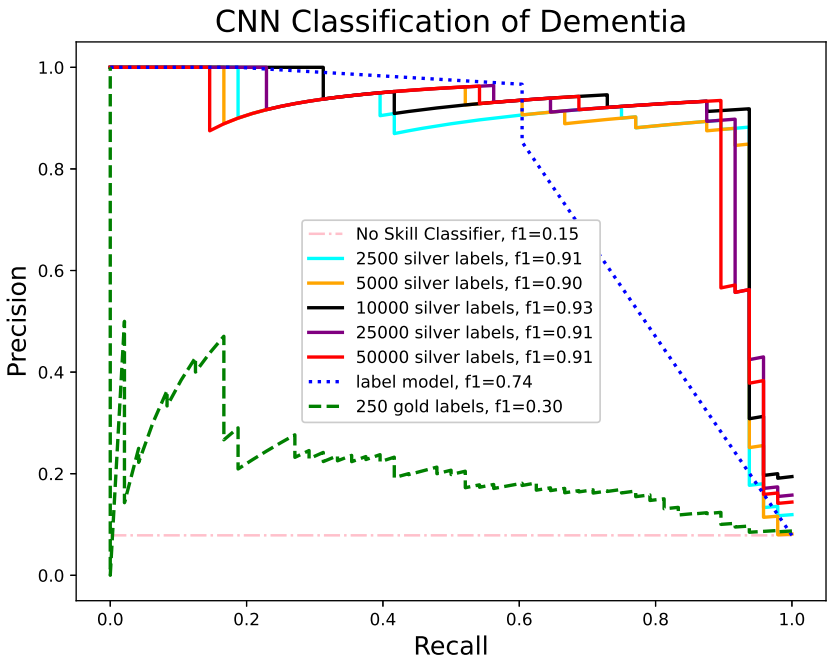


Figure 4.1: Precision-Recall curves for CNN models trained on different amounts and type of data (gold/silver annotations), the label model used and for a baseline no skill classifier for the phenotype *Dementia*

4.1.1 Different Phenotypes

To further evaluate our approach, the precision-recall curves for 2 more phenotypes (*Obesity* and *Advanced Cancer*), classified by CNN models with the same architecture as in Figure 4.1, are presented in Figure 4.2 and in Figure 4.3 respectively.

Looking at the precision-recall curves and f_1-scores for the models trained on gold labels, it seems like the different phenotypes are not equally easy to classify. For example, in Figure 4.2 the model trained on 250 gold labels barely learned anything at all.

It is clear from these figures that the CNN models trained on silver labels are capable of generalizing beyond the silver labels they are trained on as expected. Note however that the CNN models trained on silver labels for the phenotype *Obesity* (Figure 4.2) is significantly worse at generalizing beyond the silver labels compared to the CNN models trained on the phenotype *Dementia* (Figure 4.1).

From Figure 4.2 and 4.3, it is also clear that the models are highly limited by the quality of the silver labels they are trained on.

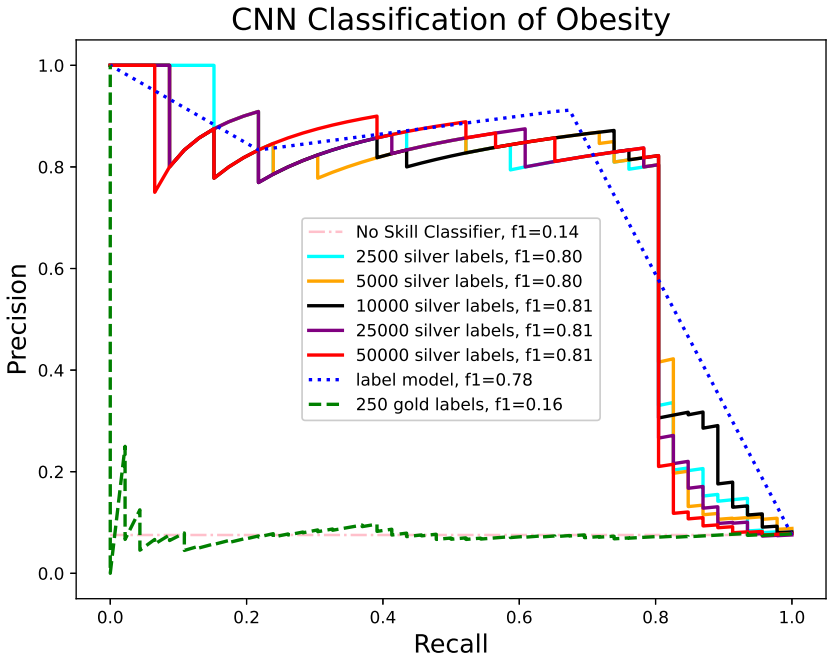


Figure 4.2: Precision-Recall curves for CNN models trained on different amounts and type of data (gold/silver annotations), the label model used and for a baseline no skill classifier for the phenotype *Obesity*

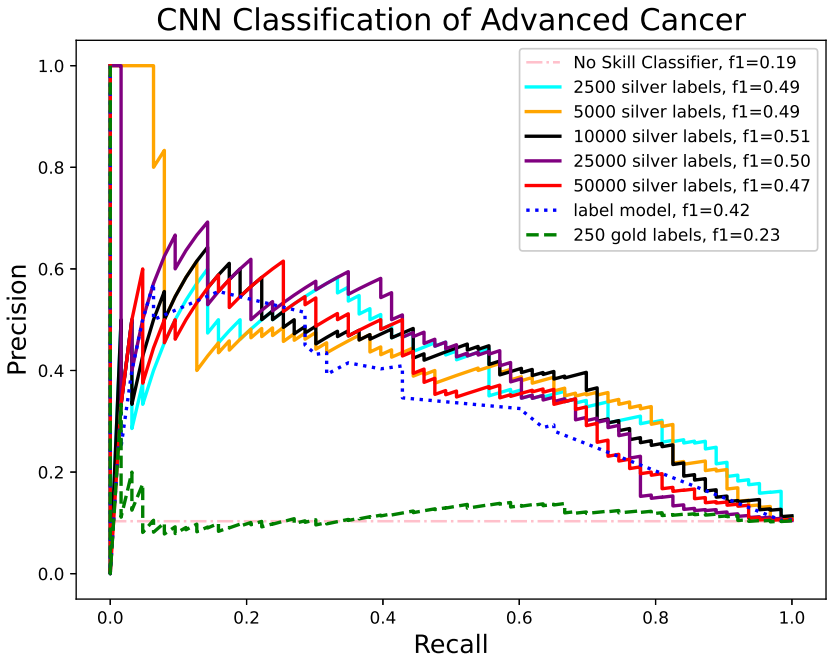


Figure 4.3: Precision-Recall curves for CNN models trained on different amounts and type of data (gold/silver annotations), the label model used and for a baseline no skill classifier for the phenotype *Advanced Cancer*

4.2 Logistic Regression

In Figure 4.4 the precision-recall curves for the phenotype *Dementia* with a Logistic Regression classifier (see Figure 3.2) as end model is presented. It can be observed that while it does learn, it is heavily outperformed by both the CNN (Figure 4.1) and, most notably, the label model itself. The best result is achieved with the maximum number of silver labels, 50 000. It can be concluded that Logistic Regression is not a suitable end model for the given task.

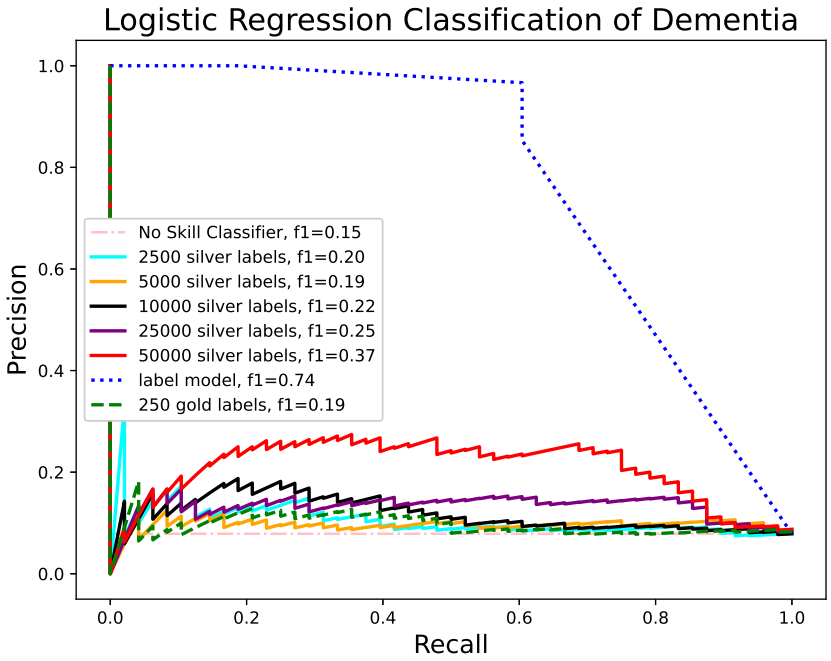


Figure 4.4: Precision-Recall curves for Logistic Regression models trained on different amounts and type of data (gold/silver annotations), the label model used and for a baseline no skill classifier for the phenotype *Dementia*

4.3 Bidirectional LSTM

The precision-recall curves for a Bidirectional LSTM model (see Figure 3.3) as the end model for the phenotype *Dementia* is presented Figure 4.5. The bidirectional LSTM model is clearly capable of generalizing beyond the labeling model when trained on a high amount of data (25 000 and 50 000 silver labels). Note that the highest f_1-score achieved by the different bidirectional LSTM models was 0.85 which is slightly worse than 0.93 for the CNN models (see Figure 4.1).

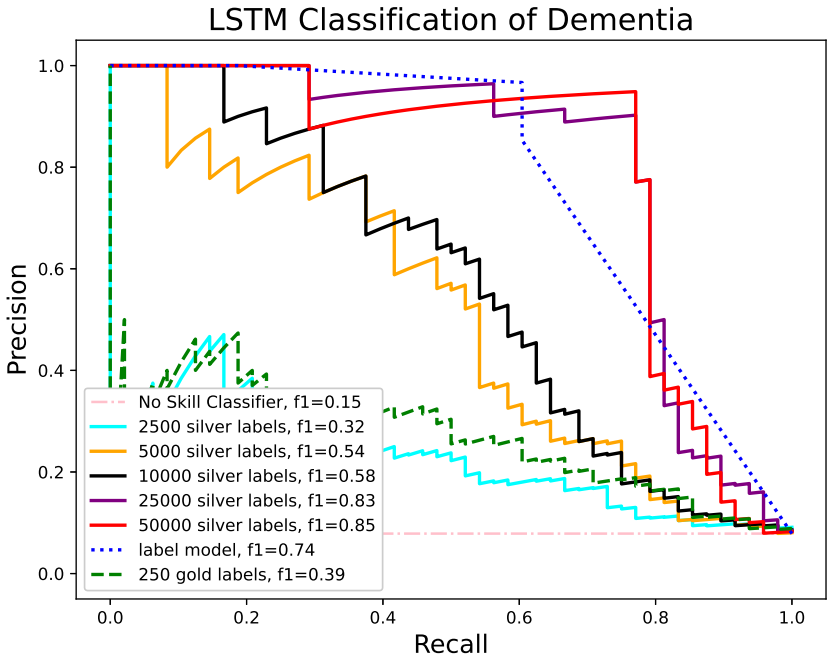


Figure 4.5: Precision-Recall curves for bidirectional LSTM models trained on different amounts and type of data (gold/silver annotations), the label model used and for a baseline no skill classifier for the phenotype *Dementia*

4.4 Summary Table

In Table 4.1, a summary of the results presented in Figure 4.1, 4.2, 4.3, 4.4 and 4.5 is presented for the benefit of the reader. No new information is added.

Table 4.1: Summary table of the results presented in Figure 4.1, 4.2, 4.3, 4.4 and 4.5.

	f_1-score	Datapoints	End Model	Phenotype
Silver Labels	0.91	2500	CNN	*Dementia*
Silver Labels	0.90	5000		
Silver Labels	0.93	10000		
Silver Labels	0.91	25000		
Silver Labels	0.91	50000		
No Skill	0.15	—		
Label Model	0.74	—		
Gold Labels	0.30	250		
Silver Labels	0.80	2500	CNN	*Obesity*
Silver Labels	0.80	5000		
Silver Labels	0.81	10000		
Silver Labels	0.81	25000		
Silver Labels	0.81	50000		
No Skill	0.14	—		
Label Model	0.78	—		
Gold Labels	0.16	250		
Silver Labels	0.49	2500	CNN	*Adv.Cancer*
Silver Labels	0.49	5000		
Silver Labels	0.51	10000		
Silver Labels	0.50	25000		
Silver Labels	0.47	50000		
No Skill	0.19	—		
Label Model	0.42	—		
Gold Labels	0.23	250		
Silver Labels	0.20	2500	Logistic Reg.	*Dementia*
Silver Labels	0.19	5000		
Silver Labels	0.22	10000		
Silver Labels	0.25	25000		
Silver Labels	0.37	50000		
No Skill	0.15	—		
Label Model	0.74	—		
Gold Labels	0.19	250		
Silver Labels	0.32	2500	Bidir. LSTM	*Dementia*
Silver Labels	0.54	5000		
Silver Labels	0.58	10000		
Silver Labels	0.83	25000		
Silver Labels	0.85	50000		
No Skill	0.15	—		
Label Model	0.74	—		
Gold Labels	0.39	250		

5
Discussion

This chapter consists of a discussion of the results presented in Chapter 4 and the method described in Chapter 3 and is ended by a section about future work.

5.1 Evaluation of the CNN model

The main result from Section 4.1 is that the CNN model can clearly generalize its predictions beyond the silver labels it is trained on. This is especially evident in Figure 4.1. From Figure 4.2 and 4.3 it can also be concluded that the performance of the end model is clearly dependent on the quality of the silver labels it is trained on.

Note that the labeling model's f_1-score for classifying *Dementia* (Figure 4.1) is lower than than the f_1-score for the labeling model classifying *Obesity* (Figure 4.2). Since these models are used to create the silver training data, the difference in performance for the CNN models (Figure 4.1 and 4.2) trained on silver labels is interesting. If these phenotypes were equally hard to classify for the end model, one would assume that the models trained on *Obesity* would perform better or equal at worst to the models trained on *Dementia*. Since the opposite result is obtained, f_1-score of 0.81 for the best CNN model trained on *Obesity* and f_1-score of 0.93 for the best CNN model trained on *Dementia*, it can be concluded that the phenotypes are most likely not equally hard to classify. This conclusion is also supported by comparing the f_1-scores for the models trained on 250 silver labels since the model trained on 250 gold labels for the phenotype *Obesity* barely learned anything at all.

It seems like the quality of the silver labels can be lower the easier the classification task is, and satisfactory performance can still be achieved. However, too few phenotypes were compared to be able to conclude it with certainty. We leave it up to future work to explore this possibility.

5.1.1 Labeling

It is possible to assume that it should be easier to generate high-quality silver labels on an easier task to classify. It seems like this is not the case comparing the labeling model's f_1-score of 0.74 for classifying *Dementia* (Figure 4.1) compared to the f_1-score of 0.78 for the labeling model classifying *Obesity* (Figure 4.2). At least not using our method to generate labeling functions. If we also compare to the

f_1-score of 0.42 for the labeling model classifying *Advanced Cancer* it seems like our method to generate labeling functions is not a suitable general method, especially considering that some of the remaining phenotypes would be even harder to generate labeling functions on with our method. Without expert knowledge, it can be hard to create high-quality labeling functions since it seems like the method to generate labeling functions needs to be quite specific to the given task.

The method to generate labeling functions was an iterative process by trial and error for the phenotype *Dementia*, so it is not a coincidence that the labeling works relatively well for this phenotype. The fact that the method works even slightly better for the phenotype *Obesity* in terms of f_1-score suggest that, even though it is not a suitable general method, our method to generate labeling functions could in some cases be a viable approach when expert knowledge is not available.

It is possible that the phenotype *Advanced Cancer* is not actually harder to weakly label with the help of expert knowledge, just unsuitable for the method we used. Expert knowledge would, in this case, optimally be one of the annotators, but any doctor would likely be able to help create relevant labeling functions for any of the phenotypes in the dataset.

5.2 Logistic Regression

From Figure 4.4 it can be concluded that Logistic Regression is not a suitable model for the given task. Note that the highest f_1-score is achieved with the highest amount of silver labels (50 000), further indicating that weak supervision could be a viable approach for deep learning tasks which require a lot of data. It is a bit unclear why the logistic regression model's performance seems to scale with an increasing amount of data in a way that could not be noticed for the CNN model. For example in both Figure 4.1 and 4.2 the highest f_1-score was obtained by the CNN model trained on 10 000 silver labels. Our expectation was that the CNN model would require more data than the Logistic Regression model since it is deeper. Perhaps not too many conclusions about weak supervision should be drawn from the Figure 4.4 since the model is significantly worse than the labeling model, indicating that the Logistic Regression model has trouble learning at all. The need for deeper, more complex models is clear.

5.3 Bidirectional LSTM

The bidirectional LSTM model was mostly implemented to confirm that the silver labels work as training data for different deep learning models and not just the CNN implementation. This is confirmed by the model trained on 25 000 and 50 000 silver labels in Figure 4.5. Comparing the best CNN model from Figure 4.1 and the best bidirectional LSTM from 4.5 it seems like the CNN model is a more suitable model for the given task. This might not be a fair conclusion since not much tuning for the bidirectional LSTM model was made at all compared to the CNN model.

The entire method, from preprocessing data and generating labeling functions to hyperparameter optimization, was an iterative process mainly developed and tuned on the CNN model. However, it is interesting to note that for the bidirectional LSTM, the model trained on 2500 silver labels performs even worse than the model trained on 250 gold labels. On the other hand, for the CNN, the model trained on 2500 silver labels performs almost as well as the best model. This is true for the phenotypes *Obesity* and *Advanced Cancer* as well (see Figure 4.2 and 4.3). In combination with the fact that the bidirectional LSTM model trained on 250 gold labels, with an f_1-score of 0.39, perform better than the CNN model trained on 250 gold labels, with an f_1-score of 0.30, suggests that the CNN model is faster (in the number of data points) at generalizing beyond the silver labels it is trained on. Since the highest f_1-score for the bidirectional LSTM model was achieved by using the highest amount of silver labels available it is not possible to say if the CNN model is actually better at generalizing beyond the silver labels or if the LSTM just needs more silver data to train on to achieve the same performance.

5.4 Reflections on Gold and Silver Labels

The purpose of training the end models on gold labels was to have some sort of baseline comparison and confirm that the architectures of the models were reasonable. The optimal baseline comparison would be training the end models on a large amount of gold annotated data and then compare it to training on the same amount of silver annotated data. Since we did not have access to a lot of gold annotated data, this comparison could not be made. Instead, we had to compare the case of a small amount of gold training data versus a large amount of silver training data. Since this is the use case of Weak Supervision, it is not a bad comparison for visualization. However, not too useful since it comes as no surprise that 250 data points for a highly imbalanced dataset are too few for a deep learning model. It would be preferable to know the performance of the end model trained on a large amount of gold annotated data in order to conclude exactly how well our weak supervision implementation works more accurately. However, being able to achieve a f_1-score of 0.93 when classifying *Dementia* (see Figure 4.1) with a deep CNN model trained on **zero** annotated data suggest that weak supervision could potentially be a viable approach when dealing with large amounts of non-annotated data. Annotations will, however, still be needed for validation and testing, but the number of annotations could potentially be reduced drastically with the use of weak supervision.

One of the significant limitations of Weak Supervision is the need for expert knowledge. This makes it hard for non-experts to generate high-quality silver labels by themselves. However, if anyone manages to create high-quality silver labels for a dataset, the silver labels could be used by anyone basically in the same way as if they were gold labels. The only significant difference when implementing an end model is that the labels are probabilistic.

Since the silver labels annotated by the labeling models are probabilistic the labeling models performances presented in Figure 4.1, 4.2 and 4.3 might not fully capture

the quality of the silver labels. Depending on how certain a labeling model is about a silver label, it will affect how the end model learns from this label. For example, if the silver label is incorrect but only has a probability of 60 percent, it will not negatively affect the loss calculated by the end model as much as if the silver label is incorrect and the labeling model is 99 percent certain. The certainties of the silver labels are not considered directly when comparing the f_1-scores of the labeling models in Figure 4.1, 4.2 and 4.3. This could be a potentially be another reason, in addition to the phenotype *Obesity* being harder to classify than *Dementia*, why the CNN models can not generalize beyond the silver labels as well for the phenotype *Obesity* than for *Dementia*, even though the labeling models $f1$-score for *Obesity* is slightly higher. However, the certainties of the silver labels are indirectly considered since the f_1-score is calculated using the best threshold for prediction, and which threshold is the best will be affected by the certainties of the silver labels.

Another possible explanation could be that the silver labels for the phenotype *Obesity* are not as general as the silver labels for *Dementia*. In other words, maybe the silver labels for *Obesity* that are incorrect are all from a subset of very similar discharge summaries, making the end model converge towards the silver labels instead of generalizing beyond them.

5.5 Future Work

Since the phenotype *Obesity* seems to be harder to classify than *Dementia*, it might also be possible that our CNN architecture is not complex enough for the model to generalize beyond the silver labels in the same way when classifying the phenotype *Obesity* (Figure 4.2) compared to the phenotype *Dementia* (Figure 4.1). It would be interesting to implement a fine-tuned version of a state-of-the-art model such as ClinicalBERT [19] as the end model and train it on the silver labels. We leave it up to future work to explore this possibility.

Since the quality of the silver labels is the most crucial factor for the end model's performance, a possible future work is to evaluate how much the quality of the silver labels affects the performance of the end model to a greater extent than we were able to do in this thesis. One problem is that there is no obvious way to generate a set of labeling functions for a labeling model which gives a desired quality of the silver labels.

There are most likely numerous possible ways to create more suitable labeling functions than those used in this thesis, even without the help of expert knowledge. One approach we tried to implement but did not complete is considering negations. Since our labeling functions only look at the presence of certain words, if the word is present but in a negated context, the labeling functions will use the complete opposite label desired. Since the labeling functions only look at certain parts of the discharge summaries (*Discharge Diagnosis*, *Secondary Diagnosis* and *Medical History*), it might be unlikely that the keywords considered are present in a negated way but not impossible. The labeling functions could possibly also be

improved by considering more parts of the discharge summaries. However, we tried using the labeling functions on the whole discharge summaries first and achieved worse results than only when only considering the parts *Discharge Diagnosis*, *Secondary Diagnosis* and *Medical History*. A possible explanation could be that some parts in the discharge summaries are less critical or more likely to contain negations.

More generally, some experimentation with more complex labeling functions using NLP-preprocessors such as spaCy has been done. These were found to degrade or not improve the label models. The possibility of improving the label models by writing labeling functions over such higher-order primitives (e.g., negations - NegEx) extracted using preprocessors cannot be ruled out, however. The language models of, e.g., spaCy are themselves trained on domain and language-specific corpora, and the standard "off the shelf" models might not be suitable to parse medical literature. A wide array of trained spaCy models are available, such as medspaCy for clinical NLP, negspaCy for negations, "NegEx", or NeuroNer for coreference resolution. Some further experimentation using these or other libraries such as Stanford corenlp could therefore be interesting.

The usage of the MeSH dictionary for finding keywords for a given concept is, at the moment, not completely automatic. A worthwhile endeavor could be the automating of this process using API-queries.
It would also be interesting to compare the performance of the end model when training on different sets of silver labels with the same quality in terms of f_1-score but are generated with labeling models using significantly different labeling functions. This experiment could verify if the potential reason given in section 5.4 to why the CNN models could not generalize beyond the silver labels as well for the phenotype *Obesity* compared to the phenotype *Dementia* is correct or not.

6
Conclusion

Weak supervision seems to be a promising approach for NLP problems within the medical field. From Chapter 4 it can be concluded that end models can generalize beyond the silver labels they are trained on. However, due to the limited amount of gold labels available for the dataset used, it can not be concluded from this thesis if training on silver labels can result in a comparable performance of the end model as using a sufficiently large amount of gold labels. A bottleneck for weak supervision is the need for expert knowledge, making it hard for independent machine learning engineers and students to generate high-quality silver labels themselves. Nevertheless, high-quality silver labels could potentially drastically decrease the need for expert annotations, which is both time-consuming and expensive.

Our general research question was:

Is weak supervision a suitable approach for annotating large sets of unstructured data in the medical domain?

And our specified research question was:

Is weak supervision a feasible approach for annotation of datasets with enough fidelity that it can be used to train deep learning classification models without ground truth labels?

Based on the results from this thesis, the answer to both research questions seems to be *yes*.

Bibliography

[1] Abien Fred Agarap. "Deep learning using rectified linear units (relu)". In: *arXiv preprint arXiv:1803.08375* (2018).

[2] Stephen H Bach et al. "Learning the structure of generative models without labeled data". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 273–282.

[3] Avrim Blum and Tom Mitchell. "Combining labeled and unlabeled data with co-training". In: *Proceedings of the eleventh annual conference on Computational learning theory*. 1998, pp. 92–100.

[4] Vincent S Chen et al. "Slice-based learning: A programming model for residual learning in critical data slices". In: *Advances in neural information processing systems* 32 (2019), p. 9392.

[5] Youngjin Choi. "New form of block matrix inversion". In: *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. 2009, pp. 1952–1957.

[6] Moo K. Chung. *Introduction to logistic regression*. 2020. arXiv: 2008.13567.

[7] Thomas Davenport and Ravi Kalakota. "The potential for artificial intelligence in healthcare". In: *Future healthcare journal* 6.2 (2019), p. 94.

[8] Cassio P De Campos and Qiang Ji. "Efficient structure learning of Bayesian networks using constraints". In: *The Journal of Machine Learning Research* 12 (2011), pp. 663–689.

[9] Ish Kumar Dhammi and Sudhir Kumar. *Medical subject headings (MeSH) terms*. 2014.

[10] Jared A Dunnmon et al. "Cross-modal data programming enables rapid medical machine learning". In: *Patterns* 1.2 (2020), p. 100019.

[11] Andre Esteva et al. "A guide to deep learning in healthcare". In: *Nature medicine* 25.1 (2019), pp. 24–29.

[12] Ronald A Fisher. "On the mathematical foundations of theoretical statistics". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 222.594-604 (1922), pp. 309–368.

[13] Alan E Gelfand. "Gibbs sampling". In: *Journal of the American statistical Association* 95.452 (2000), pp. 1300–1304.

[14] Shalini Ghosh et al. "Contextual lstm (clstm) models for large scale nlp tasks". In: *arXiv preprint arXiv:1602.06291* (2016).

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[16] Kristiina Häyrinen, Kaija Saranto, and Pirkko Nykänen. "Definition, structure, content, use and impacts of electronic health records: a review of the research literature". In: *International journal of medical informatics* 77.5 (2008), pp. 291–304.

[17] Jerónimo Hernández-González, Inaki Inza, and Jose A Lozano. "Weak supervision and other non-standard classification problems: a taxonomy". In: *Pattern Recognition Letters* 69 (2016), pp. 49–55.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[19] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. *ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission*. 2020. arXiv: 1904.05342 [cs.CL].

[20] Alistair EW Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific data* 3 (2016), p. 160035.

[21] Rie Johnson and Tong Zhang. "Effective use of word order for text categorization with convolutional neural networks". In: *arXiv preprint arXiv:1412.1058* (2014).

[22] Siddhartha Jonnalagadda et al. "Enhancing clinical concept extraction with distributional semantics". In: *Journal of biomedical informatics* 45.1 (2012), pp. 129–140.

[23] Daniel Martin Katz, Michael James Bommarito, and Josh Blackman. "Crowdsourcing accurately and robustly predicts Supreme Court decisions". In: *Available at SSRN 3085710* (2017).

[24] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] Volodymyr Kuleshov et al. "A machine-compiled database of genome-wide association studies". In: *Nature communications* 10.1 (2019), pp. 1–8.

[26] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *CoRR* abs/1901.08746 (2019). arXiv: 1901.08746. URL: http://arxiv.org/abs/1901.08746.

[27] Carolyn E Lipscomb. "Medical subject headings (MeSH)". In: *Bulletin of the Medical Library Association* 88.3 (2000), p. 265.

[28] Po-Ling Loh and Martin J Wainwright. "Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses". In: *The Annals of Statistics* (2013), pp. 3022–3049.

[29] Marc Moreno Lopez and Jugal Kalita. "Deep Learning applied to NLP". In: *CoRR* abs/1703.03091 (2017). arXiv: 1703.03091. URL: http://arxiv.org/abs/1703.03091.

[30] Henry J Lowe and G Octo Barnett. "Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches". In: *Jama* 271.14 (1994), pp. 1103–1108.

[31] Xuezhe Ma and Eduard Hovy. "End-to-end sequence labeling via bi-directional lstm-cnns-crf". In: *arXiv preprint arXiv:1603.01354* (2016).

[32] *Medical Subject Headings (MeSH) in MEDLINE/PubMed: A Tutorial*. https://www.nlm.nih.gov/bsd/disted/meshtutorial/introduction/index.html. Accessed: 2021-06-13.

[33] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[34] Mike Mintz et al. "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 1003–1011.

[35] Edward Moseley et al. *Phenotype Annotations for Patient Notes in the MIMIC-III Database*. 2020. URL: https://doi.org/10.13026/txmt-8m40.

[36] Gonzalo Navarro. "A guided tour to approximate string matching". In: *ACM computing surveys (CSUR)* 33.1 (2001), pp. 31–88.

[37] Stuart J Nelson, W Douglas Johnston, and Betsy L Humphreys. "Relationships in medical subject headings (MeSH)". In: *Relationships in the Organization of Knowledge*. Springer, 2001, pp. 171–184.

[38] Andrew Y Ng and Michael I Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". In: *Advances in neural information processing systems*. 2002, pp. 841–848.

[39] Jacqueline Peng et al. "Natural language processing (NLP) tools in extracting biomedical concepts from research articles: a case study on autism spectrum disorder". In: *BMC Medical Informatics and Decision Making* 20.11 (2020), pp. 1–9.

[40] Franz Pernkopf, Robert Peharz, and Sebastian Tschiatschek. "Introduction to probabilistic graphical models". In: *Academic Press Library in Signal Processing*. Vol. 1. Elsevier, 2014, pp. 989–1064.

[41] Alex Ratner et al. *Weak Supervision: A New Programming Paradigm for Machine Learning*. 2019. URL: http://ai.stanford.edu/blog/weak-supervision/.

[42] Alexander Ratner et al. "Snorkel: Rapid Training Data Creation with Weak Supervision". In: *CoRR* abs/1711.10160 (2017). arXiv: 1711.10160. URL: http://arxiv.org/abs/1711.10160.

[43] Alexander J Ratner et al. "Data programming: Creating large training sets, quickly". In: *Advances in neural information processing systems* 29 (2016), pp. 3567–3575.

[44] Alexander J Ratner et al. "Learning to compose domain-specific transformations for data augmentation". In: *Advances in neural information processing systems* 30 (2017), p. 3239.

[45] Alexander Jason Ratner. "Accelerating Machine Learning with Training Data Management". PhD thesis. Stanford University, 2019.

[46] Douglas Redd, Joseph Goulet, and Qing Zeng-Treitler. "Using Explainable Deep Learning and Logistic Regression to Evaluate Complementary and Integrative Health Treatments in Patients with Musculoskeletal Disorders". In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*. 2020.

[47] Irena Spasic and Goran Nenadic. "Clinical text data in machine learning: Systematic review". In: *JMIR medical informatics* 8.3 (2020), e17984.

[48] Sandro Sperandei. "Understanding logistic regression analysis". In: *Biochemia medica* 24.1 (2014), pp. 12–18.

[49] Paroma Varma et al. “Learning dependency structures for weak supervision models”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6418–6427.

[50] Paroma Varma et al. “Multi-resolution weak supervision for sequential data”. In: (2019).

[51] Bo Xu et al. “Leveraging biomedical resources in bi-lstm for drug-drug interaction extraction”. In: *IEEE Access* 6 (2018), pp. 33432–33439.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY