FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:	The spring semester, 2022
Applied Data Science	Open / ~~Confidential~~
Author:	[illegible]
Anne Helland Amundsen	(signature author)

Course coordinator:

Supervisor(s):
Ferhat Özgur Catak, UiS
Jakob Voigt, Neddy

Thesis title:

Machine Learning for Tagging of Educational Content

Credits (ECTS): 30

Keywords:
Neural Networks, LSTM, CNN, BERT, Multi-label Classification, Educational Content, NLP

Pages: 78

+ appendix: 0

Stavanger, 23. June 2022
date/year

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Machine Learning for Tagging of Educational Content

Master's Thesis in Computer Science
by

Anne Helland Amundsen

Internal Supervisor

Ferhat Özgur Catak

External Supervisor

Jakob Voigt

June 23, 2022

Abstract

Online education has become a popular education form in recent years, with its use increasing massively during the COVID-19 pandemic. Neddy is a start-up company created at the start of the COVID-19 pandemic with the aim of making a learning tool for teachers facing distance learning for the first time. They created Addito, an online learning platform for learning content aimed at Norwegian primary and secondary education. This thesis aims to tag educational content from Addito with subject and school year using neural networks based on LSTM, CNN and BERT architectures. The problem is solved as a multi-label classification problem as each learning resource can have multiple tags related to either subject or school year. The results show that the chosen models are able to predict subject labels with sufficient instances in the dataset quite well; however, it fails at predicting the two classes with the least instances in the dataset. None of the models are able to predict school year very well. The best results are obtained using pre-trained BERT models.

Acknowledgements

I would like to thank my supervisors Jakob Voigt and Ferhat Özgur Catak for their guidance and support during the duration of the thesis. I would also like to thank Neddy for letting me write this thesis for them, and providing help, support and Jakob's time.

Contents

Abbreviations

AI	**A**rificial **I**ntelligence
AUC	**A**rea **U**nder receiver operating characteristic **C**urve
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
CNN	**C**onvolutional **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
MLP	**M**ulti **L**ayer **P**erceptron
NLP	**N**atural **L**anguage **P**rocessing
RNN	**R**ecurrent **N**eural **N**etwork

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Distance learning has been an education form for decades, with the prevalence increasing massively due to the COVID-19 pandemic. Distance learning describes any learning that happens without the students being physically present in the lesson. Historically, this described correspondence courses in which students would communicate with their schools or teachers by mail [8]. Online education has been the most prevalent type of distance learning in recent years, and several universities have offered online courses for years. The COVID-19 pandemic resulted in the closure of the vast majority of schools worldwide, forcing them to move the lessons online. This included lessons on video conferencing tools such as Zoom or Microsoft Teams and the use of online learning resources [9].

Neddy [10] is a start-up company created at the start of the COVID-19 pandemic to make a learning tool for teachers in Norway facing distance learning for the first time. Addito [11] was one of Neddy's first projects, and it is an online learning platform specializing in learning content for Norwegian primary and secondary education. The site is managed by teachers and educators in Norwegian schools, and any teachers in Norwegian schools can add content.

Neddy is currently creating a planning tool for teachers that will incorporate learning content from Addito as well as partner sites. The motivation for this thesis is to use machine learning techniques to tag content with suitable subject and school year labels to make navigation easier for teachers and present them with relevant content.

1.2 Problem Definition

The over-arching goal of this thesis is to automatically tag learning content from Addito with the most suitable subject and school years to make sure that the teachers using Addito and Neddy are presented with relevant content. The learning content can have more than one tag for subject label or more than one for school year label. This will therefore be considered a multi-label classification problem. The learning content on Addito is primarily in Norwegian, and therefore methods that will work for the Norwegian language will have to be established. This thesis will use neural networks to try to tag the content with school year or subject by using metadata from the Addito website, making it a text classification problem.

1.3 Challenges

There are fewer resources for natural language processing (NLP) in Norwegian as most of the community has focused on English or other major languages. This means that a lot of the state-of-the-art approaches are not available for datasets in Norwegian. The language technology group at the University of Oslo [12] has contributed a lot to this field in the last few years and helped alleviate this challenge.

The learning resources to be tagged could be labeled with more than one subject or school year, making this a multi-label classification problem. In addition, the dataset is relatively small, and the classes are imbalanced, which could create challenges. Multi-label classification is an important yet challenging task in natural language processing. It is more complex than single-label classification in that the labels tend to be correlated. Furthermore, existing machine learning methods tend to ignore the correlations between labels [13].

1.4 Outline

The rest of the thesis is structured as follows:

Chapter 2, Background will give the necessary background knowledge regarding machine learning, deep learning and resources used in this thesis.

Chapter 3, Dataset and Solution Approach describes the dataset and the analysis and preprocessing of this, as well as the methodology of the proposed solution approach.

Chapter 4, Experimental Results gives an overview of the experimental results from the methods in outlined Chapter 3.

Chapter 5, Discussion discusses the experimental results given in Chapter 4.

Chapter 6, Conclusion sums up the approach and results and proposes further work.

Chapter 2

Background

This chapter will present the theoretical background most relevant for this thesis. Firstly, there will be a brief presentation of the Norwegian education system, generally about online learning platforms, and Neddy and Addito in particular. NLP is then presented before introducing machine learning and neural networks, which will be used to build models in the next chapter. Finally, in the last section, some resources that have been useful to this thesis will be presented.

2.1 Education and Online Learning

This thesis is written in collaboration with Neddy, which will be introduced further in Section 2.1.3. Neddy focuses on learning content for primary and secondary education in Norway, hence this will be the focus of the background information given in this thesis.

2.1.1 Norwegian School System - Primary and Secondary Education

All people in Norway are entitled to primary and lower secondary education, and for children and young people, this is also an obligation. Everyone who completes this education is further entitled to upper secondary education, qualifying for further studies or a vocation. Primary and secondary education in Norway typically lasts a total of 13 years [14].

Primary and secondary education is split into three main parts:

- Primary school (years 1-7)
- Lower secondary school (years 8-10)
- Upper secondary school (year 11-13, also called VG1-VG3)

In addition, there is adult education created for adults who have not finished parts of primary or secondary school.

The Norwegian Directorate for Education and Training is responsible for developing kindergarten and primary and secondary education. They have developed learning plans for subjects taught at all levels to ensure that all students receive the high-quality education they are entitled to and that the curricula are aligned across the country [15].

2.1.2 Online Learning Platforms

Online learning has become an important part of education in recent years. Its popularity has risen steadily in the last decade, with the COVID-19 pandemic accelerating it even further in the last two years, as most students were forced into some form of distance learning.

An online learning platform is a web space or portal for educational content and resources that offer a student everything they need in one place. This can include lectures, resources, opportunities to meet and chat with other students, monitoring of student progress, and more [16].

Traditional online learning can offer a wide range of benefits:

- The comfort of working from home
- Self-paced learning
- Improved technical, communication, and critical thinking skills
- Lower cost
- Better time management skills

Most of the online learning platforms available for Norwegian students are focused on higher education and are not available in Norwegian. Some popular online learning platforms include Coursera, Khan Academy and Udemy.

2.1.3 Neddy

Neddy [10] is a start-up company that offers an online learning platform and planning tool focused on primary and secondary education in Norway. It was created during the start of the COVID-19 pandemic in Norway when schools had to close and start remote teaching online. It was created with teachers and students in the Norwegian school system in mind.

Neddy's first project was to create an online learning platform called Addito [11], which was populated with learning content by teachers in Norwegian schools. This gave teachers access to share learning resources across the country when the schools closed, and they had to offer distance learning. Neddy then focused on creating a planning tool for teachers to plan their work. The goal is for the platform to recommend learning content relevant to the topic and the learning goals set by the subject's learning plan. Students should then be able to access and interact with this content on the platform. Neddy both offers content from Addito and other partner learning platforms.

2.1.4 Addito

Addito [11] is Neddy's own platform with learning content. It is managed by teachers and educators in the Norwegian school system, and content can be added by any teacher in Norwegian schools. The learning content was then shared with teachers all over Norway to help the teachers create learning content that would fit the new situation.

The Addito database is continuously updated with new content from teachers. The content can be filtered on subject and school year. Good content will be given an apple, equivalent to "like" on other sites, and can be sorted based on the number of apples, newest or oldest.

The teachers will fill out a form with information about the content to add content to the site. They will be prompted to enter a title and description, tag for school year and subject, and optionally other tags created by the teacher. In addition, the teacher can add files or links to the content. All these fields are optional.

2.2 Natural Language Processing

NLP is the computer's ability to understand human language as spoken and written, referred to as natural language. It is an increasingly important component of artificial intelligence (AI). NLP is split into two main phases: data preprocessing and algorithm development [17].

The data preprocessing phase involves preparing and "cleaning" text data in a form that a machine can analyze. It highlights features and arranges data in a form with which an algorithm can work. There are several ways to do this, including:

- **tokenization:** Splitting of a text into words or sub-words, which are then converted to ids through a look-up table.
- **Stopword removal:** Stopwords are commonly used words that do not give any context or meaning to the sentence. For example, in English, words such as "the", "it", and "and" are considered stopwords. If not removed, the algorithm may place

too much weight on these words as they frequently occur in most texts, which may decrease the algorithm's accuracy.

- **Stemming:** Stemming is the process of reducing a word to its word stem or root so that words of a similar kind lie under a common stem. For example, the words "send", "sent", and "sending" lie under the same stem, "send". This might not necessarily mean that the word is reduced to its dictionary root, which may lead to problems with over-stemming and under-stemming. Over-stemming is removing a larger part of the word than required, leading to two words with different meanings being reduced to the same stem. Under-stemming is the opposite, not reducing the word enough, leading to two words with the same dictionary root getting different stems [18].
- **Lemmatization:** Lemmatization is the process of reducing a word to its dictionary root. It is a more complex process than stemming that requires a higher knowledge of a language and often requires access to a lexical database.

Once data preprocessing is complete, an algorithm to process it is developed. The two main types of natural processing algorithms that are commonly used are:

- **Rules-based system:** This system is based on carefully designed linguistic rules. It was one of the earliest approaches in the development of NLP and is still in use.
- **Machine learning-based system:** Machine learning algorithms use statistical methods to perform tasks based on the training data they are fed. They adjust their methods as more data is processed, learning from the data during the training. NLP hones its own rules through repeated processing and learning by using a combination of machine learning, deep learning, and neural networks [17].

2.2.1 Word Embeddings

A word embedding is a learned vector representation of a text where words with a similar meaning have a similar representation. Every word in a training corpus is mapped to a multidimensional vector and is learned based on the usage of the words [19].

Popular word embedding algorithms are Word2Vec, GloVe, and FastText. Researchers at Google developed Word2Vec in 2013, and it uses neural networks to calculate word embeddings based on words' context. GloVe was published by researchers at Stanford one year later and focuses on co-occurrences over the whole corpus. Its embeddings relate to the probabilities that two words appear together. The last method, FastText, was developed by Facebook in 2016, and it improves Word2Vec by taking word parts into account as well. This enables training of words on smaller datasets and generalization to unknown words. The word embeddings outputted by FastText look very similar to the ones provided by Word2Vec. However, they are not calculated directly; instead, they are a combination of lower-level embeddings [20].

2.3 Machine Learning and Neural Networks

Machine learning is a type of AI where the goal is for the computer to learn from prior experience. It is split into three main sub-areas: supervised learning, unsupervised learning, and reinforcement learning.

Modern AI is mainly based on the study of deep learning and artificial neural networks, which will be the focus of this thesis. Artificial neural networks arose as an attempt to model the human brain structure and functioning. The network is composed of several simple units, called neurons, arranged in a certain topology and connected with each other. Neurons are organized into layers. Depending on their position, layers are denominated as input layer, hidden layer, or output layer. Neural networks with multiple hidden layers are referred to as deep neural networks, and a simple neural network with one hidden layer is shown in Figure 2.1 [1]. The following sections will present a few different configurations of neural networks.

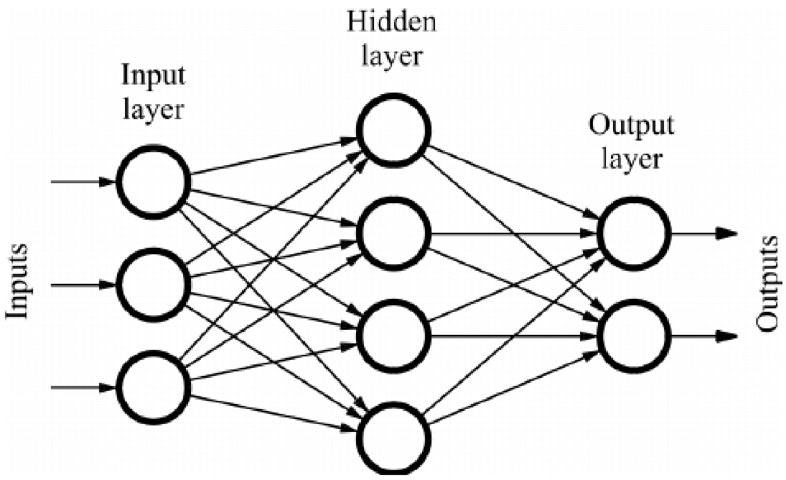


Figure 2.1: Simple feed-forward neural network with one hidden layer.

2.3.1 Feed-forward Neural Networks

The feed-forward neural network is the simplest form of an artificial neural network. In a feed-forward neural network, the information flow is forwards from the input layer, through the hidden layers, and to the output layer, as shown in Figure 2.1 for a network with one hidden layer. A feed-forward network with multiple hidden layers is often referred to as a multi-layer perceptron. The neurons in each layer are only connected to those in the following layer. Node outputs, y are computed by first taking the sum of the input weights w_i multiplied with their respective input x_i plus an independent term called bias, b. This is then subject to an activation function, g, before giving the output, as shown in Equation 2.1.

$$y = g\left(b_1 + \sum_{i=1}^{m} x_i w_i\right) \tag{2.1}$$

This is further shown in Figure 2.2. These activation functions are usually non-linear; however, purely linear functions can be used for this purpose as well, especially in output layers [1].

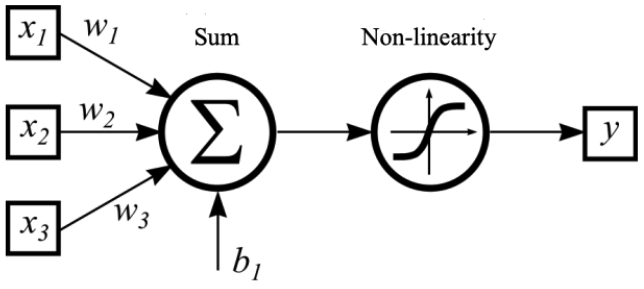


Figure 2.2: A typical neuron consists of a linear combination of the input followed by a non-linear activation function [1].

Some frequently used activation functions are the step, sigmoid and hyperbolic tangent (tanh) functions shown in Figure 2.3.

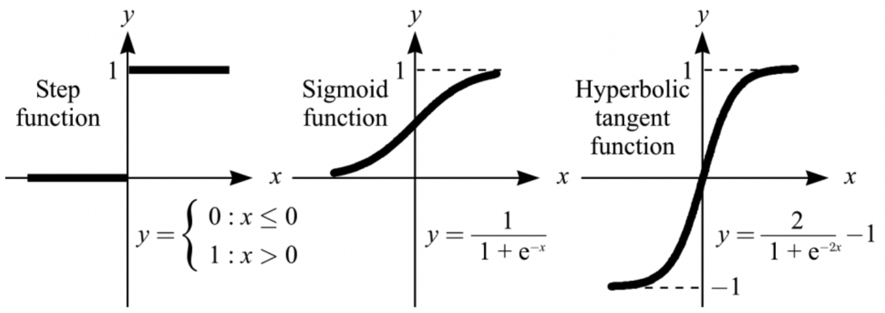


Figure 2.3: Common activation functions [1].

To be able to learn, the network uses back-propagation to update the weights between each layer. While forward-propagation is used for computing outputs, the purpose of back-propagation is to minimize the loss function. The loss function represents the differences between the predicted values and the target values. When the weights have been updated through back-propagation, the training data is fed through the network again, and the loss is calculated again until training is finished.

2.3.2 Recurrent Neural Network

A recurrent neural network (RNN) is an artificial neural network that builds on the feed-forward neural network. It is mainly used to detect patterns in a sequence of data, such as handwriting, genomes, text, or numerical time series. The main improvement to the feed-forward neural network is the inclusion of a recurrent connection, or cycle, from the end of the hidden layer to the beginning, as is visualized in Figure 2.4. This allows the network to extend the functionality to consider previous inputs $\mathbf{X}_{0:t-1}$ and not only current input $\mathbf{X}_t$ [2].

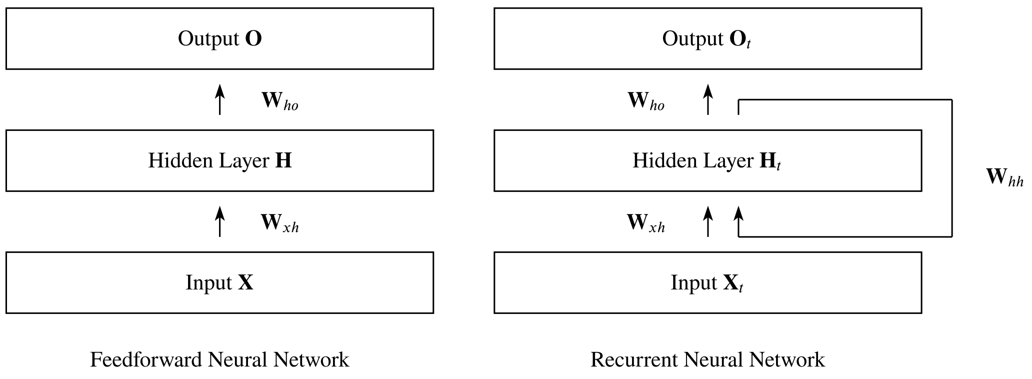


Figure 2.4: Visualization of differences between feed-forward neural networks and recurrent neural networks [2].

As in most neural networks, vanishing or exploding gradients is a key problem of RNNs. Gradients are values used to update the weights of a neural network. The vanishing gradient problem is when the gradient becomes smaller as it backpropagates through time. If a gradient value becomes extremely small, it will prevent the network from learning new weights. Exploding gradients is the opposite problem where large gradients accumulate and result in very large updates to the model weights during training.

2.3.3 Long Short Term Memory Networks

Long short term memory units (LSTMs) are an advanced form of RNN designed to handle the vanishing gradient problem properly [2]. A typical LSTM unit comprises a cell, a forget gate, an input gate, and an output gate, as shown in Figure 2.5. The cell stores values over arbitrary time intervals in its memory and the three gates control the flow of information into and out of the cell. The gates can learn what data in a sequence is important to keep or throw away. By doing this, it can pass relevant information down the long chain of sequences to make predictions. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram in Figure 2.5. It can be visualized as a conveyor belt through which information can flow unchanged. Information can be added to or removed from the cell state in LSTM and is regulated by gates. The sigmoid layer, denoted by σ, gives out numbers between zero and one, where zero means that nothing should be let through and one means everything should be let through. The first gate is the forget gate which decides what information to throw away from the cell state. The input gate decides what new information to store in the cell state, and the output gate gives the output based on the input and memory [21].

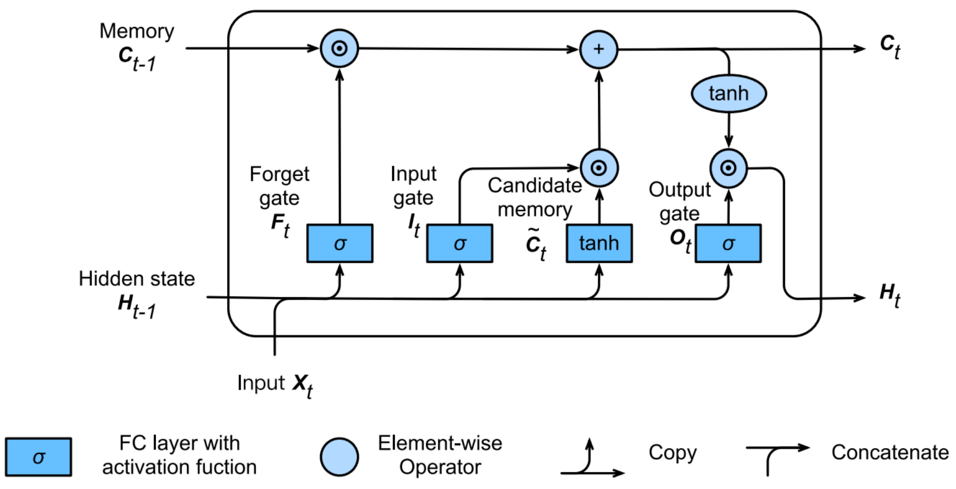


Figure 2.5: Visualization of the repearing component of an LSTM Network [2]

2.3.4 Convolutional Neural Networks

Convolutional neural networks (CNN) is a special kind of feed-forward network and is often used where the input data has a grid-like topology, such as images which can be thought of as a 2D grid of pixels. The network derives its name from the mathematical operation called convolutions, and CNNs are simply neural networks that use convolutions in place of general matrix multiplication in at least one of their layers [22]. A CNN gains strength by requiring fewer weights than a fully connected network by preserving the spatial relationship in the grid and using small parts of the input to draw features.

A CNN consists of 3 types of layers: convolutional layers, pooling layers, and fully connected layers, as shown in Figure 2.6. The convolutional layers extract features, typically using a combination of linear and non-linear operations, i.e., convolution operation and activation function. Parameters of the convolutional layer are filter size, kernel size, and stride. The filter size is how many neurons there are in the layer, the kernel size is how large the feature mapping is, and the stride is how many places the feature mapping should move at one time. The pooling layer downsamples the features found in the convolutional layer and keeps the best performing features. In the end, there is a fully connected layer that flattens the structure and, with its weights and biases, produces a final output [3].

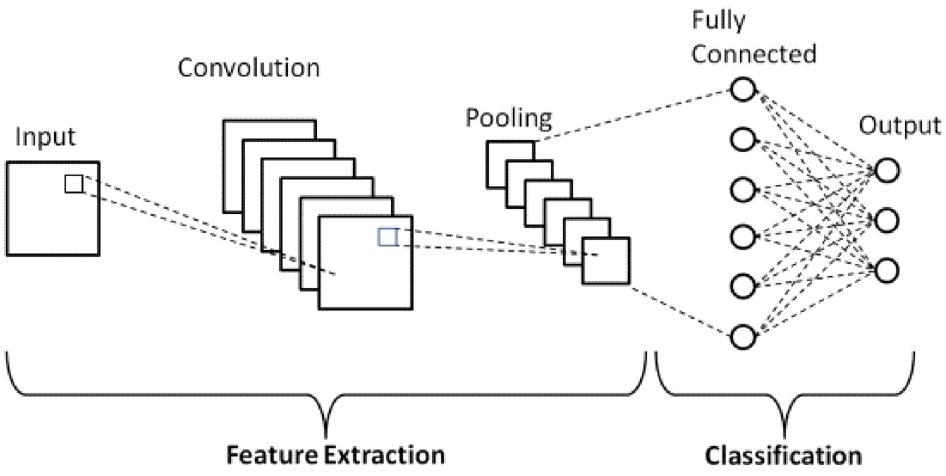


Figure 2.6: General architecture of CNN [3].

2.3.5 Attention Mechanism and Transformers

The attention mechanism is partly motivated by human visual focus and peripheral perception. It is what allows humans to focus on a particular region to achieve high resolution while adjacent objects are perceived with a relatively low resolution [2]. The attention mechanism learns contextual relationships between words in a text. It makes use of three main components, the queries, the keys, and the values. For a sequence of words, the attention mechanism takes the query vector attributed to a specific word in the sequence and scores it against each key in the database to get the weights. The weighted sum of the values dependent on the query is then computed. This is to retain focus on the words that are relevant to the query and produces an attention output for the word under consideration. Attention mechanisms were used with other architectures, but the Transformer [4] introduced a new architecture relying almost exclusively on attention. The Transformer uses self-attention, which computes the relationship between every word in the input sequence. The Transformer architecture is shown in Figure 2.7, where each gray unit on the left is an encoder, and the gray unit on the right is a decoder. The Transformers have the same number of encoders as decoders [2].

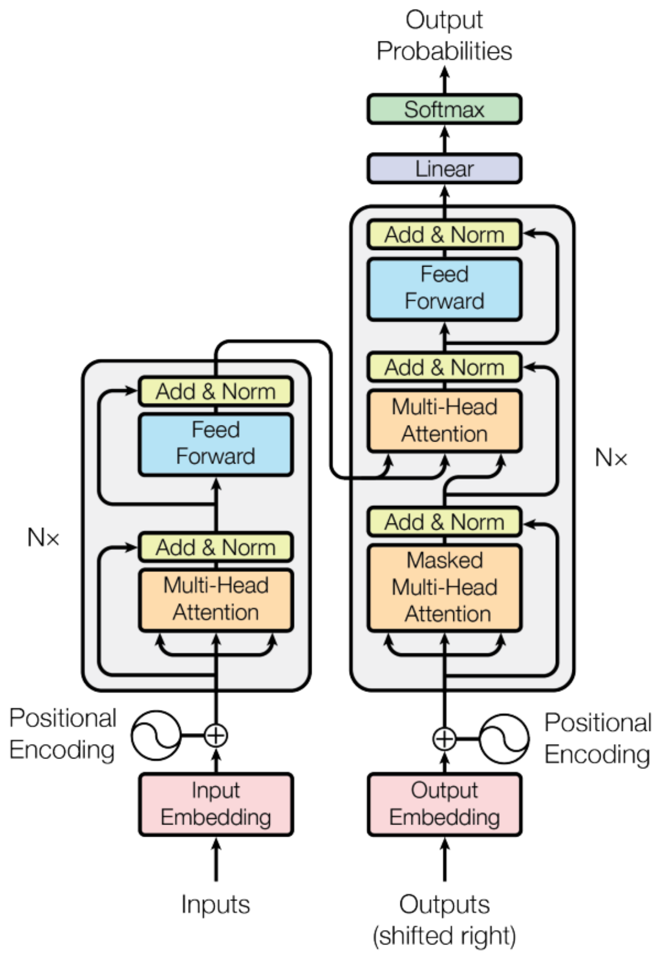


Figure 2.7: The transformer model architecture with encoder (left) and decoder (right) [4].

2.3.6 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model that has been a milestone in the field of NLP. BERT is designed to pre-train bidirectionally by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model only needs an additional output layer for fine-tuning to create state-of-the-art models for a wide range of tasks [5]. BERT relies on the encoder part of a Transformer. The input to the encoder for BERT is a sequence of tokens. These are first converted into vectors and then processed in the neural network [23].

BERTs model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in [4]. BERT was originally released in two versions, $BERT_{BASE}$ and $BERT_{LARGE}$, where the former has L=12 layers (transformer blocks), a hidden size of H=768, and A=12 self-attention heads, giving a total of 110M parameters.

$BERT_{LARGE}$ has L=24 layers, H=1024 hidden size, and A=16 self-attention heads, giving a total of 340M parameters [5].

There are two steps in the BERT framework: *pre-training* and *fine-tuning*. Pre-training is performed using two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM is the process of masking some percentage of the input tokens at random and then predicting those masked tokens. NSP is used to learn the relationship between sentences during pre-training. During training, the input sentence pairs, A and B, are chosen so that B is 50% of the time the actual sentence that follows A, and 50% of the time it is a random sentence from the corpus. The model then predicts whether the sentence is the actual next sentence in the corpus [5].

To help the model distinguish between the two training sentences, the input is preprocessed according to Figure 2.8. A [CLS] token is added at the beginning of the first sentence and a [SEP] token is inserted between sentences. A segment embedding is added indicating whether it is sentence A or B. Lastly, a position embedding is added indicating its position in the sequence. The input embeddings is then the sum of the token embeddings, the segmentation embeddings, and the position embeddings [23].

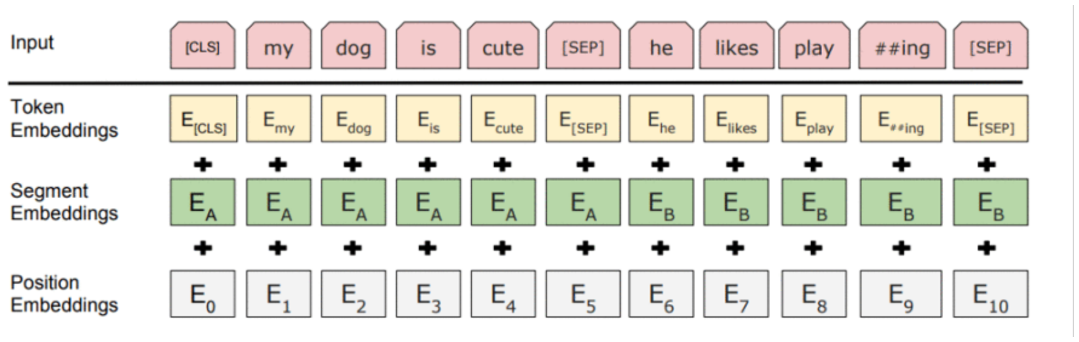

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.8: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings [5].

By swapping out the appropriate inputs and outputs, the self-attention mechanism in the Transformer allows BERT to model many different tasks, whether they involve single text or text pairs. This makes fine-tuning straightforward, and compared to pre-training, fine-tuning is relatively computationally inexpensive [5].

2.4 Resources

2.4.1 Environment and Packages

Due to its ease of use and large machine learning community, all models and scripts are written in Python. To aid the development of the models, several Python packages were used, and the most important ones are:

- **Natural Language ToolKit (NLTK)** is a leading platform for building Python programs for natural language processing. It provides text processing libraries for most tasks within NLP, such as stopword removal, stemming, tokenization, parsing, and semantic reasoning [24]. NLTK is mostly focused on English, however it includes some libraries for Norwegian language.
- **SpaCy** offers industrial-strength natural language processing for Python [25]. It includes trained models and pipelines for several languages that can be used for most tasks within NLP, inlcuding, but not limited to, tokenization, stopword removal, lemmatization, part-of-speech tagging, and named entity recognition.
- **Keras** [26] is a deep learning API written in Python, built on top of the machine learning platform Tensorflow 2 [27]. It was developed with a focus on easy implementation and fast prototyping.
- **Scikit-Learn** [28] is a package with tools for machine learning in Python. It includes simple and efficient tools for predictive data analysis and evaluation of results.
- **iterative-stratification** is a package made to provide scikit-learn compatible cross validators for multi-label data [29].

2.4.2 Hugging Face

Hugging Face is an AI community that provides open-source NLP technologies [30]. Through transformers, they host pre-trained models for anyone to use. These models can be fine-tuned to fit most use-cases within NLP. Companies like Facebook AI, Microsoft, and Google AI use Hugging Face and make some of their models public.

2.4.3 Language Technology Group Oslo

The Language Technology Group (LTG) Oslo [12] is a group that does research on a number of topics related to NLP and computational linguistics. They are a part of the University of Oslo and an important priority for LTG in recent years has been to create NLP resources for the Norwegian language. Through the Norwegian Large-scale Language Models (NorLM) initiative [31], LTG has provided several large-scale language models for Norwegian, including models based in the ELMo (Embeddings from Language Models) [32] and BERT architectures. The pre-trained BERT based models, called NorBERT, have been made available on Hugging Face and are presented in the following section.

NorBERT

NorBERT is a series of BERT deep learning language models trained from scratch for Norwegian [33]. The models are part of the ongoing NorLM initiative [31] for very large contextualized Norwegian language models and associated tools and recipes. NorBERT was based on $BERT_{BASE}$ from Google AI [5].

NorBERT 1 training corpora:

- *Norsk Aviskorpus* (NAK): 1.7 billion words
- *Bokmål Wikipedia*: 160 million words
- *Nynorsk Wikipedia*: 40 million words

In total about 2 billion word tokens in 203 million sentences, both in Bokmål and Nynorsk.

NorBERT 2 training corpora:

- *Norwegian Colossal Corpus* (NCC), non-copyrighted part: 5 billion words
- *C4 web-crawled corpus* Norwegian Part, random sample: approximately 9.5 billion words

In total about 15 billion word tokens in about 1 billion sentences, both in Bokmål and Nynorsk.

There are two official standards for written Norwegian; Bokmål, the main variety and Nynorsk, used by 10-15% of the Norwegian population. NorBERT is trained jointly on both varieties with the minority variant, Nynorsk, represented by comparatively less data than Bokmål, reflecting natural usage [33].

Chapter 3

Dataset and Solution Approach

In this chapter, firstly, the dataset is presented with data format, and data exploration is performed. The problem is approached as a multi-label classification problem to predict labels for school year and subject where each record may be suitable for more than one school year and more than one subject. The proposed solution is then presented, and the data preprocessing steps are outlined for these methods. The last section presents the performance measures used to evaluate the performance of the models.

3.1 Dataset and Data Analysis

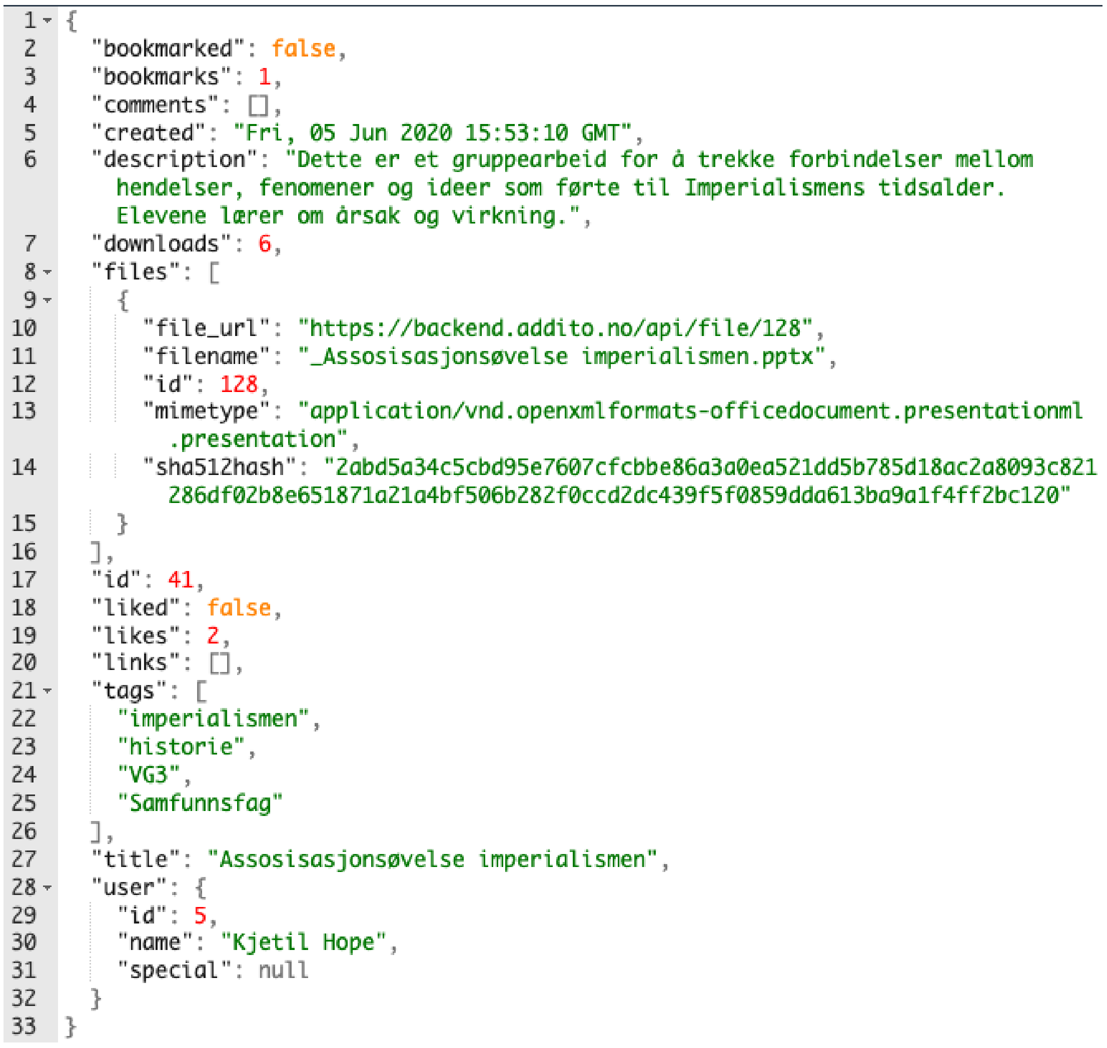
The dataset used for this thesis is extracted from the Addito website as a json file. The format of the data file is presented in the next section with an overview of the information available. Then the data will be explored to find the best data to input into the proposed models.

3.1.1 Data Format

The dataset collected from Addito includes the metadata with the following subcategories:

- bookmarked
- comments
- created
- description
- downloads
- files
- id
- liked
- likes
- links
- tags
- title
- user

A typical record can be seen in Figure 3.1. The metadata that is found to be most useful for this thesis is the description, files, tags and title.



```
{
  "bookmarked": false,
  "bookmarks": 1,
  "comments": [],
  "created": "Fri, 05 Jun 2020 15:53:10 GMT",
  "description": "Dette er et gruppearbeid for å trekke forbindelser mellom
    hendelser, fenomener og ideer som førte til Imperialismens tidsalder.
    Elevene lærer om årsak og virkning.",
  "downloads": 6,
  "files": [
    {
      "file_url": "https://backend.addito.no/api/file/128",
      "filename": "_Assosisasjonsøvelse imperialismen.pptx",
      "id": 128,
      "mimetype": "application/vnd.openxmlformats-officedocument.presentationml
        .presentation",
      "sha512hash": "2abd5a34c5cbd95e7607cfcbbe86a3a0ea521dd5b785d18ac2a8093c821
        286df02b8e651871a21a4bf506b282f0ccd2dc439f5f0859dda613ba9a1f4ff2bc120"
    }
  ],
  "id": 41,
  "liked": false,
  "likes": 2,
  "links": [],
  "tags": [
    "imperialismen",
    "historie",
    "VG3",
    "Samfunnsfag"
  ],
  "title": "Assosisasjonsøvelse imperialismen",
  "user": {
    "id": 5,
    "name": "Kjetil Hope",
    "special": null
  }
}
```

Figure 3.1: Example of metadata for one record

The tags data consists of a list of tags entered by the teacher. This list can include both predefined tags for school year and subject, which will be used as training labels, and also other tags that the teacher can write freely. For example, in Figure 3.1, "VG3" is a school year label and "Samfunnsfag" is a subject label. The various labels are discussed more in the following section.

The file's metadata includes filenames and URLs to the file in question. From the filenames, we get the file type as the filename ending. One record can have multiple files, with the same or different file types, or no files at all.

3.1.2 Data Exploration

The Addito website has been updated with new content throughout the semester. The total number of entries as of 17.05.22 is 1170 individual learning resources. The metadata was last collected at this date, and any updates after this have not been included in the thesis. The most useful parts of the metadata were found to be the tags, as this is what will be predicted, and title, description and files, used as input to the classifiers.

Tags

On the Addito website, the content can be filtered on school year and/or subject. There are 14 predefined tags or labels for school year and 15 predefined labels for subject, as shown in Tables 3.2 and 3.1. When a teacher enters a new learning resource into Addito, they will be prompted to choose one or more of each of these. However, they can also select not to include either. In addition, they can write additional tags as they choose. Figure 3.2 shows the number of tags entered per learning resource, and it can be seen that most of the resources have between 3 and 9 tags.

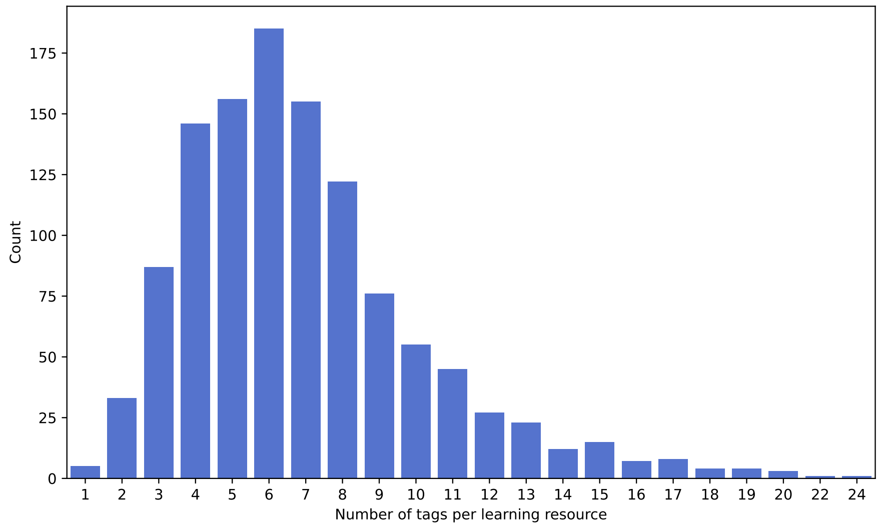


Figure 3.2: Number of tags per learning resource

Table 3.1 gives an overview of all the labels related to subject, and the count of each of them in the dataset, presented in descending order by count. The subject names are in Norwegian, so an English translation is provided. There is a significant over-representation of "Norsk" in the dataset, and some labels have very few entries, making the classes imbalanced. This can make predictions more difficult.

Each learning resource may have been tagged with zero, one or multiple of these labels. Figure 3.3 shows that most of the records only have only one tag for subject, with some

Table 3.1: Subject labels and value counts in descending order based on count

Label Name	Count	English Equivalent Name
Norsk	452	Norwegian
Samfunnsfag	196	Social Sciences
Matematikk	162	Mathematics
Engelsk	140	English
KRLE	99	Religion and Ethics
Naturfag	62	Natural Sciences
Fremmedspråk	56	Foreign Language
Norsk 2. språk	52	Norwegian as 2. language
Kunst og håndverk	28	Arts and Crafts
Kroppsøving	23	Physical Education
Musikk	12	Music
Mat og helse	9	Food and Health
Arbeidslivsfag	7	Working Life Subjects
Samisk	7	Sami
Utdanningsvalg	2	Education Choice
None	123	

having multiple tags and 123 having no tags related to one of the 15 main subject labels at all.

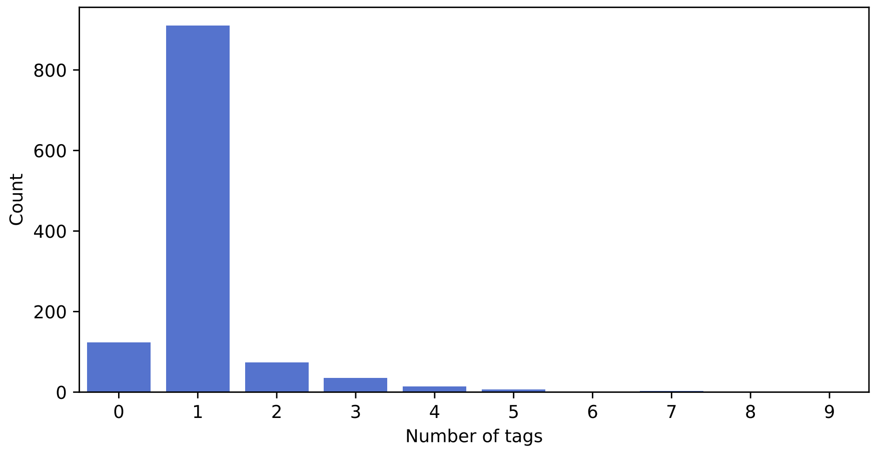


Figure 3.3: Number of tags related to subject with counts showing how many learning resources have zero, one or multiple tags

Table 3.2 gives an overview of all the labels related to school year and the count of each of them in the dataset presented in ascending order by school year.

As for subject, each learning resource may have been tagged with zero, one or multiple of these labels. Figure 3.4 shows the number of tags per learning resource when only including the tags in Table 3.2. This shows that almost 300 of the learning resources have only one tag for school year, however, the vast majority of the entries are tagged with more than one school year, with the most prevalent being 3. A few are even tagged with

Table 3.2: School year labels and value counts in ascending order based on school year

Label Name	Count	School Year
1. trinn	170	1
2. trinn	153	2
3. trinn	165	3
4. trinn	177	4
5. trinn	186	5
6. trinn	183	6
7. trinn	205	7
8. trinn	475	8
9. trinn	476	9
10. trinn	528	10
VG1	342	11
VG2	310	12
VG3	300	13
Voksenopplæring	31	Adult Education
None	19	

all 13 school years. A large number of tags per learning resource may make predictions more difficult.

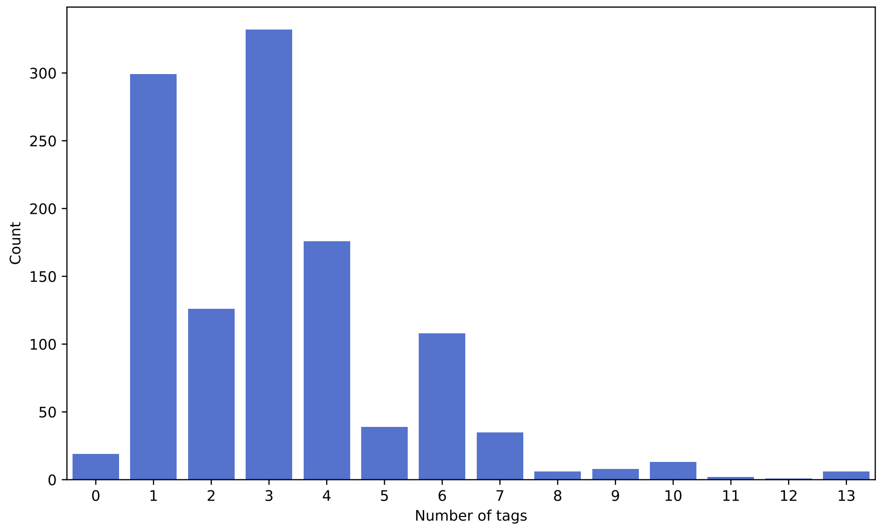


Figure 3.4: Number of tags related to school year with counts showing how many learning resources have multiple tags

In addition to the tags for school year and subject, the teachers can optionally add more tags. In total 771 unique tags have been added to the tags list with 115 of those occurring more than 5 times. 426 of the learning resources have been tagged with at least one of these tags while 744 have none of these tags.

Title and description

The metadata for the learning content includes a field for a title and a description. With only removing punctuation and white spaces, the mean title length is 4.3 words, and the mean description length is 37.9 words. All learning resources have a title of at least one word, while 50 resources have no description at all.

Table 3.3: Statistics title and description

	Median	Mean	Std	Shortest	Longest
Title	4.0	4.3	2.5	1	18
Description	27.0	37.9	36.4	0	266
Title + description	31.0	42.2	36.9	1	271

Some statistics related to the title and description and the combined title and description is shown in Table 3.3 with the number of resources with titles and descriptions of each length is visualised in Figures 3.5 and 3.6.

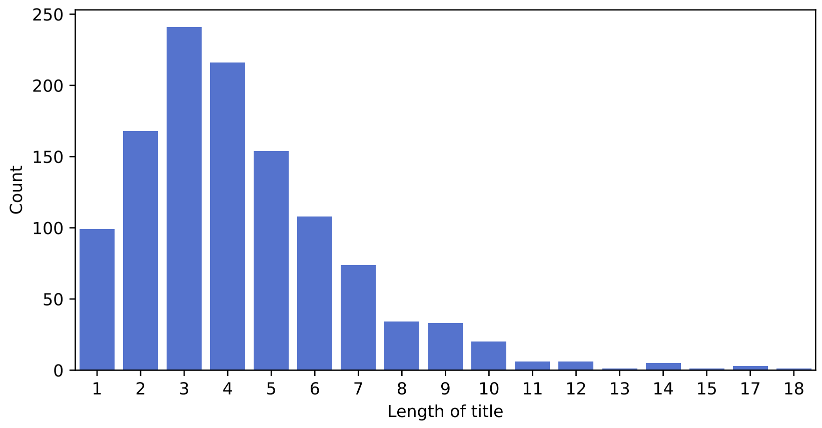


Figure 3.5: Count of resources with number of words in title

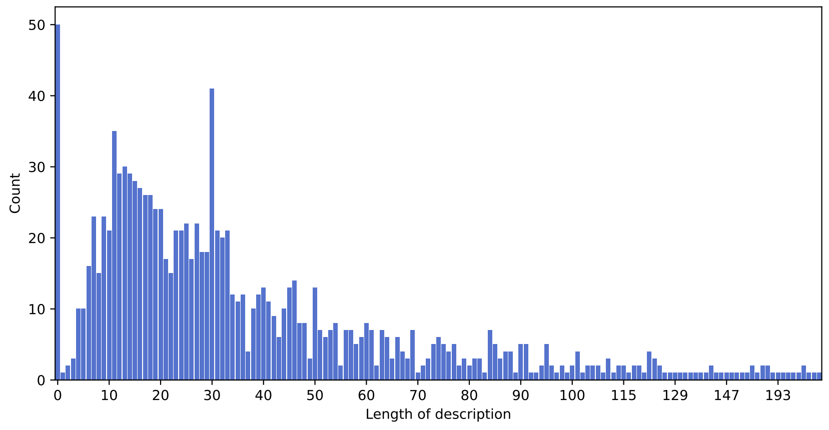


Figure 3.6: Count of resources with number of words in description

Files

The metadata includes filenames and links to files related to the learning resources. Most of the learning resources have at least one file linked to it, with only 90 resources without any files. 179 entries have more than one file linked to the resource. Table 3.7 shows an overview of the filetypes that have been used and it shows that the most commonly used filetypes are docx with 418 files, pptx with 391 files and pdf with 200 files. These filetypes are mostly text based formats, however some may have images that can not be used in text classification.

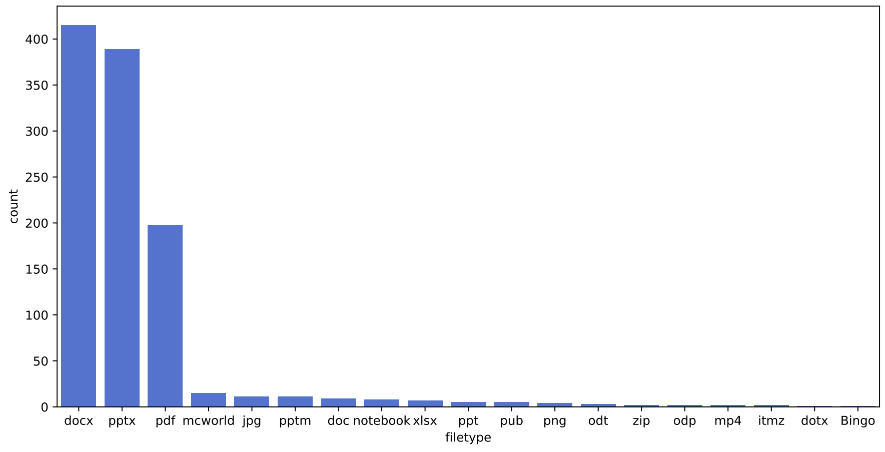


Figure 3.7: Number of each filetype

3.2 Data Preprocessing and Solution Approach

The proposed solution uses a combination of metadata, such as description, title, and tags, and file content as input to tag learning resources with the most suitable subject or school year. As each learning resource can have multiple labels related to each category that will be predicted, the problem will be solved as a multi-label classification problem. The metadata undergoes multiple preprocessing steps to be suitable for the classification. First, the labels used for prediction are separated from the remaining tags and converted into binary label vectors. Then the input text for the classifiers is combined from different parts of the metadata such as title and description and put through preprocessing steps that include removal of symbols and punctuation, lower casing of the text, and removal of stopwords, depending on the model used. Tokenization is then performed to convert the text into machine-readable language. The dataset is then split into training, validation, and test sets, ready to be fed into the neural network models that will be presented in Section 3.3.

3.2.1 Multi-label classification

It is proposed to use multi-label classification for the classification problem as each entry in Addito may have more than one label, either belonging to more than one school year or subject. Multi-label classification involves predicting zero, one, or multiple class labels. Unlike regular classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or labels.

Neural networks were chosen as the preferred method to solve this problem as they natively support multi-label classification. This is done by specifying the number of target labels as the number of nodes in the output layer. A sigmoid activation function is then used in the output layer to predict a probability of belonging to each class between 0 and 1. This is combined with a binary cross-entropy loss function, like in binary classification, because a binary classification is performed in each output node. The advantage of a neural network is that it can take into account that the labels may be correlated due to their shared weights. This is not the case in simpler machine learning methods such as one-vs-rest classification [13].

3.2.2 Selection of labels for prediction

The main focus of this thesis is to predict the subjects and the school year in two separate models. These labels are available for filtering on the Addito website and are shown in Table 3.1 and 3.2.

Subject

Table 3.1 shows that some subject labels have been used on very few entries in the dataset. Therefore, it was decided to remove the three labels with the lowest count: "Utdanningsvalg", "Samisk", and "Arbeidslivsfag". These were selected as it was difficult to separate these into train, test, and validation sets due to a low number of entries, and these are also smaller, less general courses. In addition, "Samisk" is only chosen by a small number of students. It was also decided to merge "Norsk" and "Norsk 2. språk" into one category as these will have considerable overlap given that it is a course in the same language. The updated categories and counts can be found in Tables 3.4. There are now 123 entries with no labels from any of the remaining 11 classes.

Table 3.4: Subject labels and value counts in descending order based on count with updated labels

Label Name	Count	English Equivalent Name
Norsk	504	Norwegian
Samfunnsfag	196	Social Sciences
Matematikk	162	Mathematics
Engelsk	140	English
KRLE	99	Religion and Ethics
Naturfag	62	Natural Sciences
Fremmedspråk	56	Foreign Language
Kunst og håndverk	28	Arts and Crafts
Kroppsøving	23	Physical Education
Musikk	12	Music
Mat og helse	9	Food and Health

School Year

Table 3.2 gives an overview of the labels related to the school year. Overall the classes are better balanced than for subject labels, with "voksenopplæring" being an exception. Therefore, "voksenopplæring" was removed as a class label due to its low number of entries and that this category could encompass course information from several different school years. This could make it more difficult to predict.

3.2.3 Binary Label Vectors

For the computer to understand the labels, binary vectors were created. The length of the vectors was the same as the number of labels to predict, and they were filled with 0 if not true, and 1 if true and were created using scikit-learn's *MultiLabelBinerizer*.

3.2.4 Input text

Several metadata combinations have been used to create the input text for classification. The simplest one is using the title and description combined as one text. Then other tags, which are tags not related to school year or subject, were added to the title and description. And lastly, text extracted from files was also added. The three main input text combinations are shown below, and they were separated with space.

- Title + description
- Title + description + other tags
- Title + description + other tags + data extracted from text files in format .pdf, .docx and .pptx

3.2.5 Text Preprocessing

Depending on the algorithm used, text preprocessing has been performed on the data to optimize the results. The chosen models were based on LSTM, 1D-CNN, and BERT networks which will be presented in detail in Section 3.3. LSTM and CNN are both trained from scratch with and without word embeddings, see Section 2.2.1, while the BERT model has been pre-trained on a Norwegian corpus of raw data. The BERT model can therefore use raw data as input, and the preprocessing steps are therefore more limited than for the LSTM and CNN models.

The preprocessing steps for LSTM and CNN include stopword removal, lemmatization, punctuation removal, and lower casing. Stopword removal has been performed using a built-in library from NLTK. NLTK includes a package with a stopwords list for the Norwegian language, which includes a combination of stopwords in "Bokmål" and "Nynorsk", the two Norwegian variants. This can lead to non-ideal situations where a word is a stopword in one of the variants and an actual word with meaning in the other variant. An example is the word "dykk", which means "you" plural in Nynorsk, which is usually a stopword. The same word can mean "dive" in the noun form in Bokmål. However, this is a minor issue that only applies to a handful of words and is therefore not seen as a problem in this thesis.

Lemmatization is performed using SpaCy, which includes a lemmatizer trained on Norwegian Bokmål [25]. Three trained pipelines were offered, and the largest one, *nb_core_news_lg*, was used as the lemmatization dictionary. This was thought to be more accurate than the Norwegian stemmer offered by NLTK as this was the Snowball stemmer, an improved version of the Porter stemmer, the simplest type of stemmer. This stemmer can sometimes lead to problems with over-stemming and under-stemming, which is less of an issue when using a lemmatizer.

Punctuation, symbols, and extra white spaces were then removed, and all text was lower cased to avoid words with and without capital letters not being identified as the same words. The tokenizer could also do this.

3.2.6 Tokenization and Padding

Machines do not understand text, so it is necessary to convert the text into numbers for the machine to be able to read. To do this, tokenization is performed. Tokenization of a text is splitting it into words or subwords, also called tokens, which are then converted to ids through a look-up table of vocabulary. Keras handles this for the LSTM and CNN models, while BERT has its own tokenizer.

Keras

The Tokenizer class of Keras is used for vectorizing a text corpus. It includes methods to update the internal vocabulary with words from the input text and create sequences of integers to represent each word in the vocabulary. By default, the tokenizer splits words by space, filters out punctuation, and converts text to lower case and these settings were used in this thesis. A maximum vocabulary size can be chosen when building the vocabulary. This was only used when also including text from files. All the input words were used when only using description, title, and other tags as input, as the number of tokens created was relatively small.

Padding is performed after tokenization to make all input vectors the same length as the longest vector by adding 0's at the start of the vectors. When text from files was added to the input, truncation was also performed, where values were removed at the end of the sequence to make all vectors the same maximum length.

BERT

The NorBERT models are trained from scratch for Norwegian on the corpora described in Section 2.4.3 and can be used in the same way as any other BERT model. NorBERT features a custom WordPiece vocabulary that is case-sensitive and includes accented characters [33]. NorBERT 1 features a custom 30 000 WordPiece vocabulary, while NorBERT 2 features a 50 000 WordPiece vocabulary. WordPiece is a subword tokenization algorithm used for BERT that first initializes the vocabulary to include every character present in the training data. It then progressively learns a given number of merge rules. It chooses the symbol pairs that maximize the likelihood of the training data once added to the vocabulary meaning that it merges if the probability of the merged symbols is larger than the probability of the symbols on their own or in another configuration in the given order of the text. In short, the algorithm evaluates what it loses by merging two symbols to ensure that it is worth it [34].

NorBERT vocabularies have much better coverage of Norwegian words than the multilingual BERT (mBERT) models from Google [7]. As seen in Table 3.5, the mBERT tokenizer divides the sentence into smaller parts that will not make sense as standalone words in the Norwegian language. NorBERT 1 and NorBERT 2 perform similarly on the given sentence and much better than mBERT.

The vocabulary for the NorBERT models was generated from raw text without, e.g., separating punctuation from word tokens. This means that one can feed raw text into NorBERT [33].

Table 3.5: Example of a tokenized sentence using mBERT and NorBERT [7].

Vocabulary	Example of a tokenized sentence
Original	Denne gjengen håper at de sammen skal bidra til å gi kvinnefotballen i Kristiansand et lenge etterlengtet løft .
mBERT	Denne g ##jeng ##en h ##å ##per at de sammen skal bid ##ra til å gi k ##vinne ##fo ##t ##ball ##en i Kristiansand et lenge etter ##len ##gte ##t l ##ø ##ft .
NorBERT 1	Denne gjengen håper at de sammen skal bidra til å gi kvinne ##fotball ##en i Kristiansand et lenge etterl ##engt ##et løft .
NorBERT 2	Denne gjengen håper at de sammen skal bidra til å gi kvinne ##fotball ##en i Kristiansand et lenge etterleng ##tet løft .

Padding and truncation are included in the BERT tokenizer to make all the vectors the same length as a specified maximum length.

3.2.7 Word Embeddings

This thesis uses FastText word embeddings created by LTG [12] at the University of Oslo for the LSTM and CNN models. The algorithm used is FastText skipgram with lemmatization trained on a corpus consisting of *Norsk Aviskorkus*, *NoWaC*, and *NBDigital* with a vocabulary size of 4.428.648 [35].

The NorBERT models have their own word embeddings trained on the corpora outlined in Section 2.4.3.

3.2.8 Dataset splitting

The dataset was split into separate sets for training, testing, and validation. The goal was an approximate split of 70% training, 15% validation, and 15% testing. The training and validation sets were used during training, while the test set was used after training to assess the models. The same sets were used for all the models to ensure that differences in splits would not affect the results.

Stratified sampling was used to split the dataset. The goal was to have an equal split of all the labels into set based on the split percentage. This was to ensure that all labels were present in the training, validation, and test sets. As the data was quite imbalanced, especially for the subject labels, exactly getting a 70/15/15 % split was not possible. Therefore, the final splits are slightly different from the desired split percentage.

The dataset was split individually for the subject label problem and school year label problem, as the stratified split was based on the labels that would be predicted. The dataset was split using the *MultilabelStratifiedKFold* algorithm from the *iterative-stratification* package. It was based on the 2011 paper: On the Stratification of Multi-Label Data [36].

Subject

Table 3.6 shows the split percentage and number of records for the training, validation, and test sets used for the subject label classification problem.

Table 3.6: Dataset split for subject tag prediction

Split	No. of records	Percentage
Train set	724	69.4%
Validation set	149	14.3%
Test set	170	16.3%
Total	1043	100%

Table 3.7 gives the number of learning resources of each label in the train, validation, and test sets, along with the percentage. As the dataset was quite imbalanced regarding the number of resources with the different labels, it can be seen that some of the labels have very few entries in the validation and test sets. This can make classification very difficult for these labels.

Table 3.7: Dataset split for subject tag prediction per label with percentage given in brackets

	Labels per split		
Label Name	Train	Validation	Test
Norsk	343 (69.6)	68 (13.8)	82 (16.6)
Samfunnsfag	136 (69.4)	27 (13.8)	33 (16.8)
Matematikk	113 (69.8)	22 (13.6)	27 (16.7)
Engelsk	97 (69.3)	19 (13.6)	24 (17.1)
KRLE	68 (68.7)	14 (14.1)	17 (17.2)
Naturfag	43 (69.4)	8 (12.9)	11 (17.7)
Fremmedspråk	38 (67.9)	8 (14.3)	10 (17.9)
Kunst og håndverk	19 (67.9)	4 (14.3)	5 (17.9)
Kroppsøving	16 (69.6)	3 (13.0)	4 (17.4)
Musikk	8 (66.7)	2 (16.7)	2 (16.7)
Mat og helse	6 (66.7)	1 (11.1)	2 (22.2)

School year

Table 3.8 shows the split percentage and number of records for the training, validation, and test sets used for the school year classification problem, and Table 3.9 gives the split per label.

Table 3.8: Dataset split for school year label prediction

Split	No. of records	Percentage
Train set	772	68.9%
Validation set	160	14.3%
Test set	188	16.8%
Total	1120	100%

Table 3.9: Dataset split for school year prediction with count per label and percentage in brackets

	Labels per split		
Label Name	Train Set	Validation Set	Test Set
1. trinn	117 (68.8)	24 (14.1)	29 (17.1)
2. trinn	107 (69.9)	21 (13.7)	25 (16.3)
3. trinn	114 (69.1)	23 (13.9)	28 (17.0)
4. trinn	122 (68.9)	25 (14.1)	30 (16.9)
5. trinn	129 (69.4)	26 (14.0)	31 (16.7)
6. trinn	127 (69.4)	25 (13.7)	31 (16.9)
7. trinn	142 (69.3)	29 (14.1)	34 (16.6)
8. trinn	331 (69.7)	65 (13.7)	79 (16.6)
9. trinn	331 (69.5)	66 (13.9)	79 (16.6)
10. trinn	366 (69.3)	72 (13.6)	90 (17.0)
VG1	238 (69.6)	47 (13.7)	57 (16.7)
VG2	215 (69.4)	43 (13.9)	52 (16.8)
VG3	209 (69.7)	41 (13.7)	50 (16.7)

3.3 Models for Classification of Learning Content

Several different neural networks have been tested for the classification task: one LSTM, one CNN, and two BERT models pre-trained on different corpora. In addition, the LSTM and CNN models have been tested with and without using pre-trained word embeddings, presented in Section 2.2.1. All of the models have been implemented in Keras.

As this is a multi-label classification problem, all models use the sigmoid activation function and binary cross-entropy loss function in each neuron of the output layer. Adaptive Moment (Adam) is used as the optimizer in all the models. The maximum number of epochs are set to 200 for LSTM and CNN and 50 for BERT, however, the training is set to stop early with an Early Stopping callback with a patience of 10 epochs for CNN and LSTM and 5 epochs for BERT. The best models are saved on best validation loss to avoid overfitting on the training dataset.

All the models feature a dropout layer. This is because when all features are connected in a fully connected layer it can cause overfitting on the training dataset [3]. In the dropout layer, neurons are ignored during training at random. The dropout parameter is a value between 0 and 1 and denotes the fraction of the input neurons to drop.

3.3.1 LSTM

The initial proposed model uses LSTM (from the Keras Sequential model) because LSTMs are efficient at remembering long sequences and modeling long-distance relationships, which is preferable when using sequential textual data such as the learning resource description and title as input. The tested network is illustrated in Figure 3.8. It starts with an input layer which consists of a padded sequence of 222 neurons for the subject classification using description and title as input text. This is the longest input sequence found for this model and is different when also using other tags and files and for the school year classification. The input layer is followed by an embedding layer, a spatial dropout layer, a 100 unit LSTM layer, and a fully connected output layer with 11 neurons for subject classification and 13 neurons for school year classification. If FastText word embeddings are used, these are added as an embedding matrix in the embedding layer.

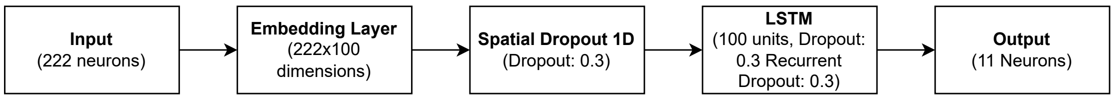


Figure 3.8: Illustration of LSTM structure for subject classification using title and description

3.3.2 CNN

Another proposed solution uses CNNs as these recognize local patterns in a sequence by processing multiple words simultaneously, and 1D CNNs are suitable for text processing tasks. An illustration of the CNN structure is shown in Figure 3.9 for subject classification using title and description as input. As for LSTM, the first layer is an input layer followed by an embedding layer which includes the embedding matrix if this is used. Following this, a 1D convolutional layer is added with the chosen kernel size, filter size, and stride. Then follows a max-pooling layer and a dropout layer, ending in a fully connected output layer with 11 neurons for subject and 13 for school year.

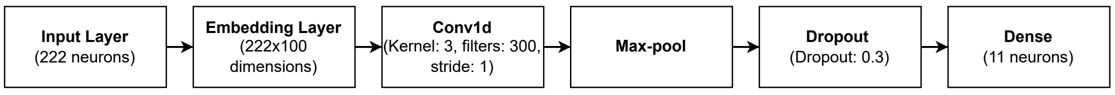


Figure 3.9: Illustration of CNN structure for subject classification using title and description

3.3.3 BERT

The final proposed model uses pre-trained BERT models. These are the NorBERT models found on Hugging Face, which has been pre-trained on Norwegian datasets described in 2.4.3. These were then fine-tuned on the Addito dataset. Only the final layer of the BERT models is trained during fine-tuning, and Figure 3.10 illustrates the structure. The NorBERT models correspond in their configuration to Google's Bert-Base Cased for English with 12 layers and hidden size 768 [33]. The input to the model's main layer is split into input ids and attention mask. This is followed by a dropout layer with a dropout defined from the configuration file accompanying the models. The last layer is a fully connected output layer with 11 neurons for subject and 13 for school year.

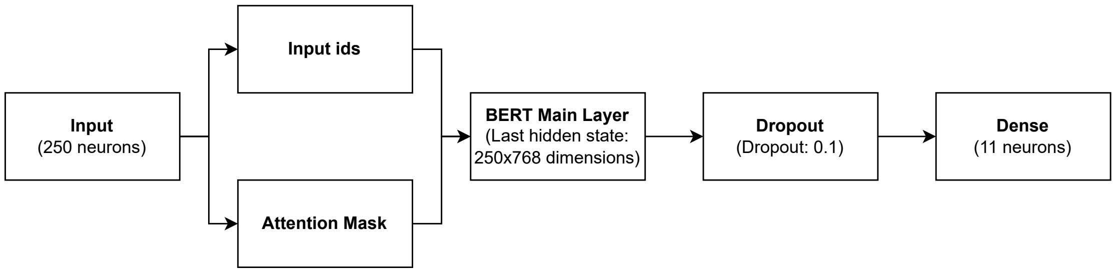


Figure 3.10: Illustration of BERT structure for subject classification using title and description

3.3.4 Hyperparameter tuning

To find the best hyperparameters to be used with the different algorithms, hyperparameter tuning was performed. This was done using the algorithm Hyperas [37], a wrapper around hyperopt for fast hyperparameter optimization with Keras models.

For LSTM and CNN, the following hyperparameters were tested:

- **Batch size:** 8, 16, 32, 64
- **Learning rate (Adam):** 1e-3, 5e-4, 1e-4, 5e-5
- **Dropout:** 0.1, 0.2, 0.3, 0.4
- **Filter length (CNN):** 100, 200, 300, 400
- **Kernel Size (CNN):** 3, 5, 7

For fine-tuning of BERT, most model hyperparameters are kept the same as in pre-training, except for the batch size, learning rate, and the number of training epochs. According to [5], the optimal hyperparameters for fine-tuning are task-specific; however, they found that the following range works well across all tasks:

- **Batch size:** 16, 32
- **Learning rate (Adam):** 5e-5, 3e-5, 2e-5
- **Number of epochs:** 2, 3, 4

As NorBERT has the same architecture as BERT, these parameters were used as a starting point for finding the optimal hyperparameters for this problem. The learning rates and batch sizes were tested with the combinations above; however, the number of epochs was decided by saving the best model with the lowest validation loss.

Hyperparameter tuning results

The results from the hyperparameter tuning are shown in Table 3.10. The LSTM models with and without word embeddings used the same hyperparameters, while the two CNN models had slightly different batch sizes and filter sizes, and the NorBERT models had different optimal learning rates. This may be actual differences in performance, or the random nature of the models may skew the results if the difference in results using the different hyperparameters are small.

Table 3.10: Hyperparameter tuning results

Model	Learning Rate	Batch Size	Dropout	Filter Size	Kernel Size
LSTM	5e-04	32	0.3		
LSTM + FastText	5e-04	32	0.3		
CNN	1e-03	16	0.3	300	3
CNN + FastText	1e-03	8	0.3	400	3
NorBERT1	2e-05	16			
NorBERT2	5e-05	16			

3.4 Performance measures

Performance measures are metrics used to evaluate the performance of the models. In traditional classification, such as binary or multi-class problems, *accuracy* is the most common evaluation criterion. In addition, there is a set of standard evaluation metrics that include *precision*, *recall*, *F1-score*, and *AUC* (area under receiving operating characteristic (ROC) curve) defined for single-label multi-class classification problems. However, in multi-label classification, predictions for an instance is a set of labels and therefore, the prediction can be *fully correct*, *partially correct* (with different levels of correctness), or *fully incorrect.* None of these existing evaluation metrics capture such a notion in their original form. This makes evaluation of a multi-label classifier more challenging than evaluation of a single label classifier [38].

For a binary classification problem the results can be summarized in a confusion matrix, as shown in Figure 3.11. True positive (TP) is the number of records that are true that have been correctly identified as true. False Positive (FP), is the number of records that are false that have been incorrectly identified as true. False Negative (FN) is the number of records that are actually true that has been incorrectly identified as false. True

Negative (TN) is the number of records that are actually false that has been correctly identified as false.

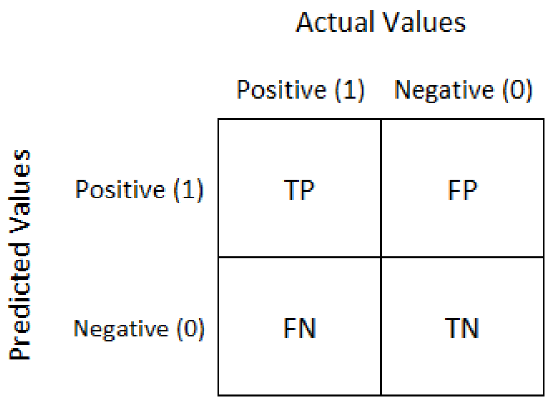


Figure 3.11: Confusion matrix for binary classification

Accuracy is the number of correct predictions divided by the total number of predictions made:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

In its simplest form for a multi-label classification problem, the accuracy score uses exact binary vector matching, meaning all the labels per resource has to be predicted correctly for the prediction to be considered a true positive. This method of accuracy is also called **Exact Match Ratio**. This means that no credit is given if some of the labels are predicted correctly. One way to mitigate this is to separate the predictions by class and calculate the accuracy score per class and take the average, called the **Global Accuracy**. This may give a high accuracy score as there will most likely be a lot of true negative predictions when there are a lot of classes. Even if no predictions are made at all, the accuracy could be high due to the large number of true negative predictions. Another method of calculating the accuracy is called the **Hamming Score** or **Multi-label Accuracy** [39] and this is the average of taking the accuracy for each instance, defined as the proportion of the predicted correct labels to the total number of labels for each instance as summarized in Equation 3.2 [38]:

$$Accuracy\ (Hamming) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i \cap Z_i}{Y_i \cup Z_i}\right| \tag{3.2}$$

where Y_i is the correct label set and Z_i is predicted label set and n is the number of instances.

Precision is the number of correctly predicted positive records divided by all the records predicted as positive:

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Recall is the number of correctly predicted positive records divided by all the records that are actually positive:

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

F1-score is the weighted average of precision and recall:

$$F1 = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} \tag{3.5}$$

Area Under Curve (AUC) is the area under the ROC curve which is a graph showing a classification model at all classification thresholds. This curve plots two parameters: true positive rate (TPR) and false positive rate (FPR), shown in Figure 3.12.

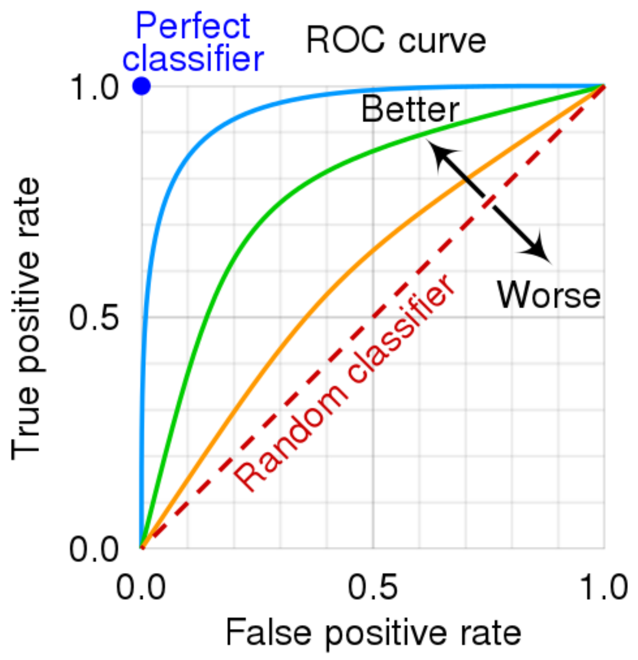


Figure 3.12: Illustration of ROC curve [6]

True positive rate is a synonym for recall as described above and false positive rate is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{3.6}$$

One way of interpreting AUC is as the probability that the model ranks a random positive example higher than a random negative example. AUC ranges in value from 0 to 1 where 1 is a model where all predictions are 100% correct, 0 is a model where predictions are 100% incorrect and 0.5 is a model performing equivalent to a random draw. AUC is classification threshold invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen [40].

In order to be used for the multi-label case, precision, recall, F1-score, and AUC is calculated per class label and the micro average of the classes are taken for the models to find the overall performance. The micro average was chosen due to class imbalance as macro average treat all classes equally, while the micro average will aggregate the contributions of all classes to compute the average metric [38].

Chapter 4

Experimental Results

This chapter presents the results obtained from the approach presented in the previous chapter. The results are split into two parts, one for predicting subject labels and one for predicting school year labels.

4.1 Evaluation metrics

The models have been assessed on the test set using accuracy, precision, recall, F1-score, and AUC. Accuracy is split into three different measures: exact match ratio, hamming score, and global accuracy. Details about how each metric is calculated can be found in Section 3.4.

The models presented in the previous chapter aim to classify the learning resources with the most suitable subject and school year labels. The exact match ratio is the simplest accuracy measure for multi-label classification that considers partially correct predictions as incorrect. This will give a low accuracy if many of the predictions are partially correct. The other extreme is global accuracy which is the proportion of true positive and true negative predictions to all predictions. This will likely give a high accuracy for all models as there will be a lot of true negative predictions due to the large number of classes. Therefore, a third measure of accuracy is included, called the Hamming score, which is the average of the proportion of predicted correct labels to the total number of labels for that instance. This excludes true negative predictions and will likely be a more suitable measure of accuracy in this case.

The other evaluation metrics included are precision, recall, F1-score, and AUC. These are given per class and micro-averaged across the classes. This is to give an overall performance of the models as well as per class performance. The balance between precision and recall is important to evaluate whether the predictions made are relevant for the teacher (high precision) and that they do not miss predictions that could be relevant (high recall). The F1-score is the harmonic mean between these two measures and a high F1-score will often be due to a good balance between these. The accuracy, precision, recall, and F1-score are calculated using binary predictions with a threshold of 0.5. The AUC is calculated using raw probabilities and will give a measure of the performance of the models independent of the chosen threshold. Generally, an AUC above 0.8 is considered good and an AUC above 0.9 is considered excellent.

4.2 Subject Labels

This section presents the results for the prediction of the subject labels. The results have been split into sections depending on the text input used in the models.

4.2.1 Input: Description and title

Table 4.1 summarizes the results for the subject label prediction using only description and title as input to the models. It can be seen that the NorBERT models give the highest scores for all metrics; however, it varies which one gets the highest for each metric.

Table 4.1: Overall results for subject label using description and title as input. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.55	0.58	0.93	0.84	0.50	0.63	0.86
LSTM + FastText	0.68	0.73	**0.95**	0.87	0.63	0.73	0.94
CNN	0.59	0.61	0.94	0.88	0.52	0.65	0.92
CNN + FastText	0.68	0.73	**0.95**	0.86	0.64	0.73	**0.95**
NorBERT1	0.69	0.76	**0.95**	0.83	**0.71**	0.76	0.93
NorBERT2	**0.77**	**0.80**	0.94	**0.93**	**0.71**	**0.80**	0.94

The results per class label can be found in Table 4.2 for LSTM and CNN without pre-trained word embeddings, Table 4.3 for LSTM and CNN with FastText word embeddings, and Table 4.4 for NorBERT 1 and NorBERT 2.

The basic LSTM and CNN models without word embeddings are not able to predict many of the labels of the minority classes. Table 4.2 shows that almost half of the labels got a precision of 1.00 and a recall of 0.00. A precision of 1.00 means that there are no false positive predictions, which means that the models will not predict unless it is sure that it is the correct class. A recall of 0.00 means that there are no true positive

Table 4.2: Results for subject label with LSTM and CNN using description and title

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.87	0.80	0.84	0.87	0.86	0.85	0.86	0.94	82
Samfunnsfag	0.82	0.55	0.65	0.87	0.83	0.30	0.44	0.91	33
Matematikk	0.80	0.44	0.57	0.86	1.00	0.59	0.74	0.94	27
Engelsk	0.80	0.50	0.62	0.79	0.86	0.50	0.63	0.83	24
KRLE	0.50	0.06	0.11	0.70	1.00	0.18	0.30	0.81	17
Naturfag	1.00	0.00	0.00	0.75	1.00	0.00	0.00	0.70	11
Fremmedspråk	1.00	0.00	0.00	0.86	1.00	0.10	0.18	0.91	10
Kunst og håndverk	1.00	0.00	0.00	0.70	1.00	0.00	0.00	0.94	5
Kroppsøving	1.00	0.00	0.00	0.65	1.00	0.00	0.00	0.79	4
Musikk	1.00	0.00	0.00	0.87	1.00	0.00	0.00	0.22	2
Mat og helse	1.00	0.00	0.00	0.65	1.00	0.00	0.00	0.51	2
Micro Avg	0.84	0.50	0.63	0.86	0.84	0.52	0.64	0.92	217

predictions either, which means that the model does not predict these classes at all. The AUC scores for some of the minority classes are close to 0.5, which means they are not much better than random chance. For "Musikk", CNN even gives an AUC score of 0.22, which means it mainly predicts the incorrect label.

Table 4.3: Results for subject label with LSTM and CNN with FastText word embeddings using description and title

Label	LSTM + FastText				CNN + FastText				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.86	0.85	0.86	0.93	0.88	0.84	0.94	0.89	82
Samfunnsfag	0.88	0.64	0.74	0.90	0.81	0.52	0.63	0.93	33
Matematikk	0.90	0.70	0.79	0.97	0.94	0.59	0.73	0.99	27
Engelsk	0.83	0.62	0.71	0.91	0.82	0.58	0.68	0.88	24
KRLE	0.80	0.24	0.36	0.90	1.00	0.35	0.52	0.88	17
Naturfag	1.00	0.18	0.31	0.89	1.00	0.27	0.43	0.93	11
Fremmedspråk	1.00	0.50	0.67	0.85	1.00	0.50	0.67	0.92	10
Kunst og håndverk	1.00	0.00	0.00	0.90	1.00	0.00	0.00	0.94	5
Kroppsøving	1.00	0.25	0.40	0.94	1.00	0.00	0.00	0.84	4
Musikk	1.00	0.00	0.00	0.86	1.00	0.00	0.00	0.45	2
Mat og helse	1.00	0.00	0.00	0.96	1.00	0.00	0.00	0.81	2
Micro Avg	0.87	0.63	0.73	0.94	0.86	0.64	0.73	0.95	217

Table 4.3 shows that the models performs better using FastText word embeddings. Fewer classes have no predictions at all for both LSTM and CNN, and the overall recall increases, which means that there are fewer false negative predictions. However, there are still three classes for LSTM and four classes for CNN where no predictions are made and a few classes with very low recall. The AUC for most classes are above 0.8 for both LSTM and CNN, except for "Musikk" using CNN, which is 0.45.

The NorBERT models performs better than LSTM and CNN; however, Table 4.4 shows that they both have a few classes where no predictions are made. Both models have an overall higher recall than LSTM and CNN, which means that they predict more labels than the other models and have fewer false negative predictions. This also means more false positive predictions are made for NorBERT 1, giving this a lower precision.

Table 4.4: Results for subject label with NorBERT 1 and NorBERT 2 using description and title

Label	NorBERT 1				NorBERT 2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.81	0.89	0.85	0.95	0.94	0.90	0.92	0.97	82
Samfunnsfag	0.80	0.61	0.76	0.87	0.85	0.67	0.75	0.90	33
Matematikk	0.88	0.81	0.85	0.95	1.00	0.74	0.85	0.99	27
Engelsk	0.82	0.58	0.68	0.87	0.88	0.62	0.73	0.88	24
KRLE	0.90	0.53	0.67	0.85	1.00	0.47	0.64	0.79	17
Naturfag	0.71	0.45	0.56	0.93	0.83	0.45	0.59	0.89	11
Fremmedspråk	1.00	0.80	0.89	0.99	1.00	0.80	0.89	0.94	10
Kunst og håndverk	0.50	0.20	0.29	0.90	1.00	0.00	0.00	0.94	5
Kroppsøving	1.00	0.25	0.40	0.88	1.00	0.25	0.40	0.87	4
Musikk	1.00	0.00	0.00	0.85	1.00	0.00	0.00	0.68	2
Mat og helse	1.00	0.00	0.00	0.87	1.00	0.00	0.00	0.85	2
Micro Avg	0.83	0.71	0.76	0.93	0.93	0.71	0.80	0.94	217

4.2.2 Input: Description, title and other tags

Table 4.5 summarizes the results for the subject label prediction using description, title, and other tags as input to the models. Compared to only using description and title in Table 4.1, all the metrics for all the models are higher when using other tags in addition.

Table 4.5: Overall results for subject label using description, title, and other tags as input. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.77	0.79	0.96	**0.97**	0.67	0.79	0.94
LSTM + Fasttext	0.77	0.82	0.96	0.89	0.77	0.83	**0.97**
CNN	0.82	0.86	**0.97**	**0.97**	0.73	0.83	0.96
CNN + Fasttext	0.81	0.84	**0.97**	**0.97**	0.74	0.84	**0.97**
NorBERT1	0.82	0.86	0.96	0.89	0.78	0.83	0.96
NorBERT2	**0.84**	**0.88**	**0.97**	0.96	**0.79**	**0.86**	**0.97**

The results per class label can be found in Table 4.6 for LSTM and CNN without pre-trained word embeddings, Table 4.7 for LSTM and CNN with fasttext word embeddings, and Table 4.8 for NorBERT 1 and NorBERT 2.

When comparing the results for CNN and LSTM without word embeddings in Table 4.6 to the results using only description and title in Table 4.2, it can be seen that there are fewer classes that the models cannot predict. The overall recall has increased from 0.50 to 0.67 for LSTM and from 0.52 to 0.73 for CNN. This means fewer false negative predictions. The average precision is high at 0.97 for both models. However, this is in part due to a precision of 1.00 being given to some classes even though no predictions are made. Despite these classes not having any predictions, the AUC was higher, which means that the model could have performed better at another threshold for these classes.

Table 4.6: Results for subject label with CNN and LSTM using description, title and other tags

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.95	0.85	0.90	0.94	0.96	0.95	0.96	0.99	82
Samfunnsfag	1.00	0.64	0.78	0.92	1.00	0.61	0.75	0.95	33
Matematikk	1.00	0.78	0.88	0.95	1.00	0.78	0.88	0.99	27
Engelsk	0.94	0.62	0.75	0.97	0.90	0.75	0.82	0.92	24
KRLE	1.00	0.59	0.74	0.96	1.00	0.65	0.79	0.94	17
Naturfag	1.00	0.18	0.31	0.84	1.00	0.27	0.43	0.89	11
Fremmedspråk	1.00	0.60	0.75	0.97	1.00	0.70	0.82	0.99	10
Kunst og håndverk	1.00	0.00	0.00	0.72	1.00	0.00	0.00	0.89	5
Kroppsøving	1.00	0.00	0.00	0.90	1.00	0.25	0.40	0.91	4
Musikk	1.00	0.00	0.00	0.91	1.00	0.00	0.00	0.57	2
Mat og helse	1.00	0.00	0.00	0.77	1.00	0.00	0.00	0.76	2
Micro Avg	0.97	0.67	0.79	0.94	0.97	0.73	0.83	0.96	217

Table 4.7: Results for subject label with LSTM and CNN with FastText word embeddings using description, title, and other tags

Label	LSTM + Fasttext				CNN + Fasttext				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.90	0.93	0.92	0.99	0.96	0.89	0.92	0.98	82
Samfunnsfag	0.78	0.76	0.77	0.97	0.96	0.67	0.79	0.98	33
Matematikk	0.91	0.78	0.84	0.99	1.00	0.85	0.92	0.99	27
Engelsk	0.94	0.67	0.78	0.92	0.93	0.54	0.68	0.94	24
KRLE	0.86	0.71	0.77	0.95	1.00	0.65	0.79	0.94	17
Naturfag	1.00	0.55	0.71	0.94	1.00	0.55	0.71	0.96	11
Fremmedspråk	1.00	0.80	0.89	0.98	1.00	0.80	0.89	0.98	10
Kunst og håndverk	1.00	0.00	0.00	0.91	1.00	0.20	0.33	0.89	5
Kroppsøving	1.00	0.75	0.86	1.00	1.00	0.75	0.86	0.97	4
Musikk	1.00	0.00	0.00	0.72	1.00	0.00	0.00	0.52	2
Mat og helse	1.00	0.00	0.00	0.96	1.00	0.00	0.00	0.79	2
Micro Avg	0.89	0.77	0.83	0.97	0.97	0.74	0.84	0.97	217

Table 4.8: Results for subject label with NorBERT 1 and NorBERT 2 using description, title, and other tags

Label	NorBERT1				NorBERT2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.94	0.93	0.93	0.98	0.95	0.94	0.94	0.98	82
Samfunnsfag	0.91	0.64	0.75	0.94	0.92	0.70	0.79	0.98	33
Matematikk	0.82	0.85	0.84	0.99	1.00	0.89	0.94	0.99	27
Engelsk	0.87	0.83	0.85	0.95	0.95	0.75	0.84	0.95	24
KRLE	1.00	0.71	0.83	0.93	1.00	0.65	0.79	0.90	17
Naturfag	0.83	0.45	0.59	0.80	0.83	0.45	0.59	0.98	11
Fremmedspråk	0.69	0.90	0.78	0.96	1.00	0.90	0.95	0.99	10
Kunst og håndverk	1.00	0.00	0.00	0.94	1.00	0.20	0.33	0.86	5
Kroppsøving	1.00	0.75	0.86	1.00	1.00	0.75	0.86	1.00	4
Musikk	1.00	0.00	0.00	0.78	1.00	0.00	0.00	0.46	2
Mat og helse	1.00	0.00	0.00	0.49	1.00	0.00	0.00	0.97	2
Micro Avg	0.89	0.78	0.83	0.96	0.96	0.79	0.86	0.97	217

Table 4.7 shows that using word embeddings increases the recall for LSTM from 0.67 to 0.77, and the model is able to predict the class "Kroppsøving", which had no predictions without word embeddings. For CNN, the recall only increases by 0.01; however, with word embeddings, only two classes have no predictions, "Musikk" and "Mat og Helse",

which none of the models have been able to predict. For CNN, the AUC for "Musikk" is only 0.52 which means that at any threshold, the model is still not able to predict this class better than chance.

Table 4.8 shows that using the NorBERT models gives similar results as using LSTM and CNN with FastText word embeddings; however, they both give a better recall.

4.2.3 Input: Description, title, other tags and file text

Table 4.9 summarizes the results for the subject label prediction using description, title, other tags, and text from files as input to the models. Compared to only using description and title in Table 4.1 and description, title, and other tags in Table 4.5, almost all the metrics are lower when also using input from files. For CNN and LSTM, the maximum vocabulary present in the files is used as input to the tokenizer, and all models uses a maximum length of 500 for the input vector.

Table 4.9: Overall results for subject label using description, title, other tags and file text as input. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.63	0.67	0.94	0.81	0.57	0.67	0.91
LSTM + Fasttext	0.74	0.78	0.95	0.88	0.66	0.76	0.95
CNN	0.79	0.82	0.96	**0.96**	0.70	0.81	**0.98**
CNN + Fasttext	0.81	0.85	**0.97**	**0.96**	0.72	0.83	0.97
NorBERT1	0.79	0.86	**0.97**	0.93	0.75	0.83	0.95
NorBERT2	**0.84**	**0.89**	**0.97**	0.95	**0.78**	**0.86**	0.96

The results per class can be found in Table 4.10 for LSTM and CNN without pre-trained word embeddings, Table 4.11 for LSTM and CNN with FastText word embeddings, and Table 4.12 for NorBERT 1 and NorBERT 2.

Table 4.10: Results for subject label with LSTM and CNN using description and title, other tags and file text

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.86	0.76	0.81	0.90	0.94	0.94	0.94	0.99	82
Samfunnsfag	0.53	0.52	0.52	0.86	1.00	0.58	0.73	0.97	33
Matematikk	0.95	0.78	0.86	0.96	1.00	0.78	0.88	0.98	27
Engelsk	0.94	0.62	0.75	0.93	0.95	0.75	0.84	0.95	24
KRLE	0.89	0.47	0.62	0.88	1.00	0.59	0.74	0.98	17
Naturfag	1.00	0.00	0.00	0.74	1.00	0.09	0.17	0.97	11
Fremmedspråk	1.00	0.00	0.00	0.81	1.00	0.40	0.57	1.00	10
Kunst og håndverk	1.00	0.00	0.00	0.65	1.00	0.00	0.00	0.90	5
Kroppsøving	1.00	0.00	0.00	0.59	1.00	0.25	0.40	1.00	4
Musikk	1.00	0.00	0.00	0.65	1.00	0.00	0.00	0.57	2
Mat og helse	1.00	0.00	0.00	0.87	1.00	0.00	0.00	0.86	2
Micro Avg	0.81	0.57	0.67	0.91	0.96	0.70	0.81	0.98	217

Table 4.10 shows that the average results for both LSTM and CNN are lower when additionally using text from files as input to the models. Both models are much worse at predicting almost all classes, and especially the minority classes, than when using description, title, and other tags, seen in Table 4.6. LSTM gives no predictions on more than half of the classes. The performance of CNN is only slightly decreased from the previous section.

Table 4.11: Results for subject label with LSTM and CNN with word embeddings using description and title, other tags and file text

Label	LSTM + FastText				CNN + FastText				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.92	0.82	0.86	0.96	0.96	0.93	0.94	0.98	82
Samfunnsfag	0.76	0.48	0.59	0.88	0.95	0.55	0.69	0.97	33
Matematikk	1.00	0.81	0.90	0.99	1.00	0.81	0.90	0.99	27
Engelsk	0.85	0.71	0.77	0.94	0.89	0.67	0.76	0.97	24
KRLE	0.73	0.65	0.69	0.87	1.00	0.71	0.83	0.93	17
Naturfag	0.67	0.18	0.29	0.93	1.00	0.18	0.31	0.95	11
Fremmedspråk	1.00	0.60	0.75	0.98	1.00	0.90	0.95	1.00	10
Kunst og håndverk	0.50	0.20	0.29	0.89	1.00	0.00	0.00	0.89	5
Kroppsøving	1.00	0.50	0.67	0.78	1.00	0.50	0.67	0.99	4
Musikk	1.00	0.00	0.00	0.85	1.00	0.00	0.00	0.40	2
Mat og helse	1.00	0.00	0.00	0.96	1.00	0.00	0.00	0.75	2
Micro Avg	0.88	0.66	0.76	0.95	0.96	0.72	0.83	0.97	217

When using word embeddings for LSTM and CNN, the performance improves as can be seen in Table 4.11. The results are still not as good as the results given in Table 4.7, where the file text is not used. LSTM now only has two classes where no predictions are made; however, it has a much lower recall than in the previous section. CNN's results are much closer to the previous section with only a drop of 0.01 in F1-score; however, there is an additional class for which no predictions are made.

Table 4.12: Results for subject label with NorBERT 1 and NorBERT 2 using description and title, other tags and file text

Label	NorBERT 1				NorBERT 2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
Norsk	0.95	0.91	0.93	0.98	0.93	0.96	0.95	0.99	82
Samfunnsfag	0.94	0.48	0.64	0.89	0.95	0.61	0.74	0.98	33
Matematikk	1.00	0.81	0.90	0.99	0.96	0.89	0.92	0.96	27
Engelsk	0.95	0.79	0.86	0.96	1.00	0.67	0.80	0.95	24
KRLE	0.71	0.71	0.71	0.90	1.00	0.71	0.83	0.85	17
Naturfag	0.86	0.55	0.67	0.89	0.80	0.36	0.50	0.90	11
Fremmedspråk	1.00	0.90	0.95	0.99	1.00	0.90	0.95	0.99	10
Kunst og håndverk	1.00	0.20	0.33	0.97	1.00	0.60	0.75	0.98	5
Kroppsøving	1.00	0.75	0.86	1.00	1.00	0.60	0.75	0.94	4
Musikk	1.00	0.00	0.00	0.68	1.00	0.00	0.00	0.81	2
Mat og helse	1.00	0.00	0.00	0.68	1.00	0.00	0.00	0.91	2
Micro Avg	0.93	0.75	0.83	0.95	0.95	0.78	0.86	0.96	217

The results using NorBERT 1 and NorBERT 2 with an input vector length of 500 is shown Table 4.12. The input vectors were double the length than when not using file text, and the running time approximately quadrupled. The results shows that using file text gives almost the same results as not using file text for these models.

4.3 School Year Labels

This section presents the results for the prediction of the school year labels. The results have been split into sections depending on the text input used in the models.

4.3.1 Input: Description and title

Table 4.13 summarizes the results for the school year label prediction using only description and title as input to the models. The results for school year is overall much lower for all models than for subject with the same input, seen in Table 4.1.

Table 4.13: Overall results for school year label using description and title as input. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.15	0.34	0.80	0.63	0.45	0.52	0.81
LSTM + FastText	0.18	0.41	0.79	0.60	0.49	0.54	0.81
CNN	0.20	0.37	0.80	**0.67**	0.40	0.51	0.85
CNN + FastText	**0.22**	0.39	0.80	0.66	0.42	0.52	**0.86**
NorBERT1	0.16	0.39	0.80	0.63	0.46	0.53	0.83
NorBERT2	0.19	**0.43**	**0.81**	**0.67**	**0.51**	**0.58**	0.85

The results per class can be found in Table 4.14 for LSTM and CNN without pre-trained word embeddings, Table 4.15 for LSTM and CNN with FastText word embeddings, and Table 4.16 for NorBERT 1 and NorBERT 2.

Table 4.14: Results for school year label with LSTM and CNN using description and title

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.85	0.38	0.52	0.87	0.93	0.45	0.60	0.91	29
2. trinn	0.79	0.44	0.56	0.85	1.00	0.40	0.57	0.88	25
3. trinn	0.79	0.39	0.52	0.82	1.00	0.32	0.49	0.88	28
4. trinn	0.77	0.33	0.47	0.75	0.80	0.27	0.40	0.84	30
5. trinn	0.41	0.23	0.29	0.81	0.67	0.06	0.12	0.86	31
6. trinn	0.53	0.32	0.40	0.80	0.67	0.06	0.12	0.85	31
7. trinn	0.57	0.35	0.44	0.78	0.50	0.06	0.11	0.81	34
8. trinn	0.60	0.68	0.64	0.73	0.63	0.62	0.62	0.78	79
9. trinn	0.62	0.71	0.66	0.73	0.62	0.61	0.61	0.75	79
10. trinn	0.67	0.73	0.70	0.76	0.67	0.68	0.67	0.76	90
VG1	0.62	0.18	0.27	0.69	0.70	0.28	0.40	0.75	57
VG2	0.69	0.17	0.28	0.73	0.69	0.35	0.46	0.81	52
VG3	0.54	0.14	0.22	0.76	0.52	0.22	0.31	0.75	50
Micro Avg	0.63	0.45	0.52	0.81	0.67	0.40	0.51	0.85	615

Table 4.14 shows that for this problem, there are a relatively large number of records per class compared to for the subject label prediction. This results in no instances where the recall is 0.00 and no predictions have been made. However, several classes have a quite low recall and precision for both LSTM and CNN which gives low overall scores and a lower F1-score. The AUC is also lower than for the subject label predictions; however all classes have an AUC higher than random chance. LSTM and CNN performs very similarly, with LSTM having a better recall and CNN a better precision which means that LSTM has more false positive predictions and CNN more false negative predictions.

Table 4.15: Results for school year label with LSTM and CNN with FastText word embeddings using description and title

Label	LSTM + FastText				CNN + FastText				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.70	0.55	0.62	0.90	1.00	0.45	0.62	0.94	29
2. trinn	0.91	0.40	0.56	0.86	1.00	0.40	0.57	0.94	25
3. trinn	0.73	0.39	0.51	0.84	1.00	0.36	0.53	0.91	28
4. trinn	0.65	0.37	0.47	0.79	0.78	0.23	0.36	0.87	30
5. trinn	0.44	0.23	0.30	0.80	0.62	0.16	0.26	0.86	31
6. trinn	0.59	0.32	0.42	0.80	0.67	0.13	0.22	0.82	31
7. trinn	0.61	0.32	0.42	0.78	0.80	0.24	0.36	0.78	34
8. trinn	0.58	0.72	0.64	0.69	0.57	0.58	0.58	0.74	79
9. trinn	0.57	0.70	0.63	0.68	0.56	0.65	0.60	0.72	79
10. trinn	0.63	0.77	0.69	0.72	0.65	0.64	0.65	0.78	90
VG1	0.53	0.35	0.42	0.70	0.77	0.35	0.48	0.81	57
VG2	0.54	0.27	0.36	0.74	0.81	0.33	0.47	0.79	52
VG3	0.57	0.26	0.36	0.74	0.65	0.22	0.33	0.78	50
Micro Avg	0.60	0.49	0.54	0.81	0.66	0.42	0.52	0.86	615

Using word embeddings for LSTM and CNN increases the overall results for both models, as shown in Table 4.15. However, the increase in performance is much smaller than for the subject label problem. The overall performance is still relatively poor.

Table 4.16: Results for school year label with NorBERT 1 and NorBERT 2 using description and title

Label	NorBERT 1				NorBERT 2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.60	0.72	0.66	0.93	0.86	0.66	0.75	0.96	29
2. trinn	0.67	0.56	0.61	0.92	0.93	0.52	0.67	0.93	25
3. trinn	0.74	0.50	0.60	0.91	0.90	0.64	0.75	0.96	28
4. trinn	0.69	0.60	0.64	0.89	0.78	0.60	0.68	0.91	30
5. trinn	0.50	0.29	0.37	0.82	0.70	0.23	0.34	0.83	31
6. trinn	0.56	0.16	0.25	0.76	0.62	0.16	0.26	0.81	31
7. trinn	0.44	0.12	0.19	0.77	0.75	0.18	0.29	0.79	34
8. trinn	0.65	0.54	0.59	0.75	0.59	0.77	0.67	0.0.78	79
9. trinn	0.59	0.48	0.53	0.71	0.57	0.67	0.62	0.71	79
10. trinn	0.70	0.79	0.74	0.78	0.66	0.82	0.73	0.79	90
VG1	0.57	0.37	0.45	0.76	0.66	0.37	0.47	0.75	57
VG2	0.73	0.31	0.43	0.75	0.80	0.23	0.36	0.78	52
VG3	0.42	0.20	0.27	0.70	0.86	0.12	0.21	0.72	50
Micro Avg	0.63	0.46	0.53	0.83	0.67	0.51	0.58	0.85	615

Table 4.16 shows the results per class using the NorBERT models. Both models perform about the same as CNN and LSTM using word embeddings for this task.

4.3.2 Input: Description, title and other tags

Table 4.17 summarizes the results for the school year label prediction using description, title, and other tags as input to the models. The overall performance of the models is still lower than for the subject label problem, but higher than not using tags. Overall, NorBERT 2 performed better than the remaining models on all metrics except precision.

Table 4.17: Overall results for school year using description, title, and other tags. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.19	0.37	0.81	0.66	0.46	0.55	0.83
LSTM + FastText	0.24	0.47	0.81	0.64	0.57	0.61	0.86
CNN	0.21	0.43	0.82	**0.71**	0.49	0.58	0.86
CNN + FastText	0.24	0.47	0.82	0.69	0.55	0.61	0.87
NorBERT1	0.23	0.48	0.81	0.65	0.55	0.60	0.85
NorBERT2	**0.28**	**0.57**	**0.83**	0.64	**0.70**	**0.67**	**0.88**

The results per class label can be found in Table 4.18 for LSTM and CNN without pre-trained word embeddings, Table 4.19 for LSTM and CNN with FastText word embeddings, and Table 4.20 for NorBERT 1 and NorBERT 2.

Table 4.18: Results for school year label with LSTM and CNN using description, title, and other tags

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.91	0.34	0.50	0.88	0.93	0.48	0.64	0.92	29
2. trinn	0.91	0.40	0.56	0.85	0.86	0.48	0.62	0.89	25
3. trinn	0.82	0.32	0.46	0.82	0.93	0.46	0.62	0.90	28
4. trinn	0.70	0.23	0.35	0.77	0.78	0.47	0.58	0.86	30
5. trinn	0.67	0.13	0.22	0.79	0.56	0.16	0.25	0.84	31
6. trinn	0.57	0.13	0.21	0.75	0.56	0.16	0.25	0.81	31
7. trinn	0.67	0.29	0.41	0.74	0.62	0.15	0.24	0.79	34
8. trinn	0.61	0.78	0.69	0.77	0.70	0.71	0.70	0.82	79
9. trinn	0.60	0.81	0.69	0.75	0.64	0.61	0.62	0.79	79
10. trinn	0.67	0.84	0.75	0.80	0.67	0.73	0.70	0.80	90
VG1	1.00	0.21	0.35	0.77	0.82	0.49	0.62	0.83	57
VG2	0.73	0.31	0.43	0.81	0.83	0.48	0.61	0.83	52
VG3	0.33	0.02	0.04	0.77	0.62	0.26	0.37	0.79	50
Micro Avg	0.66	0.46	0.55	0.83	0.71	0.49	0.58	0.86	615

There is only a small increase in performance using LSTM when tags are added to the input, as shown in Table 4.18, compared to without tags in Table 4.14. Precision increased by 0.03 and recall by 0.01 overall and most of the classes only had a very small increase, with some classes even having a decrease in recall. The increase in recall is larger for CNN; however it is only an increase of 0.09 and the precision increased by 0.04. Some classes still have a very low recall of 0.16. All classes have an AUC above 0.7 for both models, with the lowest being 0.79 for CNN.

Table 4.19: Results for school year label with LSTM and CNN with FastText word embeddings using description, title, and other tags

Label	LSTM + FastText				CNN + FastText				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.86	0.62	0.72	0.91	1.00	0.59	0.74	0.95	29
2. trinn	0.80	0.48	0.60	0.88	1.00	0.56	0.72	0.95	25
3. trinn	0.75	0.43	0.55	0.88	1.00	0.46	0.63	0.92	28
4. trinn	0.71	0.50	0.59	0.85	0.75	0.50	0.60	0.90	30
5. trinn	0.58	0.48	0.53	0.87	0.67	0.32	0.43	0.88	31
6. trinn	0.59	0.52	0.55	0.85	0.75	0.29	0.42	0.83	31
7. trinn	0.57	0.50	0.53	0.82	0.71	0.35	0.47	0.82	34
8. trinn	0.63	0.75	0.68	0.77	0.64	0.58	0.61	0.77	79
9. trinn	0.58	0.72	0.64	0.74	0.58	0.65	0.61	0.74	79
10. trinn	0.65	0.76	0.70	0.76	0.69	0.66	0.67	0.81	90
VG1	0.66	0.51	0.57	0.84	0.59	0.67	0.63	0.86	57
VG2	0.59	0.37	0.45	0.80	0.70	0.50	0.58	0.83	52
VG3	0.73	0.32	0.44	0.79	0.72	0.52	0.60	0.84	50
Micro Avg	0.64	0.57	0.61	0.86	0.69	0.55	0.61	0.87	615

The recall increases when using LSTM and CNN with FastText word embeddings, as shown in Table 4.19; however, the precision decreases slightly for both models. The recall has also improved significantly from Table 4.15 where the input was description and title for both models. CNN and LSTM performs relatively evenly on this task.

Table 4.20: Results for school year label with NorBERT 1 and NorBERT 2 using description, title, and other tags

Label	NorBERT 1				NorBERT 2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.88	0.52	0.65	0.93	0.91	0.72	0.81	0.95	29
2. trinn	0.80	0.48	0.60	0.86	0.73	0.76	0.75	0.93	25
3. trinn	0.88	0.50	0.64	0.89	0.73	0.68	0.70	0.95	28
4. trinn	0.68	0.50	0.58	0.84	0.66	0.63	0.64	0.91	30
5. trinn	0.47	0.29	0.36	0.82	0.61	0.61	0.61	0.88	31
6. trinn	0.58	0.23	0.33	0.82	0.53	0.58	0.55	0.84	31
7. trinn	0.80	0.35	0.49	0.84	0.69	0.62	0.61	0.85	34
8. trinn	0.60	0.76	0.67	0.78	0.66	0.76	0.71	0.79	79
9. trinn	0.62	0.67	0.65	0.76	0.59	0.77	0.67	0.77	79
10. trinn	0.70	0.77	0.73	0.80	0.64	0.83	0.72	0.82	90
VG1	0.67	0.42	0.52	0.80	0.62	0.65	0.63	0.86	57
VG2	0.60	0.40	0.50	0.80	0.59	0.71	0.64	0.83	52
VG3	0.55	0.54	0.55	0.77	0.68	0.52	0.59	0.82	50
Micro Avg	0.65	0.55	0.60	0.85	0.64	0.70	0.67	0.88	615

Table 4.20 shows the results for class label using description, title, and other tags as input. NorBERT 1 performs very close to LSTM and CNN with word embeddings shown in Table 4.19. However, NorBERT 2 performs significantly better than the other models on this task for recall, and about the same for precision.

4.3.3 Input: Description, title, other tags and file text

Table 4.21 summarizes the results for the subject label prediction using description, title, other tags, and text from files as input to the models. The performance decreases for almost all metrics for all models, except CNN which has similar results to the case without file text, shown in Table 4.17. The LSTM and CNN models are run with maximum vocabulary from files and all models are run with input vector length of 500. The running time increased a bit for LSTM and CNN and more than quadrupled for the BERT models.

Table 4.21: Overall results for school year label using description, title, other tags, and file text. Best value per metric in bold.

Model	Accuracy			Precision	Recall	F1	AUC
	EMR	Hamming	Global	Micro Average			
LSTM	0.16	0.27	0.78	0.60	0.33	0.43	0.77
LSTM + Fasttext	0.15	0.36	0.78	0.59	0.43	0.50	0.83
CNN	**0.28**	0.45	**0.82**	**0.71**	0.49	**0.58**	0.86
CNN + Fasttext	0.18	0.39	0.81	0.70	0.43	0.54	**0.87**
NorBERT1	0.19	**0.46**	0.80	0.61	**0.56**	**0.58**	0.84
NorBERT2	0.21	**0.46**	0.81	0.64	0.52	0.57	0.86

The results per class can be found in Table 4.22 for LSTM and CNN without pre-trained word embeddings, Table 4.23 for LSTM and CNN with FastText word embeddings, and Table 4.24 for NorBERT 1 and NorBERT 2.

Table 4.22: Results for school year label with LSTM and CNN using description, title, other tags, and file text

Label	LSTM				CNN				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	1.00	0.28	0.43	0.87	0.93	0.45	0.60	0.93	29
2. trinn	0.88	0.28	0.42	0.83	0.88	0.28	0.42	0.89	25
3. trinn	0.83	0.18	0.29	0.81	0.83	0.36	0.50	0.87	28
4. trinn	0.75	0.20	0.32	0.77	0.64	0.30	0.41	0.86	30
5. trinn	0.00	0.00	0.00	0.68	0.67	0.26	0.37	0.85	31
6. trinn	0.00	0.00	0.00	0.65	0.64	0.23	0.33	0.83	31
7. trinn	0.00	0.00	0.00	0.63	0.67	0.24	0.35	0.79	34
8. trinn	0.53	0.62	0.57	0.65	0.69	0.71	0.70	0.80	79
9. trinn	0.55	0.67	0.61	0.68	0.69	0.68	0.69	0.80	79
10. trinn	0.64	0.74	0.69	0.74	0.76	0.76	0.76	0.82	90
VG1	0.67	0.14	0.23	0.70	0.72	0.46	0.56	0.82	57
VG2	0.50	0.02	0.04	0.67	0.58	0.29	0.38	0.79	52
VG3	1.00	0.04	0.08	0.63	0.68	0.42	0.52	0.80	50
Micro Avg	0.60	0.33	0.43	0.77	0.71	0.49	0.58	0.86	615

Table 4.22 shows the results for LSTM and CNN without word embeddings and shows that LSTM gives 0.00 precision and recall for years 5 to 7. This means that it gives no true positive predictions for these years. They did, however, have both false positive predictions, since precision is 0.00, and false negative predictions since recall is 0.00. The

performance for LSTM is quite a bit lower than without file text, as seen in Table 4.18. The CNN model on the other hand performs approximately the same as without file text.

Table 4.23: Results for school year label with LSTM and CNN with word embeddings using description, title, other tags, and file textt

Label	LSTM + FastText				CNN + FastText				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.79	0.66	0.72	0.93	0.84	0.55	0.67	0.95	29
2. trinn	0.73	0.44	0.55	0.91	0.76	0.52	0.62	0.94	25
3. trinn	0.59	0.46	0.52	0.90	0.76	0.57	0.65	0.92	28
4. trinn	0.59	0.43	0.50	0.88	0.75	0.70	0.72	0.91	30
5. trinn	0.57	0.42	0.48	0.88	0.60	0.19	0.29	0.89	31
6. trinn	0.59	0.42	0.49	0.82	0.60	0.19	0.29	0.85	31
7. trinn	0.58	0.32	0.42	0.78	0.75	0.26	0.39	0.84	34
8. trinn	0.58	0.43	0.49	0.69	0.65	0.33	0.44	0.76	79
9. trinn	0.53	0.42	0.47	0.67	0.60	0.33	0.43	0.73	79
10. trinn	0.64	0.48	0.55	0.68	0.75	0.51	0.61	0.80	90
VG1	0.67	0.28	0.40	0.80	0.70	0.49	0.58	0.85	57
VG2	0.54	0.48	0.51	0.77	0.77	0.52	0.62	0.84	52
VG3	0.48	0.46	0.47	0.78	0.57	0.54	0.56	0.83	50
Micro Avg	0.59	0.43	0.50	0.83	0.70	0.43	0.54	0.87	615

The results for CNN and LSTM with word embeddings is shown in Table 4.23. Now, the results for year 5 to 7 has increased significantly; however, it is still lower than when not using file text, shown in Table 4.19. The performance of the CNN model actually decreases when file text is added to the input.

Table 4.24: Results for school year label with NorBERT 1 and NorBERT 2 using description, title, other tags, and file text

Label	NorBERT 1				NorBERT 2				Support
	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC	
1. trinn	0.81	0.59	0.68	0.92	1.00	0.45	0.62	0.94	29
2. trinn	0.71	0.48	0.57	0.91	0.90	0.36	0.51	0.94	25
3. trinn	0.71	0.61	0.65	0.88	0.83	0.36	0.50	0.94	28
4. trinn	0.65	0.57	0.61	0.88	0.62	0.43	0.51	0.91	30
5. trinn	0.52	0.42	0.46	0.87	0.57	0.13	0.21	0.85	31
6. trinn	0.48	0.35	0.41	0.82	0.62	0.16	0.26	0.81	31
7. trinn	0.54	0.41	0.47	0.73	0.75	0.09	0.16	0.84	34
8. trinn	0.60	0.73	0.66	0.73	0.63	0.52	0.57	0.74	79
9. trinn	0.55	0.66	0.60	0.70	0.64	0.62	0.63	0.76	79
10. trinn	0.66	0.73	0.69	0.76	0.65	0.82	0.73	0.80	90
VG1	0.60	0.44	0.51	0.77	0.59	0.77	0.67	0.83	57
VG2	0.64	0.44	0.52	0.77	0.60	0.48	0.53	0.79	52
VG3	0.59	0.40	0.48	0.76	0.57	0.54	0.56	0.77	50
Micro Avg	0.61	0.56	0.58	0.84	0.64	0.52	0.57	0.86	615

Table 4.24 shows that when using file text, NorBERT 1 performs close to how it did when not using file text, shown in Table 4.20. NorBERT 2 on the other hand has a significant decrease in performance. Without file text in the input, all classes have a recall above 0.52; however, now it has decreased to 0.09. The precision in both cases are similar.

All models shows that adding file text does not improve the results, and in the case of the BERT models, increasing the input vector length from 250 to 500 significantly increases the time the models took to run by about 4 times.

Chapter 5

Discussion

This chapter aims to go through and discuss the results presented in Chapter 4 using the method and dataset presented in Chapter 3.

In almost all experiments, one of the NorBERT models performed the best. For some, this was closely followed by LSTM and CNN using FastText word embeddings. Overall for almost all experiments, using word embeddings helped get better results for both LSTM and CNN. This is likely due to the pre-trained model and pre-trained word embeddings being able to relate new words to similar words that have already been used. Training a model from scratch would give a new meaning to every new word. A lot less preprocessing was required when using one of the NorBERT models as they could be fed raw text, while the CNN and LSTM models required a lot of cleaning.

The CNN models were the quickest to train for all experiments. They were both quick per epoch and required relatively few epochs depending on the input and hyperparameters. LSTM was also relatively quick per epoch but slower than CNN. In addition, it generally required more epochs than CNN. The BERT models were relatively slow per epoch compared to CNN and LSTM; however, this required much fewer epochs, often only between 3-6 epochs. It was still slower than LSTM, even though it had more than 30 epochs before training finished for some experiments.

There was a significant class imbalance for the subject label prediction problem. Some classes had very few records, and the labels "Musikk" and "Mat og helse" was never predicted using any of the models. Some ways to deal with class imbalance are undersampling, oversampling, or class weights. It was decided not to do undersampling, which is not using all records from the majority classes, as the dataset is relatively small, which means there would not be enough information to predict the majority classes either. Oversampling was not chosen either as this would require picking copies of the minority classes, and as there were very few of some of these classes, there would be a lot of identical records. Using class weights in the model would give similar results to oversampling. In this case, less weight would be given to the majority classes making these perform worse than when not using class weights. The minority classes are still too small and would still perform poorly, but the medium classes may perform better. Another method to deal with an imbalanced dataset could be augmentation, which is to create new augmented records for the minority classes. This is quite complex when it comes to multi-label classification, as each record can be relevant for multiple classes, and it is difficult to know what part of the text is relevant for each label. This could be tested further to get a more balanced dataset. However, the best solution would be to get more training data for the minority classes in the dataset.

The models generally performed better when predicting subject labels than school year labels. There may be several reasons for this. One could be that it is easier to relate input text to a subject than to a school year as the same subject may be taught in several different school years, such as English, Norwegian, and Maths. For example, given the word 'multiplication', the algorithms could probably relate this to Maths; however, it is not easy to know what school year this would be relevant for as this is a word that is used for several school years. Another reason could be that more resources were tagged with several school years, as shown in Figure 3.4. While most resources only have one subject label, they have more than one school year. A few were even tagged with all 13 school years. This gave a lot more resources per label but also made it more difficult to distinguish between labels. As opposed to the subject labels, the school year labels are a sequence from 1 to 13, and the content is most often tagged with a sequence of school years, such as 8. - 10. or VG1 - VG3 (11. - 13.). To get better results for the school year prediction, this sequence could be exploited by, for example, looking into a regression model instead of a classification model. Another solution could be to make a model that gives a higher score if a predicted school year is close to the actual prediction in the sequence to make it learn that a close school year is better than one far away.

The binary performance measures: precision, recall, and F1-score, were given for each class, and a micro average across the classes. The output from the models was given as a probability between 0 and 1 per label. A threshold of 0.5 was used to calculate these metrics. The AUC score was calculated using the raw output probabilities and measured the performance independent of this threshold. It was seen that some classes performed poorly on the former metrics; however, they may have a high AUC which means that the chosen threshold may not be the best threshold for this class. Choosing a different threshold may make the results better for some classes; however, the threshold is individual per class, making it difficult to generalize across all classes. The majority classes would have an ideal threshold closer to 0.5 as the results for these were generally better than minority classes. The minority classes would have a lower ideal threshold as the recall was low, which means that few predictions were made, neither true nor false. It could be possible to choose the threshold based on the size of the class, with a larger threshold for majority classes than minority classes, to see whether this could improve the performance.

The threshold can also be utilized to balance between a high precision and a high recall. In this case, this balance is two-sided. Firstly, it is important to have a high recall to ensure the teachers get relevant content. However, a high recall could be more important to ensure that the teachers do not miss content that could potentially be relevant due to false negative predictions. With a low recall, the model chooses not to take the risk of predicting an incorrect label. A higher threshold would increase the precision as there would be fewer false positive predictions; likewise, decreasing the threshold would give a higher recall as more predictions would be made, decreasing the number of false negative predictions.

The results became worse when also using file content as input to the models. This was very sensitive to the maximum vocabulary chosen for CNN and LSTM and the maximum length of the input vector. A large vocabulary and large input vector gave worse results than a smaller vocabulary and shorter input vector. This is likely due to introducing a large number of words that are only used once that will not add value to the prediction, only noise. All words present in the files were used in the vocabulary input to the models. It could be beneficial to regard the input vocabulary and vector length as hyperparameters that could be tuned to see if there are optimal values where the extra file text would add value to the models.

Another disadvantage of using file text was that the run time of the models, and especially the BERT models, increased significantly when doubling the vector length. When increasing the input vector length from 250 to 500, the run time of the BERT models more than quadrupled to around 20 min per epoch on a personal laptop, and these were relatively slow with the shorter input vector. The preprocessing time using LSTM and CNN also increased as the preprocessing was performed on much more text.

The best hyperparameters for each model were chosen based on the subject label problem with only description and title as input. These may not be the best hyperparameters for the class label problem or the problems using more information as input. However, this was chosen to be able to compare the models with the same hyperparameters.

Chapter 6

Conclusion

This thesis tests different neural networks on a multi-label classification problem to predict tags for educational content. The predicted tags are the subject, which consists of 11 different labels, and the school year, which consists of 13 different labels. The problem is solved as a multi-label problem as each learning resource can have multiple tags related to either subject or school year. The dataset is collected from the Addito website metadata and is primarily in Norwegian.

The results show that the chosen models are able to predict subject labels with sufficient instances in the dataset quite well using description, title, and other tags as input; however, none of them are able to predict the two classes with the least instances in the dataset. Using pre-trained word embeddings for LSTM and CNN helped the results significantly, and the best results were obtained using pre-trained BERT models, called NorBERT, trained by the Language Technology Group at the University of Oslo [33].

None of the models are able to predict the school year labels very well with any of the input data provided. This is likely due to several learning resources being tagged with a large number of tags for school year and that the input texts are not sufficiently related to the year it is taught in. Some alternative methods that can be explored are presented below in Section 6.1.

6.1 Further Work

This thesis tests a few different models for the classification of educational content based on the dataset provided by the Addito metadata. Some further points that could be researched are presented below.

- Train on a larger dataset. The dataset was too small for any of the models to predict some of the minority classes. The easiest solution is to add more training data, especially for the minority classes but also the other classes. Training data could be added from partner sites. Data augmentation methods such as SMOTE (Synthetic Minority Oversampling Technique) for multi-label classification could be utilized to create augmented data for the minority classes.
- The school year label classification models did not perform as well as the subject label models. A few options that could be explored are to split the school years into small groups of three or four years and classify them into these groups instead of individual school years. Alternatively, as the school years are a sequence, it could be an option to look into a regression model or a model that gives a higher score if the predicted school year is close to the actual school year. A third option could be to make a joint model for school year and subject. This could possibly learn the dependency that some subjects are only taught in some school years.
- The models are sensitive to the thresholds chosen for the evaluation metrics. A further study could explore the sensitivity of the thresholds and if there is any merit to using individual thresholds per class.
- This thesis got worse performance when using text from files; however, the vocabulary and input vector length was not optimized. The optimal size of these parameters could be investigated along with the use of more advanced models. Some options can include models joint models for both subject and school year to exploit the correlation between subject and school year, or a multi-modal model that can also utilize images from the files, and also links to external learning resources and videos.

Bibliography

[1] Ramon Quiza and J. Davim. *Computational Methods and Optimization*, pages 177–208. 01 2011. ISBN 978-1-84996-449-4. doi: 10.1007/978-1-84996-450-0.

[2] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019. URL https://arxiv.org/abs/1912.05911.

[3] MK Gurucharan. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network, 2020. URL https://www.upgrad.com/blog/basic-cnn-architecture/.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

[6] Wikimedia Commons. File:Roc curve.svg, 2021. URL https://commons.wikimedia.org/wiki/File:Roc_curve.svg. Online; accessed 22-May-2022.

[7] NorBERT. Norwegian Large-scale Language Models: NorBERT, 2022. URL http://wiki.nlpl.eu/Vectors/norlm/norbert.

[8] Michael Simonson. Britannica: Distance Learning, 2020. URL https://www.britannica.com/topic/distance-learning.

[9] Cathy Li and Farah Lalani. The COVID-19 pandemic has changed education forever. This is how, 2020. URL https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/.

[10] Neddy, 2022. URL https://www.neddy.no/.

[11] Addito, 2022. URL https://addito.no/.

[12] LTG. Language Technology Group Oslo, 2022. URL https://www.mn.uio.no/ifi/english/research/groups/ltg/.

[13] Jason Brownlee. Multi-Label Classification with Deep Learning, 2020. URL https://machinelearningmastery.com/multi-label-classification-with-deep-learning/.

[14] NOKUT. General Information about education in Norway, 2022. URL https://www.nokut.no/en/norwegian-education/general-information-about-education-in-norway/.

[15] UDIR. The Norwegian Directorate for Education and Training, 2022. URL https://www.udir.no/in-english/.

[16] My Computer Career. What are Online Learning Platforms? URL https://www.mycomputercareer.edu/news/what-are-online-learning-platforms/. Accessed: 2022-05-05.

[17] Tech Target. Natural language processing (NLP). URL https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP. Accessed: 2022-05-04.

[18] Sunny Srinidhi. Stemming of words in Natural Language Processing, what is it?, 2020. URL https://towardsdatascience.com/stemming-of-words-in-natural-language-processing-what-is-it-41a33e8996e2.

[19] Jason Brownlee. What Are Word Embeddings for Text?, 2017. URL https://machinelearningmastery.com/what-are-word-embeddings/.

[20] Timo Böhm. The General Ideas of Word Embeddings, 2018. URL https://towardsdatascience.com/the-three-main-branches-of-word-embeddings-7b90fa36dfb9.

[21] colah's blog. Understanding LSTM Networks, 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

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

[23] Rani Horev. BERT Explained: State of the art language model for NLP, 2018. URL https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[24] NLTK. Natural Language ToolKit documentation, 2022. URL https://www.nltk.org/.

[25] Spacy. Trained Model and Pipeline for Norwegian Bokmål, 2022. URL `https://spacy.io/models/nb`.

[26] Keras. URL `https://keras.io`.

[27] Tensorflow. URL `https://www.tensorflow.org/`.

[28] Scikit-Learn. Machine Learning in Python, 2020. URL `https://scikit-learn.org/stable/`.

[29] iterative-stratification, 2021. URL `https://github.com/trent-b/iterative-stratification`.

[30] Hugging Face. The Hugging Face Home Page, 2022. URL `https://huggingface.co/`.

[31] NorLM. Norwegian Large-scale Language Models, 2022. URL `http://wiki.nlpl.eu/Vectors/norlm`.

[32] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

[33] Andrey Kutuzov, Jeremy Barnes, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. Large-scale contextualised language modelling for norwegian, 2021. URL `https://arxiv.org/abs/2104.06546`.

[34] Hugging Face. Summary of the Tokenizers, 2022. URL `https://huggingface.co/docs/transformers/tokenizer_summary`.

[35] NLPL. Word Embeddings Repository, 2022. URL `http://vectors.nlpl.eu/repository`.

[36] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 145–158, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23808-6.

[37] Hyperas. Keras + Hyperopt: A very simple wrapper for convenient hyperparameter optimization. URL `https://maxpumperla.com/hyperas/`. Accessed: 2022-05-15.

[38] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. Technical report, 2010.

[39] Noam Bressler. How to Check the Accuracy of Your Machine Learning Model, 2021. URL `https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/`.

[40] Google Machine Learning Crash Course. Classification: ROC Curve and AUC, 2020. URL `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc`.